ALBERT-LUDWIGS-UNIVERSITÄT FREIBURG
WIRTSCHAFTS- UND VERHALTENSWISSENSCHAFTLICHE FAKULTÄT

Sustainable and Efficient Management of Enterprise Information Systems:

Strategies & Operation Modes

Inaugural-Dissertation
zur
Erlangung der Doktorwürde
der Wirtschafts- und Verhaltenswissenschaftlichen Fakultät
der Albert-Ludwigs-Universität Freiburg i. Br.

vorgelegt von
Markus Adolf Hedwig
geboren in Brühl

WS 2011/2012

Druckdatum: 21.02.2012

URL: http://www.freidok.uni-freiburg.de/volltexte/8472

Albert-Ludwigs-Universität Freiburg im Breisgau
Wirtschafts- und Verhaltenswissenschaftliche Fakultät
Kollegiengebäude II
Platz der Alten Synagoge
79085 Freiburg

Dekan:	Prof. Dr. Dieter K. Tscheulin
Erstgutachter:	Prof. Dr. Dirk Neumann
Zweitgutachter:	Prof. Dr. Günter Müller
Datum des Promotionsbeschlusses:	21.02.2012

ACKNOWLEDGEMENTS

First and foremost, I would to thank Prof. Neumann for all his mentoring, encouragement, academic advice, and all the time he invested in me throughout the last years.

Over the last years, I had the honor to work and discuss with various inspiring people who significantly impacted my research work and supported me with their experience and resources. I would like to express my sincere gratitude to Prof. Günter Knieps from the University of Freiburg, Prof. Teo Yong Meng from the National University of Singapore, Prof. Günter Müller from the University of Freiburg, Prof. Calton Pu from the Georgia Institute of Technology, Dr. Frank Schulz from SAP Research and the ValueGrids project team, Prof. Guido Schryen from the University of Regensburg, Prof. Simon See from Nvidia, as well as Prof. Christof Weinhardt and his team from the Karlsruhe Institute of Technology.

In particular, I would like to thank Simon Malkowski for sharing with me the ups and downs of the last years and for all his support in my research. I would also like to thank all the other colleagues at the chair of Information Systems Research at the University of Freiburg. I would like to especially say “thank you” to Carla Li-Sai for her selfless support and care.

Last but not least, I would like to thank my parents Almuth and Rudolf Hedwig for their love and support, as well as my girlfriend Susanne Langer who always supported me in my work and kept me away from working when I did not have to.

TABLE OF CONTENTS

CHAPTER I

INTRODUCTION

1. Introduction

The operation of powerful information systems or information technology in general has become one of the key success factors in modern enterprises. Today, Enterprise Information Systems provide real-time information about all business processes along the value chain and enable cost-effectiveness as well as fast communication with customers and suppliers. Nonetheless, with the ever increasing demand and necessity for more complex and advanced information systems, corporate decision makers do not only face increasing costs of IT, but also have to become aware of its substantial environmental impact (Singh et al. 2007). Already in 2005, servers worldwide consumed energy worth $ 7.2B (Koomey 2007); the global IT industry has been responsible for about 2% of the global carbon dioxide emissions. As a consequence, achieving a high degree of efficiency in information technology operation has become a top priority in corporate governance.

The benefits of information systems for today's business practice clearly outweigh the aforementioned problems. Furthermore, considering new trends such as Smarter Planet initiatives (IBM 2011) or the information technology demand of the emerging markets, the volume and importance of IT will even increase more. A recent study estimates the growth of the server market by 5% every year (Gartner 2010). Consequently, the solution to the increasing resource consumption of IT is not on the demand side. Considering additional factors such as increasing resource prices (e.g. electrical energy) and the growing power demand of servers, the economic and environmental problems will persist or even get worse (Singh et al. 2007). Therefore, the sensible utilization of IT will pave the way towards sustainable and efficient operation modes.

A closer examination of common IT operation practices reveals a wasteful handling of computing resources. Nowadays, most corporate datacenters operate their IT infrastructure with a low average utilization. For instance, large scale internet services may have only an average utilization of about 20% to 30%, whereby the utilization seldom exceeds 50% (Luiz André Barroso & Hölzle 2007). Consequently, most of the infrastructure remains weakly utilized or idle and its potential computing power is wasted. The reasons for this deficiency are manifold (T. Velte et al. 2009; Schulz 2009; Bailey 2009). For example, ecommerce websites or enterprise information systems are usually scaled to handle peak demands. Nonetheless operation often faces highly volatile workloads, as usage is frequently correlated to external factors such as time of day (e.g. office hours). Furthermore, most information system architectures are static, so that excessive reserve resources are provisioned to meet future user demands as well as for failure safety. Moreover, these systems are designed with focus on functionality and QoS (Quality of Service) requirements rather than on cost considerations.

The origin of the inefficiencies in computing may be reasoned in the cost structure of information systems. Until the beginning of the century the main costs in IT were initial investment, licenses, and administration expenditures (Singh et al. 2007). However, already today, operational expenditures (e.g. energy cost) for IT exceed the initial investment costs. This problem is well known, and so called Green IT initiatives already offer a variety of best practices to reduce the economic and environmental impact of IT (T. Velte et al. 2009; Schulz 2009). Popular approaches comprise the use of highly energy efficient equipment and the consolidation and virtualization of servers. Nonetheless, while infrastructure enhancements do not directly benefit the weak utilization of datacenters, concepts for better resource utilization are often limited to small instances of such systems. In particular for large systems these concepts are not applicable, as these systems are often distributed and operated on several servers.

One approach towards highly efficient and agile operation of large information systems is operating elastic information systems in cloud computing environments. This new software design concept allows resource adaptive operation modes, and leverages the potential of the cloud for instantaneous resource provisioning. More specifically, this software design allows to add and remove computing resources from the system at runtime and thus to adapt the infrastructure size to the user demand online. If implemented correctly, it allows for highly efficient operation modes while at the same time provides a continuously high quality of service. Generally, this approach can be implemented in any application, regardless of whether it is operated in corporate datacenters or remotely at a service provider. Although there is heterogeneity of hardware, software and user requirements, the operation of these systems is non-trivial, and elastic operation concepts have to be adapted to the specific use case requirements.

Apparently, given technological advances and changing customer requirements, efficient and sustainable management has evolved into a hard task. However, larger companies, in particular web-companies, may be able to develop and operate highly efficient systems. For instance Google developed a highly efficient system layout for their search engine (Barroso et al. 2003) and Facebook is developing its own highly energy efficient server layout (Open Compute Project 2011). Nonetheless, the efficient and sustainable management of IT requires high expertise and practice which is often beyond the financial capabilities and expertise of small and midsized companies. With the emergence of stable cloud solutions however, they have obtained an alternative to their established IT in-house operation. With the global availability of inexpensive high speed internet connections, companies can purchase their computing demand online on a pay as you go basis. This turns the former assets of IT into expenses (Carr 2005) and allows enterprises to focus on their core business while at the same time provides their computing demand at lower cost. As a result of this trend, a new industry has emerged. Today various service products have already reached market maturity, such as Infrastructure-, Platform- or Software as-a-Service products, and are slowly transforming into a real alternative and the future of enterprise computing.

While the potential of this new way to provide corporate computing demand has been widely accepted, many significant challenges still need to be addressed. Simply moving corporate systems into cloud environments does not help and may even lead to higher cost of operation (Risch & Altmann 2008). In order to facilitate the sensible utilization of IT, a full redesign of the current practice is necessary, including the adaptation of existing systems to the new computing paradigms. Elastic applications are however particularly hard to manage. Volatile user workload, significant reconfiguration lead times, as well as non-trivial performance characteristics make it exceptionally hard to operate the systems efficiently and QoS aware. For instance, if a system is operated with high resource utilization, then unforeseen workload increases or hardware failures might immediately result in QoS shortcomings. Accordingly, the Quality of Service requirements, specified in the service contract, need to be well defined and included into the management concept. In the service world, Service Level Agreements (SLAs) have become the common practice to define the terms of business. They include the Service Level Object (SLO), defining the technical parameter of service operation (e.g. response time) as well as the financial agreement (i.e. service price and penalties). However, the correct and business value aware design of these contracts is still unsolved.

Today, enterprises still have reservations to migrate their systems to this new technology (Armbrust et al. 2010). In particular in public service markets, the offerings of the providers are mainly used for low-cost or ad-financed services. Next to security concern, the main problem is that information systems often belong to the critical infrastructure of a company. The risk of failure and loss of business is valued significantly higher than the saving potential. However, given the fierce cost pressure enterprises are forced to adapt to this new technology (Bailey 2009). Already in 2010 half of all enterprise applications were operated in virtual environments (Langenkamp 2010).Furthermore, Gartner predicts that by 2013, 90% of all e-commerce web sites will at least use one SaaS-based service (Gartner 2008). Hence, it will only be a matter of time until these operation modes will be the common standard in corporate IT.

Based on the current state of corporate IT, this thesis develops various models and concepts for the efficient and sustainable operation of enterprise information systems. The complexity and heterogeneity of IT requires the design and implementation of holistic models for the management and operation of such systems. Thus, in order to achieve maximum efficiency, all possible aspects of the information system environment need to be included in the operation strategy such as the user behavior, the performance characteristics of the system and configuration lead times, as well as the QoS requirements.

The remainder of this chapter presents the research questions, summarizes the five chapters and gives a conclusion and an outlook. Afterwards, in the subsequent chapters, the five research articles are presented.

2. Research Outline

This section introduces the five core research questions of this thesis. Today, corporate datacenters face increasing cost of operation (in particular energy costs) while at the same time they suffer from an averagely weak utilization. However, the efficient operation of larger enterprise information systems is non-trivial due to factors such as significant reconfiguration lead-times, complex performance characteristics as well as volatile user workload. Elastic system designs and resource adaptive operation modes are a promising solution towards highly efficient operation of large information systems. Nonetheless, the efficient and QoS aware operation requires a thorough understanding of the performance characteristics and user behavior. This can be summarized in the first research question:

Question 1: *How can the various factors of influence in a specific information system operation scenario be integrated into a holistic model to reduce the resource consumption of corporate IT?*

Given the increasing complexity and size of modern information systems as well as the new ecological awareness in the spirit of Green IT, the sustainable and at the same time reliable operation of information systems has evolved into a difficult challenge. Even worse, considering additional factors such as infrastructure and software heterogeneity, divergent user behavior, different QoS requirements as well as the customer demand for strict binding SLAs, the implementation of autonomic and efficient operation concepts has become extremely challenging. Thus sustainable operation modes require on the one hand a holistic evaluation of all factor of influence at runtime, and on the other hand, must be highly flexible in order to adapt them to any operation scenario. Furthermore, the validity of such operation concepts has to be proven empirically.

Question 2: *Based on the findings and insights of the first model, how can this model be generalized and validated to facilitate highly economic efficient operation modes for a broad range of infrastructures and applications?*

Elastic operation modes for information systems allow highly efficient and sustainable operation. Nonetheless, most management concepts incorporate the QoS requirements defined in the SLA only statically. More correctly, reconfiguration and adaptation decisions are executed on a set of fixed rules, based on the QoS and economic parameters of the agreement. Thus the capability of the system to react to unforeseen effects is limited. For instance, if an information system only achieved a weak SLA compliance at the beginning of the contract lifetime, the provider should switch a less aggressive and thus more expensive operation mode to ensure SLA compliance. Hence, in order to operate service systems with maximum efficiency according to their value, the SLA must be included into the operation strategy and the SLA compliance must be evaluated online in order to adapt Service Level Objectives continuously. The correct implementation of such a dynamic SLA management system allows a high degree of efficiency during operation.

Question 3: *How can the dynamic and flexible management of Service Level Agreements at runtime increase efficiency and reliability of service operation?*

The successful adoption of the cloud computing paradigm together with the implementation of elastic systems allows for efficiency in corporate computing never seen before. Nonetheless, outsourcing the whole infrastructure to a service provider is in many cases undesired and may even lead to higher cost of operation. This leads to the question of the optimal infrastructure size of a corporate system. More specifically, which share of the servers should be operated in-house and which part should be operated remote in order to achieve high economic efficiency in operation.

Question 4: *From an economic perspective, what is the optimal size of corporate infrastructures?*

With the shift in corporate computing to the service oriented computing paradigms and the increasing competition on the service markets, service providers are forced to offer their services at competitive price. Consequently, they have to achieve a high degree of efficiency in operation which leads to the tradeoff between cost of operation and system reliability. On the customer side, outsourcing services to a provider has a high economic risk (e.g. risk of system failure and resulting loss of business) and hence they demand compensating penalties in case of non-compliance. Thus, in order to successfully participate in the service market, service providers have to understand the relation between the QoS and the price and penalty of a service in order to derive profit optimal and risk adjusted operation modes.

Question 5: *What is the relation between the Quality of Service and the cost of operation and how can this relation be integrated into the design process of Service Level Agreement?*

2.1. Thesis Structure

According to the research question above, each of the following chapters focus on a different aspect of enterprise information system operation. Nonetheless, all chapters address the topic of this thesis: sustainable and efficient management of enterprise information systems.

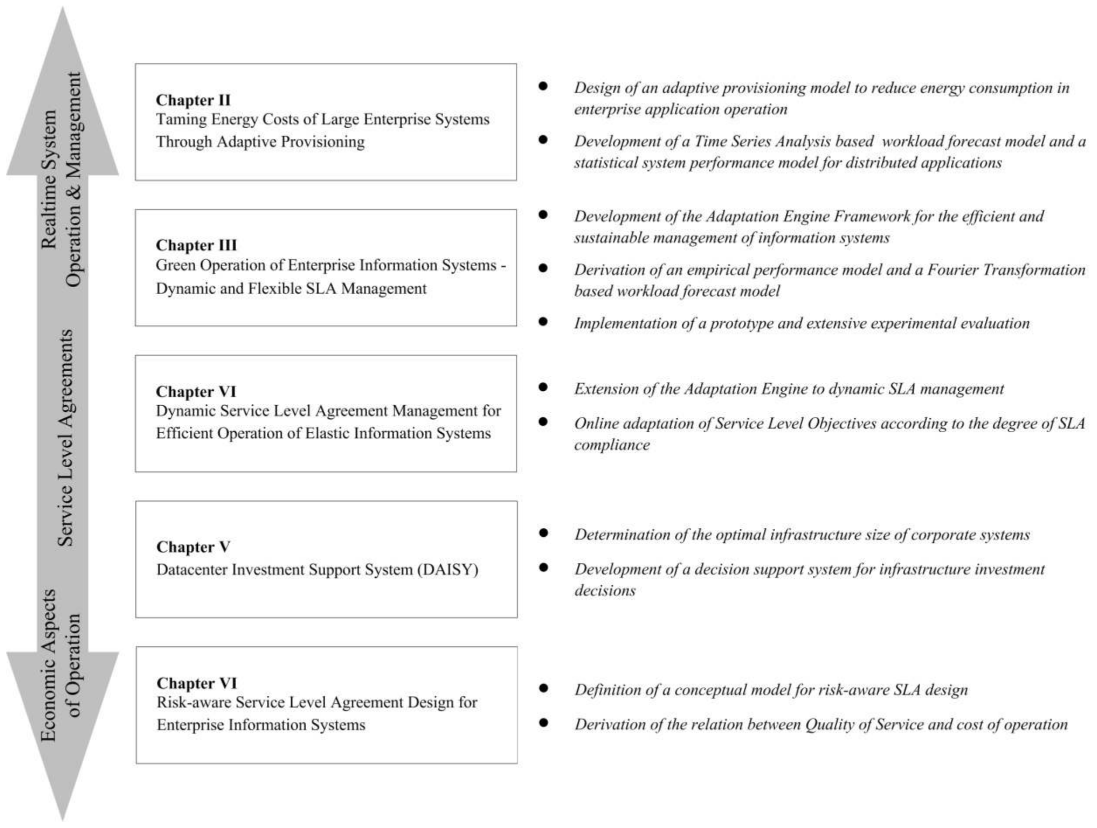

Realtime System Operation & Management
Service Level Agreements
Economic Aspects of Operation
Chapter II
Taming Energy Costs of Large Enterprise Systems Through Adaptive Provisioning
Design of an adaptive provisioning model to reduce energy consumption in enterprise application operation
Development of a Time Series Analysis based workload forecast model and a statistical system performance model for distributed applications
Chapter III
Green Operation of Enterprise Information Systems - Dynamic and Flexible SLA Management
Development of the Adaptation Engine Framework for the efficient and sustainable management of information systems
Derivation of an empirical performance model and a Fourier Transformation based workload forecast model
Implementation of a prototype and extensive experimental evaluation
Chapter VI
Dynamic Service Level Agreement Management for Efficient Operation of Elastic Information Systems
Extension of the Adaptation Engine to dynamic SLA management
Online adaptation of Service Level Objectives according to the degree of SLA compliance
Chapter V
Datacenter Investment Support System (DAISY)
Determination of the optimal infrastructure size of corporate systems
Development of a decision support system for infrastructure investment decisions
Chapter VI
Risk-aware Service Level Agreement Design for Enterprise Information Systems
Definition of a conceptual model for risk-aware SLA design
Derivation of the relation between Quality of Service and cost of operation

Figure 1: Thesis Structure

Figure 1 depicts the structure of the thesis. The thesis comprises concepts for the sustainable and efficient management of enterprise information systems. While the first chapters focus on information system operation, the later chapters investigate economic aspects of corporate computing. More specifically, the second chapter introduces Tecless, a conceptual model for the efficient operation of enterprise applications in corporate datacenters. Based on a holistic evaluation of the system and workload characteristics, this model enables resource adaptive and QoS aware operation modes. The third chapter introduces the Adaptation Engine, designed for the efficient and QoS aware operation of enterprise information systems in cloud environments. In contrast to Tecless, the Adaptation Engine facilitates a modular design and can be adapted to a broad range of infrastructure and software scenarios. The findings are supported with large scale experiments executed in a test environment. The fourth chapter of the thesis presents one extension of the Adaptation Engine to dynamically incorporate the Service Level Agreements into the operation strategy. By continuously monitoring the degree of SLA compliance and the automatic adaptation of the Service Level Objectives, this model enables highly efficient and risk-aware operation of information systems. Afterwards, chapter five and chapter six present economic aspects of information system operation. More concretely, the fifth chapter presents a model for determination of the optimal infrastructure size for in-house operated systems. Based on a user behavior analysis, the system determines the optimal infrastructure size and supports decision makers in determining the optimal infrastructure size. Finally, the sixth chapter presents a model for the economical and risk aware design of SLA's. More concretely, the model establishes the relation between the Quality of Service and the cost of operation and thus helps to derive profit-optimal and risk-adjusted operation strategies for information systems.

2.2. *Summary: Taming Energy Costs of Large enterprise Systems through Adaptive Provisioning*

The second chapter presents the Tecless model which has been published in the proceedings of the *International Conference on Information Systems 2009*. Nowadays, one of the most pressing concerns in modern datacenter management is the rising cost of operation. Therefore, reducing variable expense, such as energy cost, has become a number one priority. However, these systems are commonly subjected to highly volatile workload processes and characterized by complex performance characteristics. This chapter explicitly addresses this challenge and presents an adaptive provisioning methodology which combines a low-level technical perspective on distributed systems with a high-level treatment of workload processes. More concretely, this chapter presents an iterative model to accurately describe the performance characteristics of enterprise systems depending on the workload. The performance model is based on the statistical bottleneck detection model, which has been designed to identify performance limiting components in a distributed system. It allows determining the optimal system configuration for any given workload. Furthermore, this chapter introduces a modified time series workload forecast model to predict the near future workload level based on the past user behavior. This enables the initiation of reconfiguration decisions before the system faces the predicted workload level. Finally, in this chapter a provisioning algorithm is derived, which optimizes hardware configurations according to the performance demand of the users. The analysis shows that the model has the potential to reduce operational energy consumption up to 25 percent while retaining the same quality-of-service. Given the lower power consumption, the profits of the datacenter operators may substantially increase while the impact on the environment is reduced.

2.3. Summary: Green Operation of Enterprise Information Systems: Dynamic and Flexible SLA Management

The third chapter introduces the Adaptation Engine Framework which is an extension of the previously presented Tecless model. A preliminary version of this chapter has been published in the proceedings of the *International Conference on Information Systems 2010*. The growing awareness that Green Information System (IS) solutions, which contribute to sustainable business processes, secure a long-lasting competitive advantage has increasingly focused corporate transformation efforts on the efficient and effective usage of Information Technology (IT). In this context, the chapter provides a green perspective on enterprise IS operation and introduces a novel IS adaptation framework that harmonizes green goals with the business value chain. The Adaptation Engine framework concretely targets elastic n-tier applications with dynamic on-demand cloud resource provisioning for component servers (e.g., application and database servers). The framework forecasts future user behavior based on historic data, analyzes the impact of workload on system performance based on a non-linear performance model, analyzes the economic impact of different provisioning strategies, and derives an optimal operation strategy. More generally, the adaptation engine optimizes IT system operation based on a holistic evaluation of the key aspects of the business value chain (e.g., system usage patterns, system performance, and Service Level Agreements) in accordance with the Green IS paradigm. The modular design of the framework allows its adaptation to various scenarios in the service world. The evaluation of the engine prototype, based on a real production system workload trace, is carried out in a custom test infrastructure (i.e., cloud testbed, n-tier benchmark application, distributed monitors, and control framework). In the evaluation, the practicability, IS optimization potential, green effectiveness, and business value chain orientation is systematically investigated. Furthermore the evaluation indicates that the integration of the adaptation engine allows flexible IS operation with up to a 46 percent lower cost of operation.

2.4. Summary: Dynamic Service Level Agreement Management for Efficient Operation of Elastic Information Systems

The fourth chapter extends the Adaptation Engine framework with a stateful and risk aware operation management model for information systems. This chapter has been published in the proceedings of the *International Conference on Information Systems 2010*. In the service world, SLAs have become the common standard to define the QoS requirements as well as the financial arrangements between two parties. In today's practice however, SLAs are mainly used during the service design and provisioning process. Thus the Service Level Objectives and economic parameters of the agreement are translated into a static set of rules which are used for the management of the service at run-time. As a result, service providers face the challenging trade-off between cost of operation and risk of SLA violation. For instance, by providing more resource to a system, the risk of violating an SLA significantly decreases. Rationally, this decision should be based on the price of a service and the penalty in case of non-compliance. However, today this decision is mainly done manually during design time and consequently the operation strategy is not adapted at runtime according to the degree of SLA compliance. For instance, if the system faces minor performance limitations at runtime, the operation strategy is not adapted accordingly. The dynamic SLA management model provides an autonomic concept to this challenge. Based on the previous framework, this extension evaluates the SLA compliance at run time and adapts the SLOs objectives in real-time according to the economic parameters. By this means, it both derives a risk adjusted and dynamic operation strategy according to the value of the service as well as reacts to changes in its environment instantaneously. This dynamic view on SLAs significantly reduces the risk of violating SLAs at runtime based on wrong or outdated operational decisions. Compared to the initial configuration of the Adaptation Engine, this extended version provides comparable efficiency in operation with a significantly lower risk of SLA violations.

2.5. *Summary: Datacenter Investment Support System*

The fifth chapter introduces the datacenter investment support systems, which helps datacenter operators to determine the optimal size of their infrastructure. This chapter has been published in the proceedings of the *43rd Hawaii International Conference on System Sciences*. While the cloud computing paradigm provides the technological foundation for previously unmatched efficiency in information systems, its economic implications are complex and significant. Bringing the benefits of this state-of-the-art technology to corporate datacenters requires a systematic cost analysis of current best-practice in conjunction with the new possibilities. This chapter addresses this challenge and presents a novel resource management model based on marginal cost of computing. DAISY (Datacenter Investment Support System) aids in deriving the optimal investment strategy for large enterprise computing infrastructures taking into account all purchase and rental options. The model-based framework is evaluated through the analysis of different time discrete and continuous scenarios. The main contribution is the thorough economic analysis of the dependencies between static and adaptive operation modes for modern elastic information systems

2.6. *Summary: Risk-Aware Service Level Agreement Design for Enterprise Information Systems*

The last chapter presents a model for the risk-aware SLA design for enterprise information systems. This chapter has been published in the proceedings of the *45rd Hawaii International Conference on System Sciences*. Effective information systems have become key to sustainable business practices. However rising costs of operation and a growing system complexity are driving the search for a more efficient delivery of corporate computing. New technologies in the emerging service world such as cloud computing provide powerful alternatives to traditional IT operation concepts. Nonetheless, executive decision makers still have reservations about migrating to this new technology. In addition to security concerns, a key issue is the still prevailing lack of strict SLAs in these service offerings. In fact, service providers hesitate to offer strictly binding SLAs because assessing economic risk exposure is a major challenge. This chapter presents a novel model for sustainable SLA design for enterprise information systems. The model combines various state-of-the-art concepts from the field of system management and balances the failure risk with the cost of operation. More concretely, this chapter presents a holistic overview over all aspects of modern service operation and their relations. Furthermore, based on this integrative model, the chapter offers interesting insights into the economics of service operation. For instance, the model derives the relationship between the cost of operation and the probability of SLA compliance. In particular, this chapter shows that the deliberate choice of a higher operational risk may lead to higher provider profits. Furthermore, given a set of Service Level Objectives and a desired penalty, the model can determine the risk-neutral price of a service. Additionally, the model can be used to assess if a client request for services has an expected positive profit, and which is the most suitable infrastructure to operate the service.

3. Conclusion

This thesis presented five research chapters dealing with different aspects of efficient enterprise information system operation. The chapters comprised concepts for the determination of the optimal infrastructure size, Service Level Agreement design and operation strategies for elastic systems. Throughout the chapters, the thesis showed that the sensible management of information systems helps to significantly increase the efficiency in operation without unreasonably compromising QoS requirements.

In summary, the thesis showed in Chapter 2 that resource adaptive operation modes significantly help to reduce the energy consumption of enterprise applications in corporate datacenters and thus to reduce the cost of operation. Furthermore, based on this concept, Chapter 3 presented the generalized Adaptation Engine Framework which can be adapted to several service scenarios and significantly increases the economic efficiency in operation. In extensive experiments, the validity and applicability of the framework has been shown. The framework has been extended in chapter 4 with a dynamic and flexible SLA management model which facilitates the risk-aware operation of service. More concretely, the model automatically derives cost minimizing operation strategies balancing the risk of failure and cost of operation. In addition, chapter 5 introduced a decision support system for the derivation of the optimal corporate infrastructure size while chapter 6 demonstrates a model for the design of Service Level Agreements for elastic information system operation.

The development of the Adaptation Engine Framework opens up many questions for future research.

- The Adaptation Engine framework has been tested in a university cluster. Although the cluster allowed extensive experiments and provided detailed insights into elastic information system operation, the ultimate evidence that the Adaptation Engine works in commercial environments is still open. The next step will be the application of the Adaptation Engine in a commercial cloud system and the integration into the cloud manager CloudXplor (Malkowski, edwig, et al. 2010).
- Another option is the extension of the Adaptation Engine to the management of resource competitive information systems. For instance, if a service provider operates several services with a limited amount of resource, he might be able to decrease the risk of SLA violations by continuously monitoring the user demand and evaluating the SLA compliance in order to allocate the resource to the different systems in a profit optimizing and risk minimizing way.
- In this thesis, only stand-alone instances of systems have been considered. However with the progress in the service world, the next generation of service systems will be connected in complex service networks and thus each service may rely on other preceding and subsequent services. The management of one system in this scenario does not only rely on the user and the performance characteristics, but will also depend on external services. This leads to new challenges in the dynamic and risk adjusted management of information systems.
- The models, presented in this thesis, rely on a constant workload composition. Thus, though the models are able to adapt the system according to the workload level, they are not designed to react to changes in the workload composition. For instance, if the users start to execute more application server intensive requests, the system might enter a critical stage as the increased stress on one component of the system has not been considered.
- The performance model in this thesis assumed that each server or node has a dedicated task (e.g. web server). However, depending on the resource demand of a service it might be feasible to collocate different services on one machine. For

instance, if two services are considered whereby one service is CPU intensive and the other one is memory intensive, one might achieve a higher overall system performance by collocating both services on both machines, instead of operating each server on a separate machine. However, this mode of operation requires a very sensitive performance model and is subject to future research.

4. References

Armbrust, M. et al., 2010. A view of cloud computing. *Communications of the ACM*, 53(4), p.50. Available at: http://dl.acm.org/ft_gateway.cfm?id=1721672&type=html [Accessed September 7, 2011].

Bailey, M., 2009. *The Economics of Virtualization: Moving Toward an Application-Based Cost Model*, Available at: http://www.vmware.com/files/pdf/Virtualization-application-based-cost-model-WP-EN.pdf.

Barroso, Luiz André & Hölzle, U., 2007. The Case for Energy-Proportional Computing. *Computer*, 40(12), pp.33-37. Available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4404806 [Accessed July 21, 2011].

Barroso, L.A., Dean, J. & Holzle, U., 2003. Web search for a planet: the google cluster architecture. *IEEE Micro*, 23(2), pp.22-28. Available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1196112 [Accessed August 13, 2011].

Carr, N.G., 2005. The End of Corporate Computing. *The End of Corporate Computing*, (Spring 2005). Available at: http://www.brunomoraes.com.br/governanca-em-ti/wp-content/uploads/2010/05/end_of_corporate_computing.pdf.

Gartner, 2008. Gartner Says 90 Percent of E-Commerce Sites Will Use At Least One SaaS-Based Service by 2013. Available at: http://www.gartner.com/it/page.jsp?id=734612 [Accessed October 8, 2011].

Gartner, 2010. Gartner Says Energy-Related Costs Account for Approximately 12 Percent of Overall Data Center Expenditures. Available at: http://www.gartner.com/it/page.jsp?id=1442113 [Accessed April 30, 2011].

IBM, 2011. IBM - Smarter Planet - United States. Available at: http://www.ibm.com/smarterplanet/us/en/ [Accessed November 3, 2011].

Koomey, J.G., 2007. Estimating total power consumption by servers in the U.S. and the world. *World*. Available at: http://www.greenbiz.com/research/report/2007/09/12/estimating-total-power-consumption-servers-us-and-world.

Langenkamp, J., 2010. Enterprise Server Virtualization Market to Reach $19.3 Billion by 2014. Available at: http://www.information-management.com/news/IDC_predicts_virtualization_growth-10019216-1.html [Accessed April 30, 2011].

Malkowski, S. et al., 2010. CloudXplor: a tool for configuration planning in clouds based on empirical data. *Symposium on Applied Computing*, pp.391-398. Available at: http://portal.acm.org/citation.cfm?id=1774172 [Accessed September 14, 2010].

Open Compute Project, 2011. Open Compute Project - Hacking Conventional Computing Infrastructure. Available at: http://opencompute.org/ [Accessed October 8, 2011].

Risch, M. & Altmann, J., 2008. Cost Analysis of Current Grids and Its Implications for Future Grid Markets. *GECON 08 Proceedings of the 5th international workshop on Grid Economics and Business Models*, pp.13-27.

Schulz, G., 2009. *The Green and Virtual Data Center*, Boston, MA, USA: Auerbach Publications.

Singh, A., Hayward, B. & Anderson, D., 2007. Green IT Takes Center Stage.

Velte, T., Velte, A. & Elsenpeter, R.C., 2009. *Green IT: Reduce Your Information System's Environmental Impact While Adding to the Bottom Line*, New York, NY, USA: McGraw-Hill, Inc.

<u>**CHAPTER II**</u>

TAMING ENERGY COSTS OF LARGE ENTERPRISE SYSTEMS THROUGH ADAPTIVE PROVISIONING

This chapter is a revised version of the paper:

Hedwig, Markus; Malkowski, Simon and Neumann, Dirk, "Taming Energy Costs of Large Enterprise Systems Through Adaptive Provisioning" (2009), International Conference on Information Systems 2009 Proceedings.

Furthermore, the presented version has been complemented with a detailed discussion of the Bottleneck Detection Model taken from:

Malkowski, Simon; Hedwig, Markus; Parekh, Jason; Pu, Calton and Sahai, Akhil, "Bottleneck Detection Using Statistical Intervention Analysis" (2007). Proceedings of the Distributed systems: operations and management 18th IFIP/IEEE international conference on Managing virtualization of networks and services

Abstract

One of the most pressing concerns in modern datacenter management is the rising cost of operation. Therefore, reducing variable expense, such as energy cost, has become a number one priority. However, reducing energy cost in large distributed enterprise system is an open research topic. These systems are commonly subjected to highly volatile workload processes and characterized by complex performance dependencies. This paper explicitly addresses this challenge and presents a novel approach to Taming Energy Costs of Larger Enterprise Systems (Tecless). Our adaptive provisioning methodology combines a low-level technical perspective on distributed systems with a high-level treatment of workload processes. More concretely, Tecless fuses an empirical bottleneck detection model with a statistical workload prediction model. Our methodology forecasts the system load online, which enables on-demand infrastructure adaption while continuously guaranteeing quality of service. In our analysis we show that the prediction of future workload allows adaptive provisioning with a power saving potential of up 25 percent of the total energy cost.

Keywords: Green IT, Bottleneck Detection, Workload Analysis, Adaptive Provisioning

1. Introduction

One of the most pressing concerns in modern datacenter management is the ever-growing cost of operation. Especially, the expenses for electrical energy have become a significant cost factor. Steadily rising prices for electricity, increasing power density of IT systems, as well as the growth of the installed base of computing infrastructure, transformed energy cost reduction into a decisive decision criterion in modern datacenter design. Accompanied by increasing demand of electrical energy, the carbon footprint of the IT industry has been increasing rapidly, which resulted in high public attention. Nowadays, the IT industry emits about two percent of the total CO_2 emissions, which approximately equals the output of the global air traffic (Singh et al. 2007).

In the past, IT was considered a clean industry, but recently, a new type of "green" awareness has grown. The umbrella term "Green IT" denotes all activities and efforts incorporating ecologically friendly technologies and processes into the entire lifecycle of information and communication technology. The sustainable operation of datacenters plays a central role in this domain and focuses on the reduction of energy consumption during operation of datacenters. At a first glance, this objective seems economically motivated. However, considering the direct relation between energy consumption and green house gas emissions, economic and ecologic goals are coherent in this case.

There exist several impressive examples that show the extent of the environmental impact of the IT industry. For instance, datacenters worldwide have consumed the energy output of nineteen mid-sized power plants in 2007. The market-value of the energy was approximately $ 7.2B (Koomey 2007). A more tangible example on the level of single website requests is the fact that a single Google search consumes 8Wh of energy. With 40M search requests per day, the daily energy consumption of Google searches is around 300MWh. This is roughly equivalent to circling around the earth sixty times in a car or 350t of CO_2 based on data of 2005 (Kersten 2007). Given these numbers, it is not surprising that the IT industry and research departments have devoted high efforts to slow down the increasing energy hunger of computing.

However, the heterogeneity and complexity of IT systems dictates that the development of new system designs and new computing paradigms is a challenging task. In the meantime, several concepts to cut down energy consumption already reached market maturity. A prominent example is virtualization technology that is dominant means of consolidation. Nevertheless, today's datacenter layouts and operation concepts still have large potential for improvement. Some efforts deal with the development of new technologies to increase energy efficiency while others aim to remove shortcomings resulting from obsolete and inefficient designs. The complexity of modern datacenters makes the development of holistic improvement concepts non-trivial. Accordingly, the first green initiatives concentrated on easily accessible *"low hanging fruit"* with low complexity, and not on sustainable impact on energy efficiency.

This paper presents a novel model to increase the efficiency of enterprise systems. Tecless continuously supervises the workload process of an enterprise system and systematically analyses the performance behavior of the system. Based on these observations, Tecless dynamically adapts the infrastructure size of the enterprise system to the demand in order to reduce the energy consumption. Due to the inherent domain complexity, Tecless focuses a single common use case in IT operation. More concretely, Tecless is designed for systems providing services to private and commercial end-users. A detailed analysis of workload processes for such systems revealed that these systems often face highly volatile workload processes. Our empirical investigation has shown that such workload processes, especially for systems with a large user community, are nearly stationary. This allows the forecast of the process for the near future.

Because enterprise systems need to be scaled to handle the maximum expected workload level, the systems are only weakly utilized in off peak times. Consequently, this leads to a low average utilization. According to a recent study, the average utilization of datacenters is only around 20 percent (ITP 2007). Especially during night times these systems are typically weakly utilized. Motivated by this observation, our paper introduces and evaluates a model that allows cutting down the energy consumption of IT systems by up to 25 percent using state-of-the-art technology. The energy savings are mainly gained by tailoring the IT infrastructure of the system to the actual demand at all times. Unnecessary servers are identified and removed from the system by switching them off. Modern servers require about 50 percent of their peak power consumption in idle mode (Workstation Performance 2009). This naturally leads to a very high electrical base load. Consequently, only the complete deactivation of these servers allows the conservation of this share.

Hitherto, large IT systems have been developed to maintain large and growing user community with a constant service quality. The performance demand of these systems is often far beyond the capabilities of single machines. Hence, scalability considerations are becoming the main concern. Distributed systems are the state-of-the-art design paradigm, where the application logic is split into several layers being deployed on separate servers. By replicating certain layers of the system, the performance can be scaled. Modern applications, based on state-of-the-art middleware, usually incorporate the capability for dynamic reconfiguration (Alonso et al. 2002). However, the performance characterization of distributed systems remains a highly challenging problem. Determining the maximal possible workload is commonly based the on experience of domain experts (e.g., IT administrators). Non-trivial interactions between the different layers of the systems as well as hidden dependencies make inferences on the performance capabilities an extremely hard problem. Especially, since the hardware configuration is dynamically adapted to the demand, performance analysis becomes incomprehensible and potentially untraceable.

Though the online adaption of the hardware configuration seems feasible, dynamic modifications of the infrastructure are very time consuming. Servers require a certain time to start and the applications might require prior synchronization. Hence, reconfiguration decisions need to be made in advance. Our workload trace analysis revealed that there is a high volatility in workload processes. Elevations of the workload level above 50 percent in less than one hour are common. Apart from the complexity of performance characterization of large systems, this volatility constitutes the second major challenge. An efficient dynamic provisioning model needs to account for the delay between the activation of a server and its availability. Hence, datacenter management must accurately predict the workload process in the near future based on current and post observations. Deviations between the forecast and actual the outcome might lead to performance failures and to the violation of Service Level Agreements (SLA).

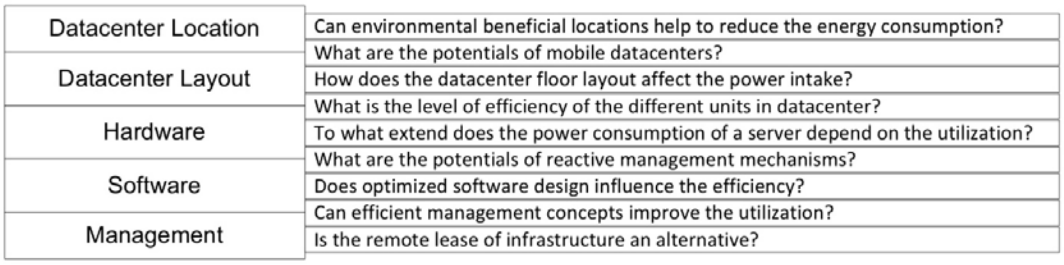
This paper is unique as it advances current norms in provisioning by introducing Tecless, a provisioning model that is designed to accommodate for the complexity of the performance behavior of large systems as well as the volatility of workloads. Guaranteeing the Quality of Service (QoS) is the main goal of the model. Thus, this paper has three main contributions:

- First, the paper designs an iterative model to accurately describe the performance behavior of enterprise systems depending on the workload based on a bottleneck detection model (Malkowski et al. 2007). This advanced performance model bridges the gap between state-of-the-art systems research and green design. The gist of the underlying observation-based method is that it can be applied to any enterprise system as it does not directly rely on software design. This method requires less domain specific knowledge and simplifies the process of provisioning optimal infrastructures.
- Second, the paper introduces a scheme designed to forecast the workload process based on past observations. Compared to related approaches, the proposed scheme identifies and models different factors of influence on the workload process. This results in considerably higher prediction accuracy, which facilitates reliable dynamic reconfiguration decisions in enterprise systems.
- Third, the paper derives a provisioning algorithm, which optimizes hardware configurations according to the performance demand of the system. Our analysis shows that our model has the potential to reduce operational energy consumption up to 25 percent while retaining the same quality-of-service. Given the lower power consumption, the profits of the datacenter operators may substantially increase while the impact on the environment is reduced.

The remainder of this paper is structured as follows. The subsequent section offers a rich overview of green initiatives ordered by their scopes. Several concepts are introduced and the model of this paper is motivated along this scheme. Subsequently, the theoretical details of Tecless are introduced, followed by an evaluation of the model on generic data taken form Wikipedia. The paper concludes with a discussion of the results and an overview on the future work.

2. Related Work

The term “Green IT” comprises all efforts and activities incorporating ecological friendly technologies and process along the whole lifecycle of IT products. In the field green datacenter operation, this especially targets the reduction of the carbon footprint of the IT industry and therefore increasing energy efficiency is major goal. As the expenses for energy rapidly increase, the goals of green operation are generally coherent with economic goals. The manifoldness of this domain cannot be comprehended with a single solution concept, but requires various efforts on the different layers.



Can environmental beneficial locations help to reduce the energy consumption?
What are the potentials of mobile datacenters?
How does the datacenter floor layout affect the power intake?
What is the level of efficiency of the different units in datacenter?
To what extend does the power consumption of a server depend on the utilization?
What are the potentials of reactive management mechanisms?
Does optimized software design influence the efficiency?
Can efficient management concepts improve the utilization?
Is the remote lease of infrastructure an alternative?

Figure 1. Fields of Green IT Operation

Figure 1 shows the different aspects of Green IT Operation and introduces the motivating questions of each area. Most Green IT initiatives target more than one layer. The layer “Location” deals with the physical location of a datacenter and examines the economic and ecologic impact of the facilities environment. The second layer is concerned with the layout of a datacenter and aims to remove inefficiency based on obsolete or inefficient designs. The “Hardware” layer is mainly interested in the optimization of single servers in a datacenter, with special regard to their physical properties. The “Software” layer comprises efforts in the field of code optimization and scheduling for resources conserving operation. The last layer deals with economic management aspects. Table 1 offers a broad overview on different Green IT activities and related initiatives.

Table 1. Scope of Green IT Initiatives

Object	Scope	Location	Layout	Hardware	Software	Management
Physical Location	Utilizing hydroelectric power or building datacenters in moderate climate zones reduces the cost for energy. For example, several US IT companies have built their datacenters next to the Columbia River,	X				

	Oregon, USA area to profit from hydroelectric energy (Kurp 2008).					
Mobile Datacenters	Mobile datacenters constitute a new design alternative, allowing to provide computing resources locally to temporary demand spots (e.g. Olympia). The project “Blackbox” of Sun Microsystems integrated a complete datacenter into a standard shipping container (Cooley 2009). Goggle applied for a patent for a “Water-Based Data Center” including concepts for sea-based electricity generators (Clidaras & Stiver 2010).	X	X	X		
Physical Configuration	Servers generally require air-conditioned environments. However, conventional layouts do not regard the airflow within the datacenters and disregard the resulting mixture of chilled and hot air. A first step to increase efficiency is setting hot and cold aisle. In state-of-the-art datacenters planning the airflow is simulated and efficiently guided through the infrastructure (R K Sharma et al. 2005; C. B. Bash et al. 2006; Ratnesh K Sharma et al. 2008).		X	X		
Datacenter Infrastructure	More than 50% of the energy consumed in datacenters is needed for the infrastructure, e.g. cooling, power supplies or light. Especially uninterruptible power supplies (UPS) are often dimensioned to provide the complete datacenter. Reducing UPS to the critical systems, reduces the power consumption significantly(T. Velte et al. 2009).		X	X		
Energy Efficient Hardware	Server design is often solely focused on performance whereas energy efficiency is often secondary. However, for example power supplies are relatively cheap and increase the efficiency significant. Volunteer certificates have been introduced, e.g. EnergyStar (EPA 2008), to encourage minimum efficiency standards for hardware. A broad overview on energy efficient hardware is given in (T. Velte et al. 2009).		X	X		
Chip Level	The processor accounts for about 30% of the energy consumption of a server. Though the main design goal of chip development is performance, energy efficiency gets more and more attention. This includes new chip layouts (Brunschwiler & Michel 2008; Puttaswamy & Loh 2006), or new chip-level cooling concepts(Linderman et al. 2008).			X		
Server Design	Innovative server design can significantly help to improve the efficiency. For instance blade systems are one common concept. Instead of using multiple stand alone servers, several systems are installed in one rack, sharing common hardware such as power supplies, network gear or fans. This helps to reduce the energy demand and allows efficient cooling concepts(T. Velte et al. 2009).			X		
Storage Systems	Enhancements of storage systems are an important part of green-IT. (Won et al. 2008) developed an energy-aware scheduling algorithm for bundling disk accesses and setting the disk into idle mode in the mean times. Virtualization and Consolidation together with the help of Network Attached Storage Devices, enables the automated storage of data on appropriate media (hard disks, tape) according to reliability, availability and access frequency concerns (Schulz 2009).			X	X	
Thermal Aware Scheduling	The heat signature of servers depends on their utilization. To achieve a uniform thermal distribution within the datacenter, the workload is distributed according to the physical layout of center (Moore et al. 2005; Ghanbari et al. 2007; Parthasarathy Ranganathan et al. 2006).			X	X	
Virtualization and Consolidation	Oftentimes enterprise systems are required to run on dedicated servers for security or compatibility reasons. However these systems are often only weakly utilized. With the help of consolidation, the system can be migrated to a single machine, while virtualization guarantees the independence of the systems (Schulz 2009).			X	X	X
Cloud Computing	Cloud Computing is a new computing paradigm, providing virtualized resources as a service over the internet. Integration of Cloud Computing into an enterprise system helps to improve the utilization as peak demands are provided with remote resources. However the derivation of the optimal size is a non-trivial (Hedwig et al. 2010).				X	X
Fast adaption of Business Process	Changes in business processes are often accompanied with new software systems. However, legacy systems often remain online for years and are hardly used. Fast adoptions and regularly reviews of the IT infrastructure can lead to significant cost reductions (T. Velte et al. 2009) (Velte et al., 2008).					X

2.1. Adaptive Provisioning Models

Adaptive Provisioning Models constitute a new Green IT concept and belong to the categories hardware, software and management. The model proposed in this paper is specifically designed for the requirements of distributed enterprise and information systems with a large user community, whereby the varying usage intensity of the clients over time is explicitly modeled. Virtualization and consolidation has a similar scope. Various standard software products already reached market maturity, e.g. Xen or VMWare. However, this concept specifically targets small- and midsized systems (T. Velte et al. 2009), which do not have the non-trivial complexity of distributed and replicated architectures. A closer examination of the problem setting reveals it's bilaterally. Adaptive provisioning does not only demand a profound understanding of the performance behavior of a large-scaled system, but eventually requires triggering reconfigurations of the hardware infrastructure in advance. This necessity demands to model the workload process in order to forecast the near future of the system load and hence to guarantee a continuous QoS. The approach of designing complex models to improve energy efficiency has also been examined in detail (Viswanathan & Monie 2006; Dhiman & Rosing 2006). Though the scope is similar to these concepts, Tecless specifically models the workload process as well as it utilizes a complex system performance model.

The field of performance characterization of large-scale, distributed systems is already alone a hot topic in research and practice; several concepts have been proposed to solve this problem. Most advanced system performance models abstract from the technical properties of the infrastructure as well as the software design characteristics and utilize empirical data coupled with statistical methods, such as machine learning, or operations research concepts. The main objectives of these approaches are (i) to evaluate the performance characteristics of a system and (ii) in case of performance failures to identify the root cause. One renown approach makes use of Tree-Augmented Bayesian Networks (TANs) to identify combinations of system level metrics with their corresponding threshold values and system performance measures (Ira Cohen et al. 2004; Zhang et al. 2005). Given an arbitrary system state and a corresponding workload, this methodology has the capability to calculate the probability of performance failures and to identify the root causes of the performance limitation. Another widely used concepts employ queuing-models (Urgaonkar, Pacifici, et al. 2005; Urgaonkar, Shenoy, et al. 2005; Urgaonkar et al. 2007). The intuition is to reproduce the structure of an n-tier application as a queuing network and to model the processing time at each layer of the system as transition probabilities. Based on the transition time of jobs through the network, the performance can be assessed. The lengths of the queues represent potential bottlenecks in the system.

Some of these models incorporate first extension for adaptive provisioning. The TAN based model has been extended with a mechanism for short-term performance forecasting (Powers et al. 2005). With the help of time series analysis and regression, the occurrences of performance violations in the system are modeled. By extrapolating the derived process, the future system stress is forecasted. This prediction, together with the TAN, provides the base for reconfiguration decisions. Compared to the Workload Prediction Model of Tecless, the prediction algorithm of the workload is rather reactive than predictive and completely disregards observations of the distant past. Furthermore the prediction regards the workload process en bloc without isolating different factors of influence. Though the introduced adaptive provisioning model is based on similar techniques, the different nature of the factors of influence is analyzed and estimated individually, leading to a significantly higher forecasting accuracy.

The discussed queuing model has also been advanced to an adaptive provision model. (Urgaonkar et al. 2008) propose a predictive algorithm to estimate the request arrival rate in the system. Based on the workload observations of the same time of day for a distinct period, they create a request arrival rate distribution and set a high percentile as the forecast of the near future usage intensity. This value is increased by the weighted average prediction error of the last hours. The prediction error of the recent past is included into the forecast. Based on the final prediction, they use their queuing network to estimate the system stress and reconfigure the infrastructure accordingly. Though the incorporate feedback mechanism accounts for the dynamic of the workload process, the predictive element is highly static and hence cannot react to the dynamic of a workload process. The derivation of a workload distribution tends to continuously overestimate the workload level, as this concept is highly vulnerable to outliers. The Workload Prediction Model of Tecless, does not only predict the level of the workload, but also models the dynamic of the process. This allows significantly higher prediction accuracy. Furthermore the use of a queuing network requires prior modeling of the software architecture. The Bottleneck Detection Model used in Tecless derives the performance directly from observations and hence does not require this step. Overall, Tecless is independent of the software design and the user behavior. Instead, all necessary configuration parameters of the model are directly derived from observations of the system during operation.

3. The Tecless Model

This paper introduces Tecless, a novel model for adaptive provisioning of distributed enterprise systems. As illustrated in Figure 2, the model comprises three major units, each dealing with the special characteristics of such systems.

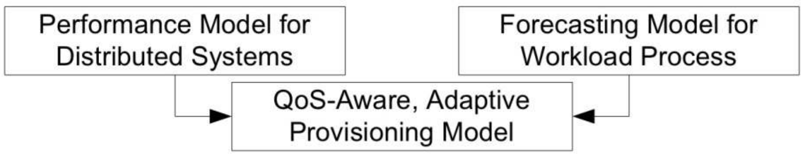

Performance Model for Distributed Systems
Forecasting Model for Workload Process
QoS-Aware, Adaptive Provisioning Model

Figure 2. Structure of the Adaptive Provisioning Model

- A bottleneck detection model (Malkowski et al. 2007)that specifies how a given enterprise system needs to be enhanced (e.g. by adding another application or database server) at runtime to increase the system performance. The Bottleneck Detection Model provides an accurate description of the performance characteristics of an enterprise system. By iteratively analyzing the performance of a system under different stress levels, the Bottleneck Detection Model is able to determine the maximal feasible workload of a hardware configuration as well as the resource, limiting the performance.
- A workload prediction model that attempts to forecast volatile user behavior: By analyzing the behavior of the past, the model predicts the level of workload in the near future. The single factors of influence are isolated and estimated separately with the help of a modified time series decomposition model. To account for unforeseen changes in the workload process, the model is complemented with a feedback

mechanism. This novel methodology offers a precise forecast of the workload process with an accuracy of up to 99%.

- A provisioning model that brings the Workload Prediction Model and Bottleneck Detection Model together in order to continuously optimize the system configuration such that a given Quality-of-Service (QoS) level can be achieved. Based on the other two models, the provision model derives an optimal infrastructure size at any time, allowing an economical infrastructure operation.

3.1. The Bottleneck Detection Model

The following paragraphs detail the various aspects of the models. The idea of performance models for large-scale, distributed systems originates from the field of systems research in computer science. The Bottleneck Detection Model is motivated by the increasing complexity of modern architectures and the need to reconfigure hardware and software at run-time. Today, IT administrators base their decisions on monitoring data, expertise and experience. This intuitive approach is often error-prone, which has lead to the demand of robust models for characterizing performance behavior during operation. Particular importance has been given to the determination of the maximum performance capabilities of a system deployed on a given infrastructure as well as on the bottleneck, limiting the performance of the system.

Before production, an effective staging phase assures system administrators that a hardware/software configuration is capable of handling workloads to be seen during production. Starting at an initial configuration, this phase augments resources allowing the configuration to better satisfy the SLOs. So far our bottleneck detection approaches consisted of a multi-step analysis. If a SLA was not met (SLO-satisfaction drops significantly) in a certain scenario, a three-step detection process began: staging the system with varying workloads and collecting performance data from system-level and application-specific metrics, training a machine learning classifier with the data, and finally querying the trained machine learning classifier to identify potential bottlenecks. Please refer to (G. Jung et al. 2006) for more details. While our three-step methodology proved to be successful, it mainly relies on machine learning algorithms to execute the final performance modeling and classification. This implies two typical shortcomings that lie in the nature of the modeling scheme. Firstly, the machine learning classifiers require a training phase. This can be cost-intensive since certain accuracy and robustness levels might be defined *a priori*. Secondly machine learning classifiers produce a model that is not necessarily interpretable in a trivial manner. We discussed suitable interpretations in (G. Jung et al. 2006). Nevertheless, this led to a residual degree of uncertainty in the interpretation of the analysis results.

In this section, we propose a novel approach based on statistical techniques, which results in an improvement of our bottleneck detection process in a consistent manner. We introduce an intuitive statistical model, which eliminated the need of machine learning on the one hand. And on the other, we observe that our approach achieves the same high accuracy level at a lower cost (fewer staging trials). Therefore we greatly increase the efficiency of the detection process and enhance the clarity of the final results at the same time.

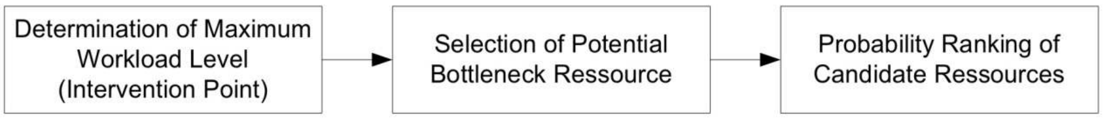

Determination of Maximum Workload Level (Intervention Point)
Selection of Potential Bottleneck Ressource
Probability Ranking of Candidate Ressources

Figure 3. Structure of the Bottleneck Detection Model

The objective of the model is to identify the bottleneck resource of the system. This is achieved by determining the maximum level of workload a system configuration is able to handle as well as the corresponding reasons for the performance limitations occurring at this maximum level. The Bottleneck Detection Model correlates the performance of the system with system level metrics of the different servers of each layer. A three-stage model (Figure 3) analyzes the behavior of the different resource and identifies the resource responsible for the performance limitations. The following presentation has previously been published (Malkowski et al. 2007) and replaces the original short version of the published version of this article.

3.1.1. Bottleneck Detection Model Assumptions

The following assumptions form the basis of our automated bottleneck methodology. They emphasize the general issues that need to be addressed by any satisfactory detection method and are reflected in previous Elba efforts.

- A single experiment trial is not sufficient to record a conclusive metric vector and thus several trials of varying workloads are required.

- Non-obvious interactions between resources make observation based bottleneck detection a hard problem. Nontrivial correlations have to be examined and the detection method needs to be able to produce a probabilistic result ranking.
- The number of recorded monitoring metrics is very high. It is critical to device an approach that is able to sort through copious metric data automatically.
- The nature and appearance of metrics can vary significantly and they are typically categorized as either system-level or application-specific.
- High utilization of a resource implies high demand from an application while it may not necessarily be indicative of a bottleneck. A detection mechanism has to be capable of distinguishing bottlenecking behavior in terms of resource saturation.
- Especially trend changes in metric graphs are of high importance. In fact we found in our previous work that it was highly effective to examine first derivative approximations instead of the actually recorded values.

3.1.2. The Detection Model

The first step in the determination of the maximum feasible workload level is to analyze the performance behavior of the system over a wide range of workload levels $w \in WS$ (workload set). In the Bottleneck Detection Model, the term workload is defined as the number of concurrent users in the system. For every workload level w the behavior of the different resources in the servers is monitored by recording the corresponding system level metrics, such as CPU utilization, memory transfer rate or hard disk response time. With the help of the Bottleneck Detection model, the maximum feasible workload level is determined, which still satisfies the required QoS, whereby the QoS itself is defined as an upper limit for the end-to-end user response time.

The previous assumptions together with observations from empirical data analysis suggest a simple performance model. We formulate the latter in terms of statistical intervention analysis, which allows us to formalize the characteristic bottleneck behavior of the system accurately.

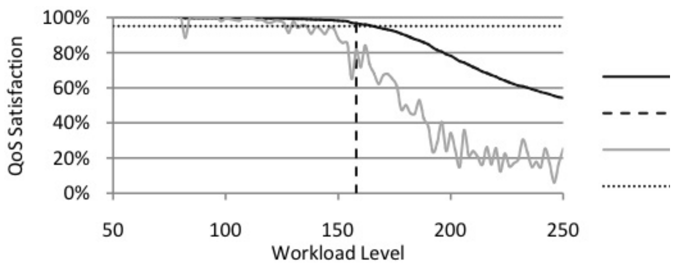


Figure 4. Illustration of the Heuristic

Figure 4 illustrates the procedure of the heuristic for the determination of the maximum workload level. First we need to define an exogenous crossover point (c ∈ WS). This specific number of concurrent user sessions can be seen as an intervention point that divides our workload span (WS) into two disjunctive intervals:

$$I \coloneqq \{w \in WS \colon w < c\} \quad (1)$$

$$I \coloneqq \{w \in WS \colon w \geq c\} \quad (2)$$

In this notation *I* represents the set of workloads that result in high levels of SLO satisfaction of the system, whereas the satisfaction levels drop significantly when exposed to workloads in I' (intervention effect).

For our purposes we also need to adapt the standard transfer functional model formulation (Brockwell & Davis 2002). For any workload $w \in WS$ an impact assessment model for the first difference of any metric value Y_w can be formulated in terms of Equation 3. Note that we use the first difference as approximation of the first derivative consistently with our findings in (G. Jung et al. 2006).

$$\nabla Y_w \coloneqq f(I_w) + N_w + \mu \quad (3)$$

$$I_w \coloneqq \begin{cases} 1 & \text{for } w \geq c \\ 0 & \text{else} \end{cases} \quad (4)$$

In this formulation N_w is the noise component, and μ denotes the constant term. The effect of the intervention variable I_w on the metric trend is defined as $f(I_w)$. Following the standard notation, I_w is defined as an indicator function (Equation 4). We can now subtract the noise component from both sides of Equation 1. Since we only have to deal with abrupt and permanent intervention effects we can assume linearity in the metric values. Based on this linearity assumption we introduce δ as the constant term of the intervention effect, which yields the following formulation:

$$\nabla Y_w - N_w = \delta I_w + \mu \quad (5)$$

In order to characterize the final model in a convenient manner, we define μ' in Equation 6 which leads to the final model formulation in Equation 7.

$$\mu' \coloneqq \mu + \delta \qquad (6)$$

$$\nabla \tilde{Y}_w \coloneqq \nabla Y_w - N_w = \begin{cases} \mu & \text{for } w < c \\ \mu' & \text{for } w \geq c \end{cases} \qquad (7)$$

This notation emphasizes the importance of the potential change in the trend of the metric value Y_w as the system progresses from I to I' with increasing workload. Moreover, it allows us to establish causality between the model parameters of the low level metric and the high level system performance in an intuitive manner.

3.1.3. Determining an Intervention Point

Since the crossover point (c) between I and I' needs to be defined *a priori*, we define an iterative algorithm for our automated analysis scheme. The main idea is to assess the workload when the SLO-satisfaction (SAT_w) loses its stability and starts to deteriorate significantly upon further workload increase (i.e. we assume Property 8 and 9). Although the model formulation requires an exact transition point, it is sufficient for our method to approximate c in a qualitative manner (refer to Table 4).

$$\forall_{i \in I}: SAT_i \approx \text{const} \qquad (8)$$

$$\forall_{i \in I'}: SAT_i \ll \frac{1}{|I|} \sum_{j \in I} SAT_j \qquad (9)$$

We start at the lowest workload in our dataset and iteratively increase the value by the smallest possible step. In every iteration, we calculate a simple heuristic approximation of the ninety-five percent confidence interval of the SLO satisfaction values seen so far. We consider n_0 values which resulted from a workload smaller or equal to $w_0 \in WS$ (the workload currently examined).

$$90\% \leq \frac{1}{n_0} \sum_{0 \leq i \leq w_0} SAT_i - \frac{1.96}{\sqrt{n_0 - 1}} \sqrt{\sum_{0 \leq i \leq w_0} \left(SAT_i - \frac{1}{n_0} \sum_{0 \leq j \leq w_0} SAT_j) \right)^2} \qquad (10)$$

We continue to the next iteration as long as the lower bound of the confidence interval is not below ninety percent (10). Thus we characterize the satisfaction level for the first interval in a binary fashion as suggested by our observations. Once the lower bound of the confidence interval drops below ninety percent we exit the algorithm. The exit point $w^* \in WS$ is a heuristic approximation of the crossover point c. We can assume that the SLO satisfaction has deteriorated significantly from its stable level for all workloads greater or equal to w^*, which yields the following formulation:

$$\hat{I} \coloneqq \{w \in WS: w < w^*\} \quad (11)$$

$$\hat{I}' \coloneqq \{w \in WS: w \geq w^*\} \quad (12)$$

3.1.4. Metrics Selection Scheme

We can now turn to the process of selecting a set of potential bottleneck metrics and discarding all metrics that do not indicate a high resource saturation level. Given a known intervention (SLO begins to deteriorate) we identify all metrics that show evidence of a corresponding plateau (i.e. significant and permanent shift in average value) and a variability change in their first derivative (further evidence for a saturated resource). To identify the candidate set we perform a basic hypothesis-testing scheme adapted from [10]. We define a rule-based analysis process for testing the null hypothesis (13) of constant mean μ and variance σ between the two intervals.

$$H_0 \coloneqq \hat{\mu} \approx \hat{\mu}' \wedge \sigma \approx \sigma' \quad (13)$$

Empirical testing revealed that we have to account for the high variability of the metric data as well as adjust the analysis to specifically detect abrupt plateau shifts. Thus we deviate from the traditional intervention analysis methodology and devise a different testing scheme. We calculate representative quantiles for each interval and metric. The latter characterize the filtered behavior of the data in a more stable manner. We proceed to apply two selection rules in order to limit the group of candidate bottleneck metrics.

$$q_{0.5} > q'_{0.5} \wedge |\, q_{0.2} - q_{0.8} \,| > |q'_{0.1} - q'_{0.9}\,| \quad (14)$$

Rule 14 accounts for all limited metrics that will saturate at a level of hundred percent. We choose all metrics where the median has decreased as well as where the distance between ten- and ninety-quantile in the second interval is smaller than the distance between twenty-

and eighty-quantile in the first interval. If this rule is satisfied we have significant evidence to reject the null hypothesis and assign the metric to a set of potential bottlenecks.

$$q_{0.9} < q'_{0.1} \wedge q_{0.9} < q'_{0.5} \wedge q_{0.9} < q'_{0.9} \quad (15)$$

Rule 15 accounts for all metrics that are not limited and show an exponential behavior when the resource saturates. We select all metrics, where all three quantiles of the second interval have increased above the ninety quantile of the first one. Again we reject the H_0 if the rule applies. In this manner we have eliminated all metrics that do not show strong indications of bottlenecking behavior near the intervention point and narrowed our attention to potentially interesting resources. Note that the complete empirical derivation of the two decision rules is omitted due to space restrictions. Nevertheless it is based on standard statistical methods and our analysis experience.

3.1.5. Impact Assessment

Once we have identified the set of candidate bottlenecks we can perform a ranking to describe the magnitude of the change. The magnitude reveals the correlation with the intervention and specifically accounts for the exact time when the change in the metric occurred. Hence we design a normalizing ranking function R by calculating the quotient of the absolute mean values of the two intervals:

$$R := \left|\frac{\hat{\mu}}{\hat{\mu}'}\right| \quad (16)$$

This mechanism has two implications. Firstly, we assess how well the crossover point was chosen for each particular metric (temporal ranking). If the split is not exact, the resulting quotient will have a value closer to one. Furthermore, we have to rank how large the relative shift in plateau levels is for each particular metric. We expect bottlenecked metrics that were chosen with Rule 14 (limited metric, such as CPU utilization) to display a very high-ranking value potentially approaching infinity. The slope of the metric values drops from a linear increase to a stable value near zero. Metrics chosen by Rule 15 (unlimited metrics, such as IO wait time) will show a very low ranking value that is close to zero on the other hand. This means that a moderate positive slope changes to a very strong (exponential) growth. According if Rule 14 or Rule 15 applies, we will assign the different metrics to two candidate sets as the ranking value has to be interpreted differently.

3.1.6. Automated Scaling Phase

The resource with the maximal score value according to rule 14 has the highest likelihood to be the bottleneck. Please note that the correct bottleneck resource might be in the set of Rule 15. However, for the purpose of the adaptive provisioning, we are only interested in the layer causing the performance problem but not in the actual resource. Our evaluations have shown that only using the first set to determine the bottleneck layer is sufficiently accurate.

In summary, the Bottleneck Detection Model identifies the maximum feasible workload level and determines the system layer with the bottleneck resource. Based on these results, the system configuration can be adapted to achieve higher performance goals by adding an additional server to this layer. More concretely, applied iteratively, this methodology can be used to determine optimal configurations for certain levels of workload.

The first step is to analyze a minimal system configuration. After the determination of its maximal workload level and the corresponding bottleneck, the bottleneck resource is replicated to achieve higher performances. By repeating this procedure iteratively, a set of workload levels and corresponding configurations can be derived. This set is later used to select on optimal configuration according to the state of the system.

3.2. The Workload Prediction Model

In the previous subsection, we presented the Bottleneck Detection Model, which derives the performance characteristics of large enterprise systems. This methodology delivers a set of configurations and their corresponding maximum workload levels, which provide the foundation for dynamically adapting the infrastructure size to the demand. However, as previously mentioned, changes in the system configuration are time-intensive, e.g. the database needs to be mirrored on additional database servers. As workload processes exhibit high volatility, reconfiguration decisions cannot be solely based on the current state of the system. Instead, the evolution of the workload process needs to be anticipated, such that QoS is continuously satisfied while at the same time the energy efficiency is increased.

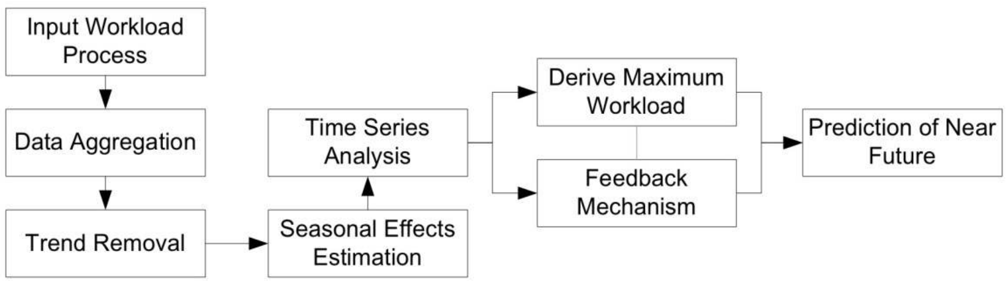


Figure 5. Structure of the Workload Prediction Model

This subsection presents a novel time series approach for analyzing and forecasting workload processes of end-user enterprise systems. Our analysis of this processes showed strong correlations of the workload level with the time of day and day of week. Furthermore, we identified long-term factors of influence which imply that the workload processes is not fully stationary. The presented Workload Prediction Model is able to forecast non-stationary workload processes, as long as the historic data allow deriving the long term dynamics of the process. Sudden changes in the process behavior are handled with a feedback mechanism. Compared to other approaches in this domain (Powers et al., 2005, Urgaonkar et al., 2008), this model can actually identify and model all significant factors of influence. Our resulting model allows predicting the near future. Together with a feedback mechanism, this novel concept is able to predict workload process with an accuracy of up to 99% in the mentioned domain. Figure 5 illustrates the structure of the Workload Prediction Model. The first step is the aggregation of the data to remove the short-term variation. Afterwards a modified time series analysis decomposition model is used to estimate the different factors of influence on the process. This includes the trend removal, the estimation of seasonal effects and the analysis of the residual process. As the decomposition model predicts the average expected workload, the final step is to calculate the maximum expected workload in the near future. This upper bound is necessary to account for the short-term stochastic variation of the process. To provide the model with capability to react to unforeseen changes, the model is complemented with a feedback mechanism.

3.2.1. Properties of the Workload Processes

In the scenario of end-user systems, the workload is generated by various clients with individual usage patterns. Users can start at new season at any time, execute an arbitrary transition path, and terminate their session. Modeling all clients individually is infeasible and hence the prediction model is designed to work on aggregated data. All clients are aggregated into a single source. This levels out individual effects and reduces the complexity to a manageable level. In our Workload Prediction Model, we assume that usages patterns (i.e. composition of workload) of the aggregated user communities are constant over time. This allows neglecting transaction type information.

The database of the Workload Prediction Model provides a training dataset based on the past observations. The database is used to estimate the various parameters of the model. The Workload Prediction Model requires a time series of the total number of request per period in the training dataset. This leads to two configurable parameters: The length of single period Δ and the total training dataset length $n\Delta$. The advantage of a longer timeframe $n\Delta$ is the reduction of the standard error of the later estimators. However, we have to take into account that usage patterns can vary in the long term (influence of non-stationary factors). This might bias the parameter estimation. Consequently choosing the dataset length is a tradeoff between accuracy of the estimators and the capability of the model to adapt changes in the process behavior. The second parameter Δ (single period length) should be long enough to eliminate the short-term stochastic variation. The optimal period lengths of both parameters need to be empirically evaluated, as their optima strongly depend on the characteristics of the individual processes. The aggregated input processes consists of $i=1,\dots,n$ data points r_i, each representing the total number of requests during a single period. Each data point r_i starts at the time t_i and ends at $t_{i+1} = t_i + \Delta$.

$$r_i = m_i + s_i + Z_i \quad (17)$$

After the definition of the model input, we are now able to define the formal model. The decomposition model (17) of the workload process r_i consists of a long term trend component m_i, a seasonal element s_i and the residual stochastic process Z_i. By definition of the decomposition model, the trend and seasonal component are considered as deterministic elements. The following subsection presents appropriate methods to estimate the single factors of influence directly from the workload process. The feedback mechanism and the determination of the maximum expected workload are subsequently discussed.

3.2.2. Removal of the Long-Term Trend

Classical decomposition models suggest a top-down approach to analyze time series data (Brockwell and Davis 2002). The first step is the removal of the trend of the process, which originates from changes in the user behavior (e.g. increasing usage intensity) or the growth of the user community. Classical decomposition modeling suggests the use of regression. However, the use of univariate regression is ineligible for the trend isolation in our scenario, as workload processes often exhibit more complex dynamics. An alternative methodology is the smoothing with a long-term moving average as suggested by (Kreiß & Neuhaus 2006). The idea is to remove the long term trend from the observed process without knowing the true factors of influence. It assumes that the effect of the long term factors is already significantly represented in the preceding data points. To avoid interference with seasonal effects, the length of the moving average window needs to be longer than the seasonal periods. Therefore it is set to one week, whereby the number of data points per week is given by n_w.

$$m_i = -\frac{1}{n_w}\sum_{j\in L_i} r_i \quad L_i \coloneqq \{j \in N | t_j < t_i \wedge t_j \geq t_{i-n_w}\} \quad (18)$$

Equation (18) calculates the average workload level of the preceding week for every data point r_i. This value yields the first component of the decomposition model. Furthermore it can be shown that this methodology satisfies the first property of weak stationarity (Kreiß & Neuhaus 2006).

3.2.3. Analysis of Seasonal Effects

The isolation of the seasonal influence in the process is the next step in the decomposition model. It deals with the removal of all factors with periodical recurring influence on process. In the context of our model two effects can be identified: the weekday and the time of day. Regarding that the length of the calibration dataset comprises at most the data from a few months, yearly effects cannot be estimated. Principally, the effect of the time of day and day of week can be estimated separately. However, evaluations have shown that - given a sufficient long training dataset - the simultaneous estimation of both effects deliver statistically better results.

$$s_i = \frac{1}{|W_i|}\sum_{j \in W_i}(r_j - m_j) \qquad W_i := \{j = 1, \dots, n \mid w(j) = w(i)\} \quad (19)$$

Equation (19) expresses our simultaneous estimator, where the help-function *w(i)* returns a unique index representing the time of day and weekday. For each index the average number of requests is calculated. This factor is estimated on the base of the trend free data and renders the second component in the decomposition model.

3.2.4. Analysis of the Residual Process with Time Series Analysis

The last element of the decomposition model deals with the residual stochastic variation of the process. The residuals are given as the difference between the observations and the trend and seasonal component *(Z_i=r_i-m_i-s_i)*. Time Series Analysis provides various tools for the process type identification (Hamilton, 1994). Nevertheless, the selection of an appropriate model requires in-depth prior analysis of the data. Furthermore, the application of time series analysis at least requires that the process is weakly stationary, which demands the following two properties. The first property requires a constant mean *$E(Z_i)=\mu$*, which is automatically met having applied the moving average smoother (Kreiß & Neuhaus 2006). The second property demands that the correlation between any two data points is constant for any given distance. The compliance to this property is hard to show. However, a thorough analysis of the results can be used as ex-post justification of this property.

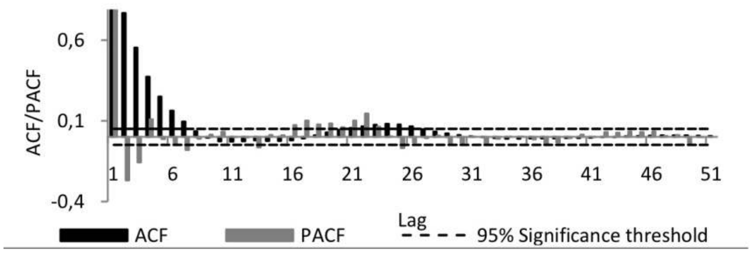


Figure 6. ACF and PACF of Residual Wikipedia Workload Process

Figure 6 shows the autocorrelation function (ACF) and partial autocorrelation function (PACF) of a sample workload trace. The exponential decrease of the ACF and the first four significant lags of the PACF indicate an autoregressive (AR) behavior of order 4. The order of the process strongly depends on *Δ* and on the characteristics of the process and hence has to be determined for every examined workload trace. Generally the fitting of a model is an optimization problem. However, in case of an AR process the Yule-Walker estimate constitutes a feasible alternative, as it allows the arithmetical computation of the model parameters. Albeit the standard error is very high, it is sufficient in this scenario, as the parameters are estimated from several hundred data points. The standard error of the Yule-Walker estimators of large datasets has approximately the same standard error as the maximum likelihood estimators.

$$\hat{\phi} = \hat{\gamma}(p)\hat{\Gamma}_p^{-1} \text{ with } \hat{\Gamma}_p = [\hat{\gamma}(k-j)]_{k,j=1}^{p} \text{ and } \gamma_Z(h) = Cov(Z_i, Z_{i+h}) \quad (20)$$

The formulation of Yule-Walker parameter estimation is given in (20). This leads to final model, given in (21).

$$\hat{Z}_i = E(Z_i) = \hat{\phi}_1 Z_{i-1} + \cdots + \hat{\phi}_p Z_{i-p} \quad (21)$$

$$\hat{r}_i = \hat{Z}_i + \hat{m}_i + \hat{s}_i \quad (22)$$

This formulation allows predicting the next value in the residual series Z_i based on the most recent observations of the process. By adding the trend and the seasonal component (22) the average expected workload in the near future is forecasted. Depending on the size of the interval Δ, the most recent values are not included in the prediction. However, these values are considered through the later discussed feedback mechanism.

3.2.5. Determining the Maximum Number of Concurrent Users

Up to this point our scheme predicts the average number of requests per single period. However, the Bottleneck Detection Model bases its analysis on the number of concurrent users in the system. By relying on the QoS requirement, being end-to-end response time < 1s (c.f. the last section) and utilizing the fact that the end-to-end response time of the system tends towards the maximal accepted response time in the case of performance violations, the number of concurrent jobs in the system can straightforwardly be derived. This, however, entails an overestimation of the workload during the uncritical phases of the system.

$$\hat{r}_i^c = \frac{\delta}{\Delta}\hat{r}_i \quad (23)$$

Having in mind that the model primarily depends on the accurate prediction of the number of concurrent users during the critical states, this issue does not alter the results. Accordingly, we have the approximated number of concurrent jobs in the system is given by (23).

3.2.6. Determining the Maximum Expected Workload

Hitherto, the Workload Prediction Model estimates the average expected number of concurrent jobs in the system. Due to the stochastic nature of the workload process, the level of workload will exhibit very high short-term variations. To ensure a continuous QoS satisfaction, the system configuration needs to handle the maximum expected workload in each single period. With the help of statistical inferences, the maximum expected workload level, which is not exceeded with a certain probability, is determined. Assuming independence of single requests allows us to model the workload process as a queuing model. Independent arrival processes are usually modeled with an exponential mean time distribution (Kleinrock 1975). By definition, in this scenario, the total number of requests during a period is Poisson distributed with a variance of the square root of the expected value. In case the number of events during the period is large enough (>100), then the Poisson distribution tends toward normal distribution.

$$\hat{u}_i^c = \hat{r}_i^c + 2{,}33 * \sqrt{\hat{r}_i^c} \quad (24)$$

This nice property enables us to formulate the maximum expected workload level. To determine the workload, which is not exceeded with a probability of 99%, the normal distribution confidence interval is calculated (24).

3.2.7. Feedback Mechanism

Recall that the idea of the Workload Prediction Model is to analyze past user behavior in order to predict near future workload. However, in case of unforeseen changes in the usage patterns, e.g. flash-crowds (Urdaneta et al. 2007), this model ultimately fails. To overcome this limitation, the model is complemented with a basic feedback mechanism, which continuously compares the model prediction with the observed data. This allows the model to reactively deal with short-term non-stationary process factors.

$$f_i^c = \begin{cases} (r_{i-1}^c - \hat{r}_{i-1}^c) & if\ (r_{i-1}^c - \hat{r}_{i-1}^c) > 0 \\ 0 & else \end{cases} \quad (25)$$

If the former prediction is lower than the corresponding observation, then the difference of both is added to the next workload prediction. The feedback mechanism (25) is the last element of the Workload Prediction Model. The final model enables forecasting the expected number of concurrent users in the system.

$$\tilde{r}_i^c = \hat{u}_i^c + f_i^c \tag{26}$$

The maximum expected workload (26) will be used in the Adaptive Provisioning Model in the next section. The value can be interpreted as an upper bound of the number of concurrent jobs in the system (which is not exceeded by at least a 99% probability). At the bottom line, the hardware configuration needs at least to handle this number of concurrent user.

3.3. *Putting Things Together in the Adaptive Provisioning Model*

The Bottleneck Detection Model provides a set of configurations and corresponding maximum workload levels. The Workload Prediction Model forecasts the expected maximum number of concurrent jobs in the system for the next period. The final step is to combine both models into the Adaptive Provisioning Model, which aims at selecting the minimal system configuration that satisfy the QoS at any time

Let $c_i \in C$ be a valid hardware configuration for the examined system, whereby c_i is defined as the smallest feasible configuration. The variable c_i^{max} defines the maximal feasible workload level of the configuration c_i which satisfies the QoS requirements. Furthermore, the configurations are sorted according to $j<i \Rightarrow c_j^{max} < c_i^{max}$.

$$c^* = \arg\min_{c_i \in C} \{c_i^{max} | c_i^{max} > \tilde{r}_i^c\} \tag{27}$$

$$\arg\min_{c_i \in C} \{c_i^{max} | c_i^{max} > \tilde{r}_i^c\} \tag{28}$$

Equation (27) constitutes the core of the Adaptive Provisioning Model. Depending on the predicted maximum workload level, the smallest sufficient system configuration is selected. During operation, the Adaptive Provision Model continuously supervises the workload process. If the present configuration is insufficient to handle the expected system stress, the model switches to the next stronger configuration c^*_{i+1}. If a weaker configuration c^*_{i-1} is also capable of handling the expected maximum workload, the capacity of the system is reduced to the next smaller configuration.

4. Case Study on Wikipedia Workload Traces

Following the presentation of the theoretical background, this section evaluates the performance of the Adaptive Provisioning Model. The model is assessed in a simulation of a real world environment based on generic data sources. The Wikimedia Foundation (Wikimedia, 2009) provides detailed workload traces (Anon n.d.) of their projects. Being ranked 8th on the list of globally most accessed websites (Alexa 2009) and with a distributed infrastructure of more than 350 servers (Wikimedia Foundation 2008) Wikipedia is a representative example for large information systems. In the following analysis, the traces of the German (DE), English (EN) and Japanese (JA) versions of Wikipedia are analyzed.

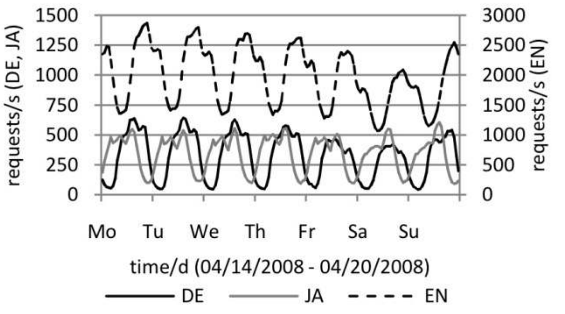


Figure 7. Wikipedia Workload Traces

Figure 7 shows a sample dataset of one week for the three Wikipedia projects. Obviously, there is a very high variation in the workload traces over time. For example, the minimum requests per second of Wikipedia Germany (DE) are as low as 45 requests per second at night, while during daytime it goes up to 660 requests per second. Specifically, the German and Japanese versions have a very low utilization at night, as their target user community is geographically concentrated.

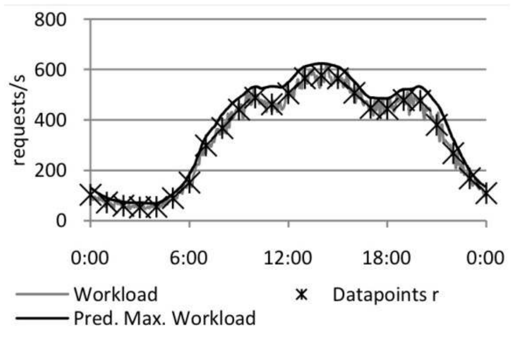


Figure 8. Workload Prediction Model

Hence, adapting the hardware configuration to the demand is likely to offer high saving potentials. Nevertheless the shape and level of the single traces vary over time. For example the characteristics of weekdays are different to weekends. Also occurrences like public holidays, have an impact on the workload trace.

4.1. Results of the Workload Prediction Model

Figure8 shows the application of the Workload Prediction Model on the Wikipedia DE project for a single day (09/23/2008). The gray line shows the simulated input processes. The marks represent the aggregated data points r_i used for the model calibration. The prediction model is setup to predict the maximum workload level in the next 15 minutes. The period length Δ is set to one hour and the calibration dataset comprises the data of the preceding twelve weeks. Empirical evaluations showed that this parameter setup delivers reliable and accurate results. The black line depicts the maximum predicted workload per second or, according to the argumentation, the number of concurrent jobs in the system. Overall, the model is able to predict the workload level accurately.

$$ROE_\alpha = \frac{1}{n}\sum_{i-1}^{n} h'_{i,} h'_i = \begin{cases} 1 & 0 \le \frac{r_i - \widetilde{r_i^c}}{r_i} \le \alpha \\ 0 & else \end{cases} \qquad (29)$$

To assess the prediction quality, the following key figure $q_{0.05}$ (29) is defined. It gives the percentage of predictions, which did not exceed the true outcome by 5% of the relative workload, whereby n donates the number of model evaluations during the regarded time frame. This gives a tangible measure for the accuracy of the prediction.

Table 2. Application of the Bottleneck Detection Model

Project	**DE**	**JA**	**EN**
$q_{0.05}$	98.75%	99.20%	98.40%

Table 2 summarizes the results of the prediction quality for one week (09/23/2008 – 09/29/2008), whereby the model has been recalibrate every second day. All three workload traces have been accurately predicted by the model.

4.2. Results of the Bottleneck Detection Model

The Workload Prediction Model provides the first input for the Adaptive Provisioning Model. The second input is given by the Bottleneck Detection Model. The following table presents exemplary results of the Bottleneck Detection Model on a RUBiS Benchmark (RUBiS, 2004). The experiments were conducted on the Emulab (White et al. 2002) . Details of the experiment setup can be found in (Malkowski et al.2007). The RUBiS benchmark is a test bed auction site modeled after eBay. In this experiment it is configured as a three-tier application, consisting of a web server, an application server and a database server. The web and application server were installed on high end systems, the database server on a low- end machine. The applied scenario consisted of a workload composition with 30% of the requests causing a database write, while the remaining 70% are sole read operations. Table 3 presents the results of the detection process. The notation (W-A-D) refers to the number of replication in each tier of the system.

Table 3. Application of the Bottleneck Detection Model

Configuration (W-A-D)	**Maximum Workload Level/ Crossover point c**	**Bottleneck Resource**
1-4-1	844	Application Server CPU
1-6-1	1400	Database Server CPU
1-8-1	1370	Database Server CPU
1-8-2	1640	Application Server CPU

The Bottleneck Detection Model correctly identifies the limiting resource in each scenario. In the 1-6-1 configuration the database server becomes the bottleneck resource. Replicating the application layer does not increase the performance. Only the replication of the database increases the overall performance of the system. These results give an idea of the performance characteristics of large systems depending on their hardware configuration.

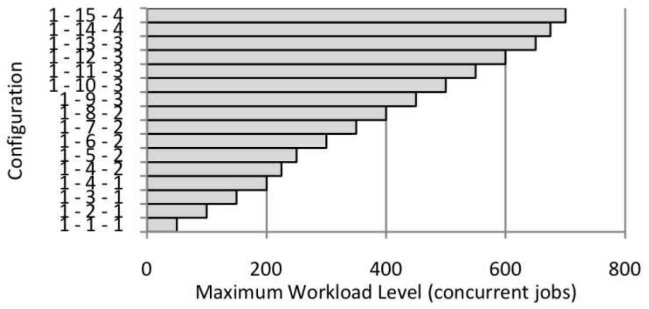


Figure 9. Configuration and Workload Levels

For the further evaluation of the Adaptive Provisioning Model, a synthetic set of configurations of the Wikipedia infrastructure and their corresponding maximum workload levels is assumed. Figure 9 shows the ascending set of configurations and their corresponding workload levels used in the further evaluation.

4.3. *Utilization depended Power Consumption*

For a realistic estimation of the saved energy, a reference server is selected. Figure 10 shows the utilization dependent power consumption of an HP Proliant DL180 G5 server (Workstation Performance 2009).

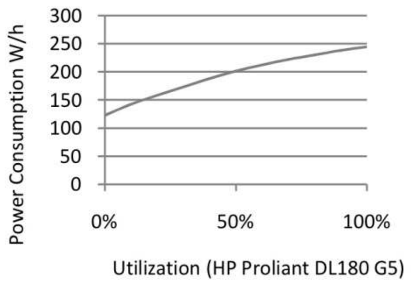
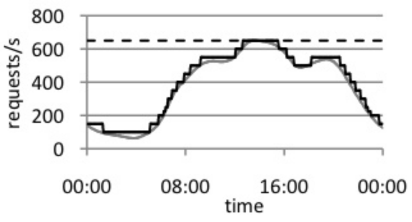


Figure 10. Utilization/ Power Consumption

The energy intake of servers partially depends on their utilization. By removing servers form the infrastructure, the utilization of the remaining machines is increased. Hence part of the energy saved by switching of machines is consumed through the higher utilization of the remaining servers. This behavior needs to be regarded in the assessment.

4.4. *Adaptive Provisioning*

The final step is the application of Tecless. Based on the results of the Workload Prediction Model and the Bottleneck Detection Model a sufficient small hardware configuration is selected at any time.



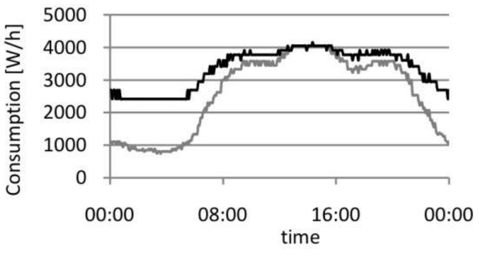


Figure 11. Adaptive Provisioning **Figure 12. Power Consumption**

Figure 11 depicts the application of the model on the same reference day as in (Figure 8). The lower gray line donates the prediction of the maximum expected workload. As the provisioning can only be modified in discrete steps, the black line shows the maximum workload level of the selected configuration according to (Figure 9). The upper, light gray line shows the static provisioning, which is set to handle 650 requests/s. It is evident, that only during peak times the maximal setup of the systems is needed. Figure 12 shows the resulting energy consumption of the system. The gray line shows the power consumption of the static system configuration. During the night, the energy intake is constant on a high level as the system utilization is close to idle. Especially during nights, the adaptive provisioning contributes significant savings, while during the peak times the consumption is equal.

Table 4. Application of the Bottleneck Detection Model

Project		**DE**	**JA**	**EN**
Average Server Utilization	**Static Configuration**	49.8%	51.8%	68.1%
	Adaptive Provisioning	79.0%	83.6%	93.7%
Average Power Consumption	**Static Configuration**	3.46 kw/h	3.99 kw/h	18.55 kw/h
	Adaptive Provisioning	2.59 kw/h	2.95 kw/h	14.17 kw/h
	Improvement	25.3%	26.0%	23.6%

Table 4 summarizes the results for all three examined projects. The adaptive provisioning increases the average server utilization of activated severs by up to 30%. Based on the utilization of the different machines and the utilization based power consumption of the reference server, the energy intake can be estimated for the static and the adaptive provisioning. On average, the energy consumption of the systems is decreased by 25%. In conclusion, the Tecless performed well on the Wikipedia Workload Traces. It is able to provide the same QoS with a significantly lower energy consumption.

5. Conclusion and Outlook

This paper presented Tecless, a novel methodology for adaptive provisioning of large enterprise systems. Tecless accounts for strong usage intensity volatilities during operation and the complexity of distributed systems. The main two objectives of the adaptive provisioning model are to guarantee a continuous QoS and to minimize the power consumption of the system. Energy conservation is achieved by adapting the size of the hardware infrastructure to the demand. Redundant servers are removed from the infrastructure and switched off because this is the most efficient way to conserve the electrical base load of the servers.

The foundation of the adaptive provisioning model consists of both the workload prediction model and the bottleneck detection model. The bottleneck model specifies how a given enterprise system needs to be scaled (e.g. by adding another application or database server) at runtime to increase the overall system performance. In other words, the bottleneck detection model provides an accurate description of the performance characteristics of an enterprise system. By iteratively analyzing the performance of a system under different stress levels, the bottleneck detection model is able to determine the maximal feasible workload of a hardware configuration as well as the resource limiting the overall performance. However, reconfiguration decisions are time intensive due to the activation delay of servers. Hence, modifications of the infrastructure need to be triggered in advance, which requires to forecast the system stress.

The workload prediction model attempts to forecast the workload process exactly. By analyzing the behavior of past user behavior, the model predicts the level of workload in the near future. The single factors of influence are isolated and estimated separately with the help of a modified time series decomposition model. To account for unforeseen changes in the workload process, the model is equipped with a feedback mechanism. This novel methodology offers a precise forecast of the workload process with an accuracy of up to 99 percent.Evaluations of the adaptive provisioning model on generic data have shown an energy saving potential of up to 25 percent, without significantly influencing the QoS. Given the steady increasing operation cost of datacenters, this methodology can help to slow down this growth process. Consequently, Tecless might reduce the impact on the environment and improve the economical efficiency of enterprise systems.

The evaluations of the adaptive provisioning model are very promising. The next step is the implementation of the model in a testbed. This will further confirm the validity of the model and investigate potential implementation challenges. Furthermore, we plan some extensions of the model. In the current version, the feedback mechanism is solely based on the prediction error. The use of more reactive machine learning feedback mechanism might enable the usage of the model in systems, which show stronger irregularities in their workload process.

6. References

Alexa, 2009. Alexa Top 500 Sites. Amazon.com Inc. Available at: http://www.alexa.com [Accessed November 3, 2011].

Alonso, G. et al., 2002. *Web Services*, Springer-Verlag.

Anon, Wikistats. Available at: http://dammit.lt/wikistats/ [Accessed August 30, 2011].

Bash, C.B., Patel, C.D. & Sharma, R.K., 2006. Dynamic thermal management of air cooled data centers. In *The Tenth Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronics Systems, 2006. ITHERM '06*. San Diego, CA, pp. 8-452.

Brockwell, P.J. & Davis, R.A., 2002. *Introduction to Time Series and Forecasting* G. Casella, S. Fienberg, & I. Olkin, eds., Springer.

Brunschwiler, T. & Michel, B., 2008. Thermal Management of Vertically Integrated Packages. In C. B. and P. R. P. Garrou, ed. *Handbook of 3D Integration: Technology and Applications of 3D Integrated Circuits*. Weinheim, Germany: Wiley-VCH Verlag GmbH & Co. KGaA, pp. 635-649.

Clidaras, J. & Stiver, D.W., 2010. WATER-BASED DATA CENTER. *WO Patent WO2010129341*.

Cohen, Ira et al., 2004. Correlating instrumentation data to system states: a building block for automated diagnosis and control. *Operating Systems Design and Implementation*, p.16. Available at: http://portal.acm.org/citation.cfm?id=1251270 [Accessed September 15, 2010].

Cooley, J., 2009. Sun ' s Project Blackbox: A Modular Approach to High Density Datacenters Who I am • Director of Engineering in Sun ' s Systems Group. *Spring*.

Dhiman, G. & Rosing, T., 2006. Dynamic Power Management Using Machine Learning. *2006 IEEEACM International Conference on Computer Aided Design*, pp.747-754.

EPA, 2008. Energy Star Computer Requirements for Computers Version 5.0 pp. EPA.

Ghanbari, S. et al., 2007. Adaptive Learning of Metric Correlations for Temperature-Aware Database Provisioning. *Fourth International Conference on Autonomic Computing ICAC07*, pp.26-26.

Hedwig, M. et al., 2010. *Datacenter Investment Support System (DAISY)*, IEEE. Available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5428555.

ITP, 2007. Data Center Energy Efficiency: Turning Challenges into Opportunities.

Jung, G. et al., 2006. Detecting Bottleneck in n-Tier IT Applications Through Analysis. *Large Scale Management of Distributed Systems*, pp.149-160. Available at: http://dx.doi.org/10.1007/11907466_13.

Kersten, R., 2007. Wieviel CO2 erzeugt eine EBAY-Auktion. *Sun Microsystems*. Available at: http://mediacast.sun.com/users/rolfkersten/media/EBAY_Keynote2.pdf [Accessed 2200].

Kleinrock, L., 1975. QUEUEING SYSTEMS, VOLUME 1: THEORY by Leonard Kleinrock John Wiley & Sons, Inc., New York, 1975, 19.95, 417 pages. *Networks*, 6(2), pp.189-190.

Koomey, J.G., 2007. Estimating total power consumption by servers in the U.S. and the world. *World*. Available at: http://www.greenbiz.com/research/report/2007/09/12/estimating-total-power-consumption-servers-us-and-world.

Kreiß, J.-P. & Neuhaus, G., 2006. *Einführung in die Zeitreihenanalyse*,

Kurp, P., 2008. Green computing. *Commun. ACM*, pp.11-13.

Linderman, R. et al., 2008. High Performance Thermal Interface Technology Overview. *2007 13th International Workshop on Thermal Investigation of ICs and Systems THERMINIC*, (September), pp.129-134.

Malkowski, S. et al., 2007. Bottleneck Detection Using Statistical Intervention Analysis. In A. Clemm, L. Z. Granville, & R. Stadler, eds. *DSOM '07*. Berlin, Heidelberg: Springer-Verlag, pp. 122-134.

Moore, J. et al., 2005. Making scheduling "cool": temperature-aware workload placement in data centers. In *annual conference on USENIX*.

Powers, R., Goldszmidt, Moises & Cohen, Ira, 2005. Short term performance forecasting in enterprise systems. *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining - KDD '05*, p.801. Available at: http://portal.acm.org/citation.cfm?doid=1081870.1081976.

Puttaswamy, K. & Loh, G.H., 2006. Thermal analysis of a 3D die-stacked high-performance microprocessor. *Proceedings of the 16th ACM Great Lakes symposium on VLSI GLSVLSI 06*, p.19.

Ranganathan, Parthasarathy et al., 2006. Ensemble-level Power Management for Dense Blade Servers. *ACM SIGARCH Computer Architecture News*, 34(2), pp.66-77.

Schulz, G., 2009. *The Green and Virtual Data Center*, Boston, MA, USA: Auerbach Publications.

Sharma, R K et al., 2005. Balance of Power: Dynamic Thermal Management for Internet Data Centers. *IEEE Internet Computing*, 9(1), pp.42-49.

Sharma, Ratnesh K et al., 2008. On building next generation data centers: energy flow in the information technology stack. *Proceedings of the 1st Bangalore annual Compute conference*, p.8.

Singh, A., Hayward, B. & Anderson, D., 2007. Green IT Takes Center Stage.

Urdaneta, G., Pierre, G. & Van Steen, M., 2007. Wikipedia workload analysis for decentralized hosting. *Computer Networks*, 53(11), pp.1830-1845.

Urgaonkar, B., Pacifici, G., et al., 2005. An analytical model for multi-tier internet services and its applications. *SIGMETRICS Perform. Eval. Rev.*, 33(1), pp.291-302.

Urgaonkar, B. et al., 2007. Analytic modeling of multitier Internet applications. *ACM Transactions on the Web*, 1(1), p.2-es. Available at: http://portal.acm.org/citation.cfm?doid=1232722.1232724.

Urgaonkar, B. et al., 2008. Agile dynamic provisioning of multi-tier Internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1). Available at: http://portal.acm.org/citation.cfm?id=1342172.

Urgaonkar, B., Shenoy, P., et al., 2005. Dynamic provisioning of multi-tier Internet applications. *Second International Conference on Autonomic Computing*.

Velte, T., Velte, A. & Elsenpeter, R.C., 2009. *Green IT: Reduce Your Information System's Environmental Impact While Adding to the Bottom Line*, New York, NY, USA: McGraw-Hill, Inc.

Viswanathan, L.P. & Monie, E.C., 2006. Reinforcement temporal difference learning scheme for dynamic energy management in embedded systems. In *VLSI Design, 2006. Held jointly with 5th International Conference on Embedded Systems and Design*. p. 6.

White, B. et al., 2002. An integrated experimental environment for distributed systems and networks. *ACM SIGOPS Operating Systems Review*, 36(SI), p.255.

Wikimedia Foundation, 2008. Wikipedia:Server. *Wikimedia*. Available at: http://de.wikipedia.org/wiki/Wikipedia:Server [Accessed January 9, 2009].

Won, Y., Kim, J. & Jung, W., 2008. Energy-aware disk scheduling for soft real-time I/O requests. *Multimedia Systems*, 13(5), pp.409-428.

Workstation Performance, 2009. Standard Performance Evaluation Corporation SPEC ' s Benchmarks and Published Results. *Power*.

Zhang, S. et al., 2005. Ensembles of models for automated diagnosis of system performance problems. *DSN*.

CHAPTER III

GREEN OPERATION OF ENTERPRISE INFORMATION SYSTEMS: DYNAMIC AND FLEXIBLE SLA MANAGEMENT

Working Paper – Parts of the paper have been published in:

Hedwig, Markus; Malkowski, Simon and Neumann, Dirk, "Towards Autonomic Cost-Aware Allocation of Cloud Resources" (2010), International Conference on Information Systems 2010 Proceedings.

Abstract

The growing awareness that Green Information System (IS) solutions, which contribute to sustainable business processes, secure a long-lasting competitive advantage has increasingly focused corporate transformation efforts on the efficient and effective usage of Information Technology (IT). In this context, we provide a green perspective on enterprise IS operation and introduce a novel IS adaptation framework that harmonizes green goals with the business value chain. We concretely target elastic n-tier applications with dynamic on-demand cloud resource provisioning for component servers (e.g., application and database servers) and implement an adaptation engine prototype. Our framework forecasts future user behavior based on historic data, analyzes the impact of a workload on system performance based on a non-linear performance model, analyzes the economic impact of different provisioning strategies, and derives an optimal operation strategy. More generally, our adaptation engine optimizes IT system operation based on a holistic evaluation of the key aspects of the business value chain (e.g., system usage patterns, system performance, and Service Level Agreements) in accordance with the Green IS paradigm. The evaluation of our prototype, based on a real production system workload trace, is carried out in a custom test infrastructure (i.e., cloud testbed, n-tier benchmark application, distributed monitors, and control framework). We systematically investigate practicability, IS optimization potential, green effectiveness, and business value chain orientation, and we show that the integration of our adaptation engine allows flexible IS operation with up to a 46 percent lower cost of operation.

Keywords: *Green IT, Green IS, Performance Model, Workload Forecast, Service Level Agreements*

1. Introduction

“Being Efficient”—How well this maxim is implemented by a company is often the deciding factor that separates market leaders from market followers. In today’s highly dynamic business environments, agile and effective adaptation in response to new market conditions, changed customer requirements, and disruptive technologies is the key to sustainable success. While there is no “silver bullet” that guarantees successful companies to permanently solidify their competitive advantage, there are, however, various prominent best practices, which show that continuous transformation efforts result in long-lasting market leadership.

Although the concrete implementation of transformation efforts in companies may differ strongly among different companies and industries, such efforts commonly share their core principles. In fact, the majority of transformation efforts target holistic optimization of the entire business value chain in addition to emphasizing customer and market orientation. For instance, Customer Relationship Management (CRM) systems allow synchronizing customer requirements with business processes in the manufacturing industry. Similarly, Lean Management principles dictate the strict alignment of the production process to actual customer demands (King & Lenox 2009).

Interestingly, these aforementioned modern management philosophies have initially not considered the operation and management of Information Technology (IT) infrastructure as a first-class citizen in their optimization efforts (Carr 2004). The main reasons for neglecting IT infrastructures were twofold; first, the concept of IT as an asset is relatively young; second, the complexity of IT systems and IT infrastructure often results in non-transparent costs, which is perceived as high risk. As a consequence, missing awareness and a lack of best practices further contributed to the perceived risk and have caused IT strategies to neglect efficiency for years (Carr 2004).

Roughly a decade ago, the role of enterprise IT has changed dramatically, fueled by the observation that enterprise IT infrastructures accounts for up to ten percent of total corporate expenditures. Moreover, empirical analysis showed that datacenters solely achieve an average utilization of around twenty, which implies that the majority of the existing computing potential remains unused (i.e., irrecoverable potential loss). Consequently, many efficiency-centered IT trends emerged, which culminated in the dawn of Green IT as a result of the conjunction of IT efficiency and ecological responsibility. Among the many facets of Green IT, which all attract considerable corporate and academic attention, are (i) design and deployment of energy-efficient hardware, (ii) operation of datacenters in areas with favorable environmental attributes (e.g., cold climate), and (iii) efficient and dynamic provisioning of IT infrastructures on demand by means of consolidation, virtualization, and cloud technology (Schulz 2009).

Since it has shown that despite its broad success, Green IT can—at best—slow down the currently observable rapid growth in corporate IT expenditures (Koomey 2007), the next wave of efficiency awareness in IT will be sensible utilization. In fact, the operating costs of large enterprise applications in datacenters are still continuously increasing at substantial rates, and in contrast to all cost reduction efforts such as consolidation, modern companies still tend to expand their overall IT system and infrastructure sizes. Consequently, the awareness that advanced IT solutions secure a competitive advantage only for a short period of time and that *sensibly utilizing IT* is the key to long-lasting success is growing. In this context, the overarching management philosophy, reflected by the term Green IS, focuses on the efficient and effective usage of IT and IT infrastructure. While Green IT is considered to be the solution to the problem of the rapidly growing resource consumption through increased energy efficiency and higher resource utilization, Green IS *“refers to the design and implementation of information systems that contribute to sustainable business processes”* (Watson 2007).

It is a well-established fact that Green IT has already significantly reduced the environmental footprint of corporate IT (e.g., energy optimized datacenters); nevertheless, Green IT initiatives will continue to fall short of their full potential as long as they do not unify their efforts to optimize the operation of IT and the application of IT. In line with this perception of Green IS, we argue that enterprise IT systems have to be strictly aligned with the business value chain. Analogous to concepts such as just-in-time production, this alignment requires a thorough analysis of the business value chain of IT systems. IT systems are typically offered as a service under certain Quality of Service (QoS) assertions. Therefore, the business value chain of IT systems is closely related to the value chain of services in the traditional sense. Because there is a design phase in which the offered service is specified, the design is affected by the service system that contains the IT infrastructure. As a result of this design, service offerings—the Service Level Agreements (SLAs)—specifying the QoS assertions, which reflect the capabilities of the service system including the IT infrastructure, are created. In the operation phase, the service is offered to customers according to these SLAs.

Although SLAs have become the de-facto industry standard for defining the terms of operation in IT infrastructures, including revenue for maintaining the services, penalties for QoS violation, monitoring metrics, and corresponding thresholds, SLAs are predominantly treated statically. More concretely, SLAs are translated into a static set of requirements at the design time of the IT services. During runtime, IT infrastructures are then provisioned according to the maximum predicted peak load, which results in very low resource utilization on average.

From a Green IS perspective, a static treatment of SLAs is highly ineffective because it's oblivious of the actual business value chain. Moreover, for small-sized systems, virtualization and consolidation are sufficient for adjusting the IT infrastructure size to the fluctuating demand; however, for mid-sized and large systems, consolidation becomes non-trivial due to the complexity of the underlying infrastructure, and dependencies may become prohibitively difficult to manage. Therefore, the Green IS philosophy mandates the integration of SLAs into agile management at runtime and the alignment of SLAs with the business goals (e.g., anticipated future performance of the IT system). While the first aspect is also highly relevant during the design phase (i.e., SLA definitions must be congruent with the true business value and service requirements), the latter must be an integral component of the operation strategy.

In this paper, we develop an adaptation engine framework that is able to align the business value chain with the operation of the IT infrastructure according to the Green IS paradigm. Our adaptation engine framework optimizes IT system operation based on a holistic evaluation of all key aspects of the business value chain (i.e., system usage patterns, system performance, and SLAs). In other words, the adaptation framework forecasts future user behavior based on historic data, analyzes the impact of workload on system performance based on a non-linear system performance model, analyzes the economic impact of different resource provisioning strategies, and derives the optimal operation strategy based on all known and predicted information. Because the adaptation engine analyzes the technical system behavior in conjunction with the current workload process, correlation analysis based on the SLAs yields very high efficiency.

Our adaptation engine prototype is designed based on the goal of economical operation of large enterprise IT systems, which are typically distributed. The distributed architecture has three major advantages. First, commodity hardware can be utilized; second, standard modular software components can be deployed; third, scalability of different application parts becomes decoupled. In fact, the functionality of different application modules is separated into separate layers, whereby each layer is responsible for conducting a designated task (e.g. persistent data storage or application logic).

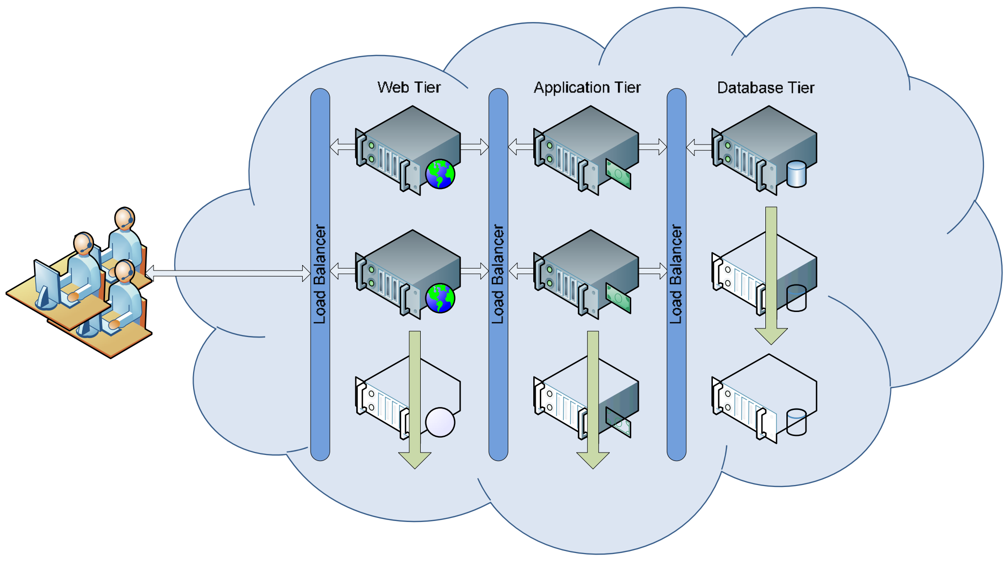


Figure 1: Elastic n-tier application system design

This modularization principle is referred to as *n-tier* application design, where *n* denotes the number of layers. In the later part of the paper we utilize a three-tier system consisting of a web server tier, an application server tier, and a database server tier (Figure 1). Typically, each tier is deployed on separate hardware nodes or separate virtual hardware nodes. Depending on the complexity of the application and the hardware characteristics, the system is able to handle a specific workload level. By replicating servers in the different tiers and adding additional computing resources to them, the maximum sustainable workload level can be increased. The actual load management within a specific tier is typically handled by load balancers (e.g., round-robin request routing).

Elastic applications have emerged as the next generation of n-tier systems. They follow the same design paradigm; however, they provide extensions enabling resource adaptive operation modes. In particular, this design is highly beneficial for cloud environments as it allows utilizing the capability of the cloud to instantaneously allocate resources to systems. In order to evaluate the proposed adaptation engine, we have developed a complete test infrastructure. The high complexity of enterprise system environments usually prohibits proving the validity solely on data analysis. Instead, the validity has to be proven experimentally. Our test system consists of a cloud infrastructure, a benchmark application, extensive monitoring and control software, and the adaptation engine itself. In order to provide a real-world evaluation scenario, we used a real production system workload process to generate our test workload. Additionally, we have deployed our test system on state-of-the-art hardware with enterprise class software.

Our evaluations indicate that the integration of our adaptation engine allows reducing resource consumption of the IT systems under test by up to 40 percent. Furthermore, our adaptation engine prototype allows operating the systems with higher flexibility and adapting dynamically on demand. The evaluation shows that our framework does not only reduce the resource consumption of IT systems, but it also enables the system to autonomically react to changes in its environment. Hence, our integrated resource management approach allows managing enterprise applications in accordance with their business value. The main contribution of this work is threefold.

1. We provide a green perspective on enterprise IT systems and introduce a novel framework that harmonizes green goals with the business value chain.

2. Our implementation prototype provides a proof of concept for operation of enterprise class applications close to their maximal economic efficiency.

3. The integration of our prototype into a test environment allows us to evaluate our framework in depth in terms of efficiency along the SLA lifetime. Based upon the evaluation we can make concise recommendations how to use the framework.

The remainder of the paper is structured as follows. The second section derives the adaptation engine framework. The third section expands on the adaptation engine framework by discussing the model steps in detail. Section 4 presents the evaluation of the adaptation engine demonstrating its usefulness. Section 5 summarizes the paper and provides a conclusion and an outlook on our future work.

2. Adaptation Engine Framework

As mentioned in the introduction, this paper aims at developing an adaptation engine framework that is able to align the business value chain with the operation of the IT infrastructure according to the Green IS paradigm. The Adaptation Engine Framework represents an artifact and strictly follows the design science paradigm suggested by (Hevner et al. 2004). In this section, we will provide important theoretical background on resource management that underlies our design problem. In addition, we will use the background to identify the main factors that influence the system performance of an IT system. Those so-called factors of influence constitute our setscrews of the IT infrastructure we can use to improve the overall system. Dependent on these factors, we design a framework of the adaptation engine that combines all factors of influence into a single artifact. In this paper, the framework describes all the used factors and concepts as a coherent mathematical model. In addition, the mathematical model has been implemented as a proof-of-concept prototype that will be used later on in the evaluation to demonstrate that the adaptation engine is successful in reducing the ecological footprint by increasing the efficiency of the IT infrastructure.

2.1. Background and Design Problem Statement

Technical advances, demanding customer expectations, as well as increasing system complexity, make sustainable operation of IT systems a challenging task. In particular mission critical systems belong to the sensible corporate infrastructure, and consequently reliability and availability are valued significantly higher than cost-effective operation modes.

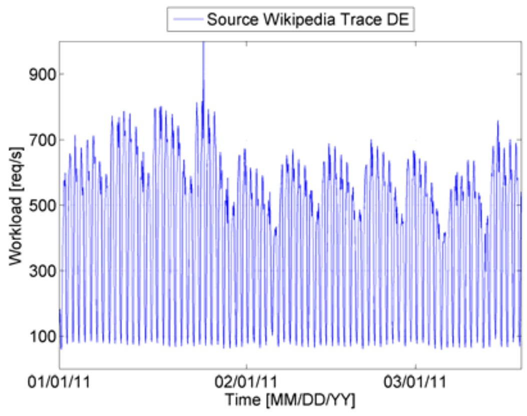


Figure 2: Wikipedia DE Workload Trace

As a consequence, classic system design paradigms prescribe to scale systems to the maximum expected workload. This inevitably leads to the situation where system operators tend to overprovision their IT (i.e., usage of more hardware resources than necessary) in order to guarantee continuous high quality of service. Many end-user applications face highly volatile workloads, as the workload trace of Wikipedia Germany demonstrates. Evidently, the workload level varies by more than one magnitude within a single day. One way to increase the utilization of enterprise applications, rather than static operation, at the maximum peak level is the use of Resource Management Systems. These management tools usually supervise the performance of one or multiple systems and allocate resources according to the user demand. While these approaches work well for small-sized systems (i.e., applications which at most require one server), their application in large environments is fraught with problems. Similar to the concept in small systems, the idea in this domain is to allocate resources according to the demand. By dynamically adding and removing resources (e.g. servers) from the system, the average utilization of the system can be significantly increased while providing at the same time a high Quality of Service and hence contributing to highly efficient operation modes. However, the optimal adaptation of the system is non-trivial. More specifically, one key challenge is to overcome the reconfiguration lead time. Usually, the hardware configuration cannot be adapted instantaneously, rather there occurs a delay between the initiation of reconfiguration and its availability due to configuration of the resources, synchronization and reconfiguration of the load balancers.

The delay time is a result of different tasks, which need to be performed prior to the use of new hardware, such as resource provisioning time or synchronization and reconfiguration tasks. Nowadays, Resource Management Systems are rather reactive controllers (i.e., they only react to changes in their environment) and thus exhibit an inherent risk of performance violations due to the lead-time. *In this paper we attempt to design a practicable green operation model – the adaptation engine framework – that optimizes IT system operation based on a holistic evaluation of all key aspects of the business value chain.*

2.2. Factors of Influence

The inclusion of the whole business value chain requires a thorough understanding about all factors of influence that affect our green operation model. Even though the idea of resource-adaptive operation of large enterprise applications seems to be very tempting, its application is absolutely non-trivial. This complexity stems from the fact that the overall performance of systems depends on various, interdependent factors; changes in each factor can lead to critical performance limitations or – in the worst case – cause total system failure. In order to manage a system efficiently, the factors of influence need to be well understood. Accordingly, we can categorize the following factors of influence (Figure 3).

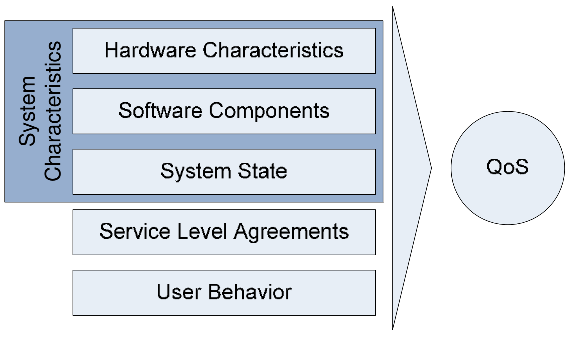


Figure 3: Factors of influence

- **User behavior:** The term workload specifies the amount of labor the system has to perform. In our scenario the workload refers to requests from the users or systems. Apparently, user behavior is responsible for the system load, where both the workload level as well as the composition of the workload (e.g. different applications may exhibit different usage patterns) have a strong impact. Due to the stochastic nature of user behavior, it can be deduced that the operation of the infrastructure needs to be dynamic, adapting to the actual user requests.

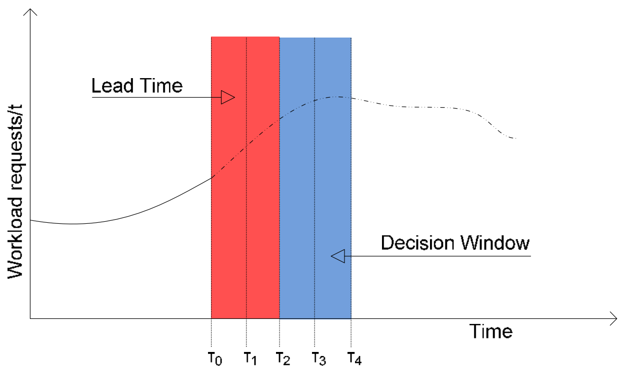


Figure 4: Reconfiguration lead time problem

- **Hardware Characteristics:** The hardware and infrastructure characteristics (.e.g. amount of RAM, disk space, CPU speed, hardware design) naturally determine the physical computation power of the system. Moving from one hardware configuration, say one web server, one application server and one database server to another (e.g. adding one application server to the old configuration) incurs some lead time. This lead time can be substantial when data is involved that needs to be synchronized.
- **Software Characteristics:** The software, including all components such as the operating system and virtualization software, is responsible for the hardware utilization. Physical limitations of the hardware can be overcome by smart code programs (e.g. implementation of virtual memory). However, in many cases programming is not an option, as the software is in many cases purchased from third-party vendors.
- **System State:** Next to hard- and software characteristics, the system state also significantly affects the overall system performance. For instance transactions on a highly populated database probably demand more computational power than the

same transaction on a small dataset. The system state is difficult to isolate from the hard- and software characteristics and is thus frequently omitted in related work.

- **Service Level Agreements:** As aforementioned, Service Level Agreements specify all aspects of the business relation between the contract partners, such as the rights and duties of each party, contract duration as well as guarantees and warranties. With the definition of the rights and duties the SLA defines what system performance really means. The SLA describes the contract partners, being the provider and the consumer. The QoS requirements are distinguished into Service Level Objectives (SLOs), Monitoring Intervals and SLA assertion. SLOs are specific measurable characteristics of the system (e.g. availability, throughput, response time). Generally, SLOs also define a target value for this characteristic. The monitoring intervals define when compliance with the SLO is checked. The SLA assertion can combine several SLOs and specifies under what conditions the SLA is violated. In addition, the SLA defines the duration of the SLA as a contract and also controls the penalty payments in case of SLA violation. Different SLAs may lead to very different operation strategies. For example, a low response time system (e.g. customer shop portal) will impose more cost-intensive requirements (i.e. strict SLOs and high penalties) than a standard system does.

As demonstrated, all these factors ultimately affect the overall system performance. However, most of the factors cannot be adjusted at operation time in the short run. Hard- and software endowment can be assumed to be constant in the short-run, as hardware upgrades or software updates have a long lead time. Essentially, the system state is also constant in the short run, as it evolves gradually over time with sudden changes very unlikely to occur. As hard- and software characteristics and the system state are all fixed in the short run, we aggregate them into the factor ‘system characteristics’. Dependent on the operation environment – being stable in the short run – the system provider tailors services by the instantiation of SLAs at design time and offers it to customers.

In the operation phase, customer requests arrive stochastically at the service provider. From an operation management perspective it is imperative to reduce the costs of provisioning the service in order to achieve operational efficiency. The only feasible option to the service provider is to dynamically manipulate the number of resources (e.g. virtualized servers) that is being used. All other factors are stable in the short run and can thus not be influenced. Dynamically manipulating the hardware corresponds with a resource-adaptive operation mode, which appears to be the setscrew of the service provider to improve the efficiency of the system. By including the SLAs directly into the operation mode, the service provider has a tool in hand to align the provisioning strategy with the business goals. From a technical perspective, this implies that the SLA is directly included into the operation strategies. In case of an important high-end service, the incorporation of the SLA in the provisioning assures a very sensitive provisioning such that the QoS assertions are constantly met.

For resource-adaptive operation modes, the workload appears to be the key factor. Since the workload is exogenously given, the service provider has only limited flexibility – within the boundaries of the SLA – to reduce the workload. As reconfiguration can be associated with considerable lead-time, resource-adaptive operation modes can only be effective when the workload forecast is accurate at any point in time.

Similar resource-adaptive operation modes have already been published in the field of dynamic research management. For instance, the authors in (Gmach et al. 2009) developed a reactive migration controller for virtualized environments. The paper (Chandra et al. 2003) presents a resource allocation model for shared datacenters based on a queuing network performance model and a time series workload forecast mechanism. Another concept (Padala et al. 2009) is an automated control model for virtual resources. The model manages the varying resource demands by dynamically allocating resources or migrating virtual machines. In (Ardagna et al. 2007) a model has been developed to manage the resource demand of multiple concurrent systems. In contrast to our work, these models focus on the technical implementation rather than taking the Green IS perspective.

3. Adaptation Engine Architecture

Having explained the factors of influence and their impact on system performance, we will develop a conceptual framework that integrates all those factors into a comprehensive model. The framework attempts to formulize all factors in a mathematical way; in addition, the degree of formulization allows the framework to serve as an architecture for a design artifact, which is henceforth denoted as Adaptation Engine.

Figure 5 shows our conceptual framework on a high level. The factors Service Level Agreements SLA, the system characteristics SC consisting of the hardware HW, software SW and the system state SS determine the performance of the system. The performance model specifies how many users can simultaneously access the service without violating the SLA. The factor user behavior UB affects the workload of the system. The Operation Strategy S defines the operation mode based on the aforementioned input factors. The workload model uses past workloads to forecast future workloads of the system. The Decision model confronts the system model with the forecast model and derives in conjunction with the operation strategy the right hardware configuration.

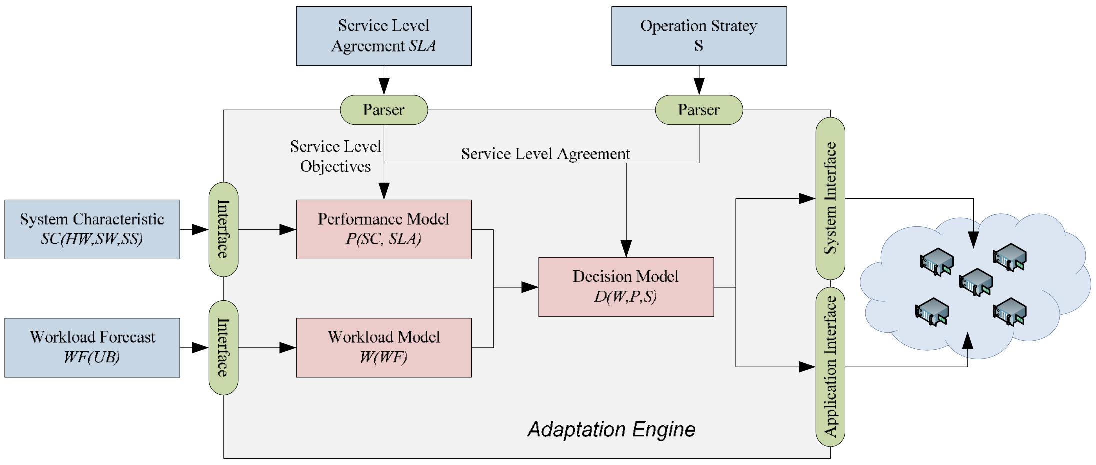


Figure 5: Adaptation Engine Architecture

Perceiving the Adaptation Engine as artifact, it is designed as a software component with clear defined interfaces to interact with its environment. In the figure, the Adaptation Engine is depicted as the shaded box containing the internal models consisting of the performance, workload and decision model.

The interfaces of the Adaptation Engine include the inputs workload, system characteristics and SLAs. From a technical point of view, the adaptation engine encompasses the actuator interfaces, which initiate the reconfiguration of the infrastructure (i.e. add and remove servers) as well as the reconfiguration of the application (i.e. configure load balancers). The

Adaptation Engine itself consists of three different internal models that convert the input information into configuration decisions. The workload model uses past workload traces to predict the workload in the short and mid range. The framework imposes requirements on the input and output data, but not on how the forecast is being made. The idea of the Adaptation Engine is that any conceivable forecasting component, such that complies with the data requirements, can be plugged into the software component. For instance (Gmach et al. 2007) used Fourier Transformation for data smoothing and applied time series analysis for workload prediction. In (Powers et al. 2005) the authors used time series as well as regression analysis for workload prediction. This design assures that the Adaptation Engine does not rely on our models that will be presented in Section 3, but allows incremental improvement as long as the data requirements of inputs and outputs are satisfied. The performance model converts information about the operation environment and the SLAs into a table that juxtaposes average SLA compliances and hardware configurations. Based on the outputs of the performance and workload model, the Decision model aims at deriving the optimal green operation strategy.

3.1. *Internal Adaptation Engine Model Design*

Having described the intuition of the Adaptation Engine Framework as generic design architecture, we will first define the mathematical representations of the aggregated workload and the system performance. Subsequently we show how this information is converted into the internal models in order to derive a green resource adaptive operation mode.

$$W = \begin{array}{c|ccc} & \tau_0 & \cdots & \tau_m \\ w_1 & x_{w_1\tau_0} & \cdots & x_{w_1\tau_m} \\ \vdots & \vdots & \ddots & \vdots \\ w_n & x_{w_n\tau_0} & \cdots & x_{w_n\tau_m} \end{array} \quad (1)$$

$$W = \begin{array}{c|cc} & \tau_0 & \tau_1 \\ w_1 & 0\% & 4\% \\ w_2 & 100\% & 23\% \\ w_3 & 0\% & 43\% \\ w_4 & 0\% & 30\% \end{array} \quad (2)$$

The internal models of the Adaptation Engine are discrete, which reflects the fact that monitoring occurs at discrete time steps as well. The internal workload model W is designed as a matrix of size n x m (1). The rows represent small step workload levels w, whereas the columns represent the forecast period τ. The element $x_{w\tau}$ is the probability to face the workload level w in the forthcoming period τ. The current workload level is given in the first column. As it is certain, the corresponding row is set to 1, whereas all other elements in the first column are set to zero. The matrix W is illustrated in (2). The first column represents the current workload. As there is no uncertainty involved, workload level w_2 has a probability of 100%; all other probabilities are set to 0. For the next period τ_1, only estimates are available. Accordingly, the second column contains a distribution estimate of the workload, where a workload level of w_1 has a probability of 4%.

$$P = \begin{array}{c} \\ c_1 \\ \vdots \\ c_k \end{array} \begin{array}{c} \begin{array}{ccc} w_1 & \dots & w_n \end{array} \\ \begin{pmatrix} x_{p_1 w_1} & \cdots & x_{p_m w_m} \\ \vdots & \ddots & \vdots \\ x_{p_k w_1} & \cdots & x_{p_k w_m} \end{pmatrix} \end{array} \quad (3)$$

$$P = \begin{array}{c|cccc} & w_1 & w_2 & w_3 & w_4 \\ \hline c_1 & 0\% & 0\% & 0\% & 0\% \\ c_2 & 15\% & 14\% & 12\% & 9\% \\ c_3 & 93\% & 92\% & 92\% & 88\% \\ c_4 & 99\% & 99\% & 98\% & 97\% \\ c_5 & 100\% & 100\% & 100\% & 100\% \\ c_6 & 100\% & 100\% & 100\% & 100\% \end{array} \quad (4)$$

The Performance Model P has a similar structure and provides the performance capabilities of the different configurations c_k facing different workload levels w_n. The element x_{pw} reflects the probability that the configuration c can satisfy a given SLA facing workload w. Expression (4) illustrates an example of matrix P. The c levels I-VI codifies different system configurations. Configuration I for example represents a system consisting of one web-server, one application server and one database server. For any workload level w_1-w_4 this configuration cannot attain the given SLA at all. Thus, any entry of the first row is zero. Configuration II encompasses one web-server, two application servers and one database server. Facing a workload level w_1 configuration II satisfies the SLA with a 15 % probability.

$$D = PW = \begin{array}{c} \\ p_1 \\ \vdots \\ p_n \end{array} \begin{array}{c} \begin{array}{ccc} \tau_0 & \dots & \tau_m \end{array} \\ \begin{pmatrix} x_{p_1 \tau_0} & \cdots & x_{p_m w \tau_m} \\ \vdots & \ddots & \vdots \\ x_{p_n \tau_0} & \cdots & x_{p_m \tau_m} \end{pmatrix} \end{array} \quad (5)$$

$$\begin{array}{c} \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{array} \begin{array}{cc} \tau_0 & \tau_1 \\ \left(\begin{array}{cc} 0\% & 0\% \\ 14\% & 11\% \\ 92\% & 90\% \\ 99\% & 98\% \\ 100\% & 100\% \\ 100\% & 100\% \end{array}\right) \end{array} \quad (6)$$

Both matrices P and W have a similar structure, which allows their multiplication. The resulting Decision Matrix D (5) aggregates the Workload Forecast and System Performance data into the matrix D of size $n\ x\ m$. This matrix contains the probability that a configuration p is able to sustain the QoS assertions of the SLA given an expected workload in period τ. Essentially, the matrix multiplication facilitates that any x_{pw} (i.e. probability that a given configuration p satisfies an SLA facing workload w) is weighted with the probability that workload w is daunting in the next period τ. Expression (6) illustrates matrix D; accordingly, configuration I has a zero probability of satisfying the SLA in period τ_1, configuration II has an 11% probability to meet the SLA in τ_1, whereas configuration III has a probability of 90%.

The Decision matrix D forms the basis for any green operation strategy. The strategy function uses matrix D as fact base to derive an efficient operation strategy for the near future of the system. We define our operation strategy c* as a function of matrix D: $c^* \leftarrow S(D)$. For example, our strategy c* may formulate the use of the configuration that has a probability of more than 99% to satisfy the SLA in the next period. Expression (6) shows that configuration V will be selected for period τ_1. In the following subsections, it is illustrated how we obtain these matrices and how we use them to define the operation strategies.

3.2. *Workload Model*

Modern elastic applications allow a resource adaptive operation without compromising system stability. However, due to significant lead-times in the reconfiguration, resource adaptations need to be initiated in advance. Depending on the replicated resources this can range to several hours taking large database synchronizations into consideration. Thus, in order to mitigate the risk of SLA violations or system crashes, the systems cannot be managed on the basis of the current state, but require near-future workload predictions. At heart, workload processes (i.e. particular workload generated by a large number of users) are stochastic processes consisting of trend and seasonal factors as well as unpredictable anomalies such as sudden appearances of users (flash-crowd effect). While trend and seasonal factors can be estimated with workload models based on historic data, unpredictable anomalies can usually be detected by their appearance only.

Having continuously obtained the most recent monitoring data, our forecast mechanism predicts the workload for the next periods based on the discrete Fourier Transformation. Depending on the lead-time and potential other factors (e.g. pricing models for the resources dependent on the SLA), the prediction horizon is usually between a few minutes and several hours. Our forecast method is designed for workload processes generated by a large number of users with predominant seasonal components as it is the case for elastic applications. The forecast mechanism itself aims at detecting and predicting all effects on the workload that can be extracted from the observed user behavior. An anomaly detection mechanism is omitted in favor for a global reactive controller, which detects all types of abnormalities in the system.

3.2.1. Forecasting Model

The Fourier Transformation in general exchanges the mathematical base of process from the time domain into the frequency domain. In order to apply the Fourier Transformation and to prevent extensive usage data storage, the process is aggregated into a time series $W^p =< w_1, \dots, w_n >$ of total number of requests per periods of length Δ. The workload forecast model constitutes an online mechanism, where the forecast is updated after each period Δ. For simplicity, in the following presentation the execution time index t is omitted. For our example of elastic applications, the length of one period Δ lasts from 5 to 30 minutes and the series W^p comprises historic data of up to one month aggregated into n data points. With the help of the discrete Fast Fourier Transformation dFFT, the workload process is converted to the frequency domain.

$$F =< f_1, \dots, f_n >= dFFT(W) \text{ with } f_l = r_l(\cos(\phi_l) + i * \sin(\phi_l)) \quad (7)$$

The result of the dFFT is a vector of the same length as the original time series W. Each element of the vector represents one part of the spectrum. While the frequency is defined by the position in the vector l, the amplitude r_l and phase shift ϕ_l of each element is specified as a complex number. Compared to other forecast mechanisms, a key benefit of dFFT is that it does not impose any preconditions and can thus be automatically executed in oblivion of the underlying process.

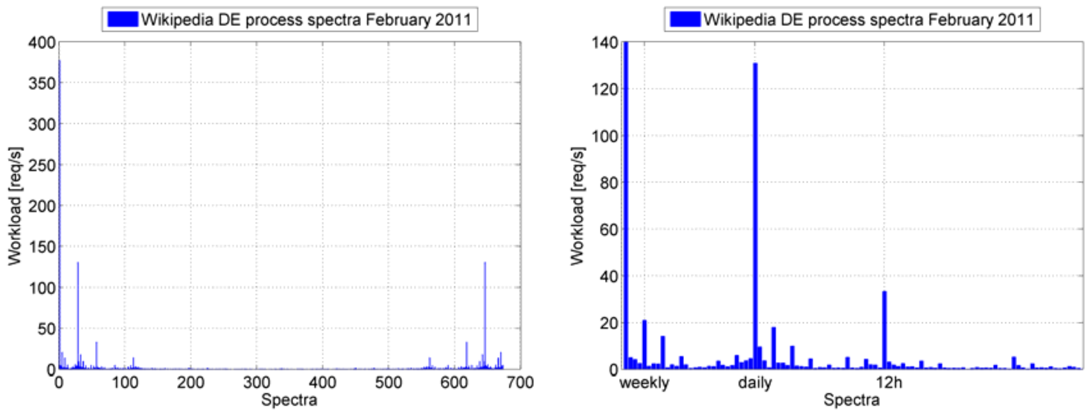


Figure 6: Wikipedia DE workload trace spectra

The spectra consist of n distinct frequencies, whereby most elements are close to zero. Figure 6 depicts the spectra of the Wikipedia DE workload trace from February 2011 (Anon n.d.). Evidently, the significant parts of the spectra are in the border regions. Spectra line 0 represents the mean of the process. As in the case of real number inputs, the FFT is mirrored at its middle, and we can see in the magnification that the daily effect has the most significant impact on the process.

The significant components of the spectra are identified using an iterative selection scheme. The heuristic starts with a zero-vector Q of length n and iteratively adds the most significant frequencies f_l (defined by the value of their amplitude r_l) (c.f. equation (8)). The selected elements are inserted at the same position as in their source vector F. After each iteration, it is tested whether the selected components of the spectra in Q sufficiently explain the original process W. This is reflected by the termination rule (9). By applying the inverse Fast Fourier Transformation iFFT (c.f. equation (10)) and calculating the residual sum of squares, it is tested whether the $(1-\alpha)$ percent of the process variation is explained.

$$\text{Selection Scheme: } q_l = f_l, l = \arg\max_{k=1,\dots,n}\{f_k | F \backslash Q\} \quad (8)$$

$$\text{Termination Rule } \|iFFT(Q) - W\| \geq \alpha\|W\| \quad (9)$$

$$w'_k = \sum_{l=0}^{N-1} e^{-2\pi i * \frac{lk}{n}} * q_l \quad (10)$$

With the termination of the selection scheme, the forecast dataset is obtained.

The workload forecast mechanism utilizes the representation of the spectra in trigonometric functions to extrapolate the process. We use the definition of iFFT and modify it to use it for extrapolation. First the original time series index k is replaced with $n + \tau$, where n represents the last historic data point and τ is the relative index of the period to predict. Subsequently the complex numbers are written in their sine and cosine representations. Afterwards the imaginary components are removed by applying the addition theorem for trigonometric functions.

$$p(\tau) = \sum_{q_j \in Q} \mathrm{r_l} * \cos\left(2\pi * \frac{\mathrm{l(n+\tau)}}{\mathrm{n}} + \phi_l\right) \quad (11)$$

$$v(\tau) = w_n * \prod_{\tau'=1}^{\tau} \frac{\Delta p(\tau')}{\Delta \tau'} \quad (12)$$

This leads to the forecast function $p(\tau)$ (c.f. equation (11)) which predicts the absolute expected workload level for the future period τ. Previous experiments on real workload traces have produced a better prediction quality if we do not estimate the absolute value but instead the dynamics of the process. By differentiating $p(\tau)$ with respect to τ we obtain the first derivative of the process. By multiplying the last observation w_n with the first derivative, we obtain the workload forecast function $v(\tau)$ (c.f. equation (12)). To forecast multiple periods in the future, the prediction of period τ is based on $v(\tau - 1)$. This leads to the final workload forecast vector $v =< v(1), \dots v(m) >$.

3.3. Converting Forecasts into Workload Matrix

As the Adaptation Engine provides a generic interface to support various workload forecast mechanisms, preprocessing is necessary to convert the forecasts into the format of the workload matrix. Most workload forecast mechanisms in literature usually deliver point estimates for the workload in the near future. However, it is better to use distribution estimates in order to facilitate more sensitive operation strategies. Thus, preprocessing requires the derivation of an empirical prediction error distribution function. Subsequently, the distribution function needs to be converted into matrix W.

$$e_\tau(t) = w_{t+\tau} - v_t(\tau) \quad (13)$$

$$F_\tau^t(w) = \frac{|\{e_\tau(t) < w - v_t(\tau) | t \in T\}|}{|T|} \quad (14)$$

The prediction error function $e_\tau(t)$ provides the error of the prediction in t for the period $t+\tau$ between the forecast and the workload (c.f. equation (13)). In order to obtain the prediction error distribution for the forecast period τ, the empirical distribution function is defined as in equation (14).

$$P_\tau^t(w_l < x_\tau \leq w_u) = F_\tau^t(w_u) - F_\tau^t(w_l) \quad (15)$$

$$W_t = \begin{array}{c} \\ w_1 \\ \vdots \\ w_n \end{array} \begin{array}{ccccc} \tau = 0 & \tau = 1 & \dots & \tau = m \\ \end{array} \left(\begin{array}{cccc} 0 & P_1^t(x < w_1) & \cdots & P_m^t(x < w_1) \\ 1 \; if \; w_i < v_t(0) < w_{i+1} & \vdots & P_\tau^t(w_i < x < w_{i+1}) \ddots & \vdots \\ 0 & P_1^t(w_n > x) & \cdots & P_m^t(w_n > x) \end{array} \right) \quad (16)$$

Based on the definition of $F_\tau^t(w)$, we can now define the probability function to face a certain workload in the next period P_τ^t (c.f. equation (15)). This allows the definition of the final workload forecast matrix. The columns define the forecast for the next m periods, whereas the rows specify the probability (refer to equation 16).

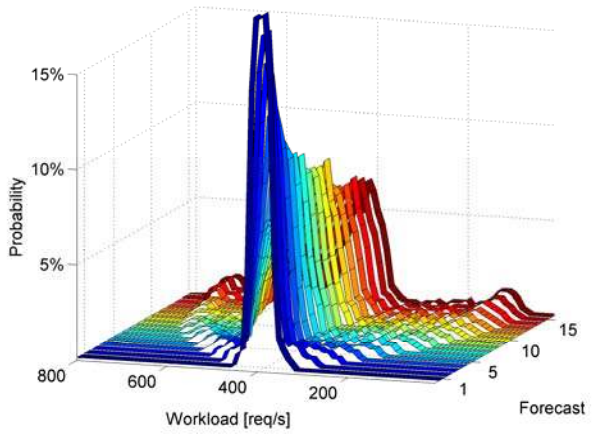


Figure 7: Workload forecast matrix

Figure 7 shows a workload forecast matrix generated on a Wikipedia DE workload trace. The workload level is given on the X-axis, whereas the Y-axis depicts the probability of facing a certain workload level in the next periods. The Z-axis indicates the forecasting period. In the figure, the workload forecast model was set to predict the next 16 periods, whereas the period length was set to 15 minutes. In the resulting four hour forecast, the near future period τ_1

produces a relatively sharp estimate. With increasing time distance, however, the forecasting error increases.

3.4. Performance Model

Elastic applications are one solution towards highly efficient operation modes of large enterprise systems. Nevertheless, in order to provision the optimal hardware configuration for any given workload, a thorough understanding of the system characteristics is required. The system characteristic is determined by the hardware, software and the system state. Furthermore, not only the workload level, but also the workload mix (i.e., share of each transaction type in the workload process) has a strong influence on the overall system performance. Usually elastic applications do not scale linearly with the amount of hardware resources. In particular n-tier systems usually show complex performance characteristics, which are caused by various, non-trivial interdependencies within and between the different layers of the system. Modeling of large, distributed systems is a very active research topic and various concepts and models have been developed to describe and predict the behavior of these systems at run-time. For instance (Urgaonkar et al. 2008) used queuing models for automated resource allocation in multitier applications. In (Cohen et al. 2004) the authors used machine learning to model the performance characteristic. The focus and methodology of the single approaches differ strongly. Nonetheless, to enable their application in enterprise environments, two design criteria are of particular importance. First, the system model should be an autonomic component, as manual interaction would be too slow, compromising system stability in situations of sharp workload shifts; second, it should not require high domain knowledge.

In this research, we pursue an observation based, empirical approach for the performance modeling of large systems. In contrast to other models, our approach solely relies on the observed system behavior during operation. More specifically, our system characteristics model monitors the workload process, system metrics, as well as the SLO relevant metrics (e.g. response time) and saves the data in an operational data store. The model is not only designed to process different workload levels but also different workload mixes. In this article, we keep the empirical model as simple as possible, as it is needed in our evaluation.

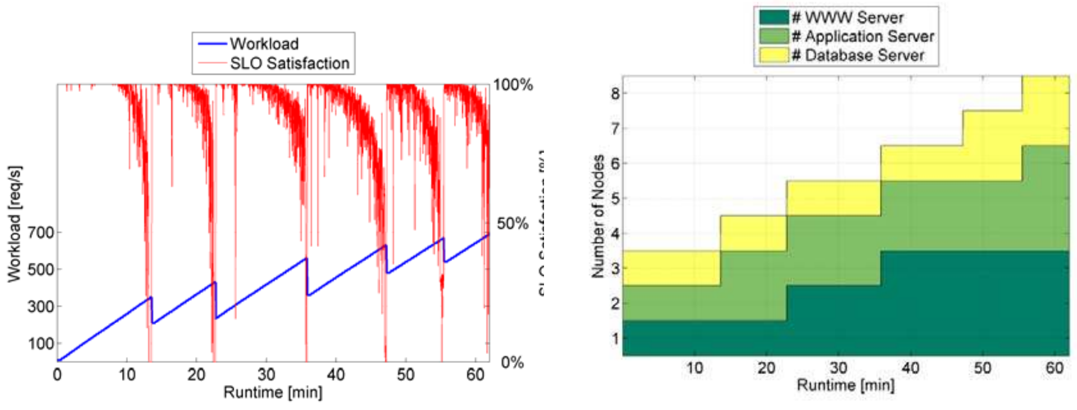


Figure 8: System characteristics analysis process

The system characteristics model is distinguished into two phases, training and operation phase. In the training phase, the model analyzes the performance behavior by iteratively scaling the system. The performance analyzer starts with the smallest hardware configuration (i.e. in our example with one web server, one application server, and one database server) and continuously increases the workload. The left panel of 8 shows how the number of a workload is increased over time. During the training, the system performance is continuously monitored, as depicted as the black line in the left panel representing the SLO compliance, defined as the percentage of requests with a response time below 200ms in monitoring intervals of one second. As soon as the system performance significantly drops (i.e. response time increases rapidly), the maximum workload level for the configuration is determined. In our example the SLA compliance drops below 95 % at a workload level of 300 requests per second. Usually, the reason for the performance drop and associated SLA violation is a bottleneck resource (e.g. CPU, memory). To find out which resources are representing the bottleneck, the performance analyzer utilizes a horizontal scale controller. The scale controller monitors all system resources and detects saturated resources. By adding an additional resource unit to the specific layer, the maximum workload level which the system is capable of sustaining can be elevated. The right panel of Figure 8 shows the used configuration of the resources at a certain point of time. Apparently, up to a workload of 300 requests per second, which is reached after 15 minutes, the configuration is one web-server, one application server and one database server. An increased maximum load level is attained by adding one additional application server. The performance layer terminates on reaching a predefined system size. As a result of the training phase, a dataset is obtained, which reveals the maximum workload levels for specific infrastructure, application, and system states.

In the operation phase, the system model feeds the Adaptation Engine with the performance characteristics of the system. Essentially, he Adaptation Engine queries the system model for performance data of specific workload levels and configurations. The operation data store (ODS) is continuously updated with the most recent monitoring data. The advantage of this feedback is twofold: firstly the feedback contributes to a larger dataset, which allows more fine-grained provisioning decisions; secondly, the feedback loop facilitates that the system model continuously adapts to the system characteristics. This is important when the system state (e.g. software release) changes over time and leads to a different system behavior. The system model may eventually incorporate these changes.

3.5. *Converting System Characteristics into the Performance Matrix*

Similar to the workload model, the Adaptation Engine does not directly use the system characteristics model, but instead converts it to its internal system performance model. As part of the conversion process, the SLO compliance of the different configurations and workload is determined.

$$p_{c,w} = \frac{1}{|R_{c,w}|} \sum_{r \in R_{c,w}} \begin{cases} 1 & RT(r) < \Theta \\ 0 & else \end{cases} \quad (17)$$

$$P = \begin{array}{c} \\ c_1 \\ \vdots \\ c_k \end{array} \begin{array}{c} \begin{array}{ccc} w_1 & \dots & w_m \end{array} \\ \begin{pmatrix} p_{c_1 w_1} & \cdots & p_{c_m w_m} \\ \vdots & \ddots & \vdots \\ p_{c_k w_1} & \cdots & p_{c_k w_m} \end{pmatrix} \end{array} \quad (18)$$

We define $R_{c,w}$ as the set of all requests r observed on configuration c with a workload level w. Furthermore we define Θ as the response time threshold. Therefore all requests with a response time smaller than Θ comply with the SLO. The resulting value $p_{c,w}$ represents the average SLO compliance of the distinct configuration and workload (17). Based on the SLO compliance, we can now derive the final system performance matrix, which describes the expected performance behavior of the system under different workload levels and configurations. In order to incorporate the most recent observation, the performance matrix needs to be updated frequently during operation. The final system performance matrix *P* contains the average degree of SLO compliance for the different configurations and workload levels (18).

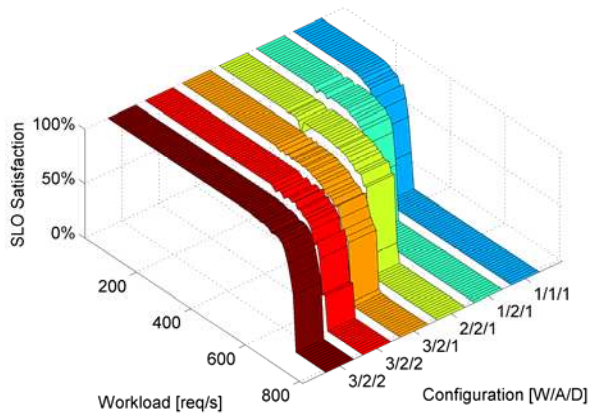


Figure 9: System performance model

Figure 9 shows the 3D-plot of the system performance matrix generated on the base of the previous performance analysis with the SLO target Θ set to 200 milliseconds. Essentially, we assume a monotonic performance behavior (i.e., if we add a resource to a layer, we expect performance to not be negatively affected). Therefore, we define the SLO compliance for all workload levels as 100%, whereas all workload levels exceeding the level of significant SLO violation are set to 0%. The figure clearly shows the growing system performance with the increasing number of resources. Furthermore the figure indicates that the individual performance curves for each configuration can be divided into three sectors. The first sector includes the workload range which reaches a 100% SLO compliance. The second sector comprises the workload levels, in which the SLO compliance decreases. The last sector contains all workload levels which significantly violated the SLO and hence are infeasible during operation. While the first sector is usually inefficient during operation as the system is over provisioned (i.e., a smaller system configuration may also be able to achieve the same SLO compliance), the last sector is most of the time very critical. If the system remains too long in an overloaded state, the system will probably become instable and crash. Hence, during operation it is crucial to prevent the system from entering work levels of the third sector. With the presentation of the performance model, we conclude the inputs of the decision model.

3.6. *Operation Strategy Functions*

In the previous sections, we presented the derivation of the final decision matrix D, which aggregates the workload forecast, system characteristics and the SLA. In order to manage an enterprise information system with the Adaptation Engine, we need to evaluate the decision matrix.

The operation strategy function determines an optimal system configuration for the next period based on the anticipated system behavior and the SLA. We define the operation strategy function $S(D, c^*, \alpha, \Delta)$ that decides upon the configuration for the next period. D is the decision matrix, c* the current configuration, α the SLO target, and Δ the lead time. Depending on the definition of the strategy function S, we can identify three categories of strategies. The first category involves a static system operation mode, keeping the hardware configuration constant. The second category comprises of reactive operation modes that reconfigure the hardware according to actual observations (basic reactive, and reactive with performance reserves). The third category comprehends operation modes that utilize forecasting information in addition to actual observations (predictive, lifetime optimization, expected profit).

3.6.1. Static system operation mode

The most basic operation strategy is a static system operation mode that keeps the hardware configuration unchanged over the whole SLA lifetime (19).

$$c \leftarrow S^{static}(\emptyset) \quad (19)$$

The static system operation mode reflects a classical operation strategy that is naturally extremely costly, as the system needs to be adjusted to the peak workload. In the later evaluation, we will use the static operation mode as baseline.

3.6.2. Basic reactive operation mode

The basic reactive operation mode follows an observation-based operation strategy. The basic intuition is that the system configuration will be scaled up if the current configuration

c^* cannot satisfy the SLO at a desired target probability of α. This information can be obtained from Decision matrix D at element $0c^*$. The system will be scaled down if the *next best* to the currently used configuration satisfies the SLA at a probability that exceeds or is equal to the target α. Suppose the provider has set the target probability α to 95%. The current configuration c^* consisting of 3 web servers, 3 application servers and 2 database servers succeeds in meeting the target probability. However, the *next best* to the currently used configuration $c^* - 1$, consisting of 3 web server, 2 application servers and 2 database servers also guarantees the target probability level. The strategy of the basic reactive operation mode advises to scale down the current system, which is over-dimensioned. In all other cases, the basic reactive operation mode leaves the current configuration untouched.

$$S^{reactive}(D_0, c^*) = \begin{cases} D_{0c^*} < \alpha & \text{scale-up} \\ D_{0(c^*-1)} \geq \alpha & \text{scale-down} \\ else & \emptyset \end{cases} \quad (20)$$

Note that the system may have a certain lead time to adjust the system to the desired configuration. Scaling down the system will take place immediately, as removing resources from the system require only informing the load-balancers of the elastic application. Scaling-up actions entail a lead time, as the resources need to be integrated into the running systems. The lead time can be substantial if large databases need to be synchronized. In the later evaluation, we will use the reactive operation mode as an additional baseline to argument the benefits of the Adaptation Engine instantiated with workload forecast and dynamic SLA model.

3.6.3. Reactive operation mode with performance reserve

The reactive operation mode with performance reserves adheres to the intuition of the basic reactive operation mode but implements a performance reserve. Apparently, the operation mode does not select that system configuration c^* that meets the target probability level α. Instead, the further advanced configuration c^*+1 is chosen.

$$S^{rr}(D_0, c^*) = S^{reactive} + 1 \quad (21)$$

For example while the configuration 3 web server, 2 application servers and 2 database servers may sustain the target probability level of 95% to satisfy the SLA, the reactive operation mode with performance reserves selects the next best configuration consisting of 3 web server, 3 application servers and 2 database servers. Clearly, this strategy is more defensive as it increases the minimal configuration c^* by the next best advancement. This reserve built into the system assures that small prediction errors can be absorbed without violating the SLA. The basic reactive controller might not be implemented as it has the inherent risk of SLA violations. By adding additional performance reserves this limitation can be compensated, which might be a standard industry solution. In the evaluation, we will use this strategy to benchmark the effectiveness of the complex Adaptation Engine instantiation.

3.6.4. Predictive operation mode

While the reactive operation modes base configuration decisions upon the current situation of the system, the predictive operation mode uses the workload forecasts to make the decision. The intuition behind the scaling actions is the same as for the basic reactive operation mode. Scaling up becomes necessary if the current configuration c^* cannot meet the SLA in the next future (i.e. from time t to t + Δ), during which the configuration is fixed due to the lead time. In those cases it is necessary to increase the system configuration, such that the advanced configuration at time t + Δ is available to satisfy the SLA accordingly. Apparently, the predictive operation mode uses not only the first column of the Decision matrix but later columns that refer to the lead time.

$$S^P(D,\alpha,c^*) = \begin{cases} D_{c^*\Delta} \leq \alpha & \text{scale-up} \\ D_{(c^*-1)\mathrm{d}} \geq \alpha, \forall_{d=0,\dots,\Delta} & \text{scale-down} \\ else & \emptyset \end{cases} \quad (22)$$

Having in mind that downscaling takes place immediately, scaling down is more complicated, as the predictive operation mode needs to verify that the downscaled system with configuration c^*- 1 satisfies the SLA for all periods to come until t + Δ, and not only the last period.

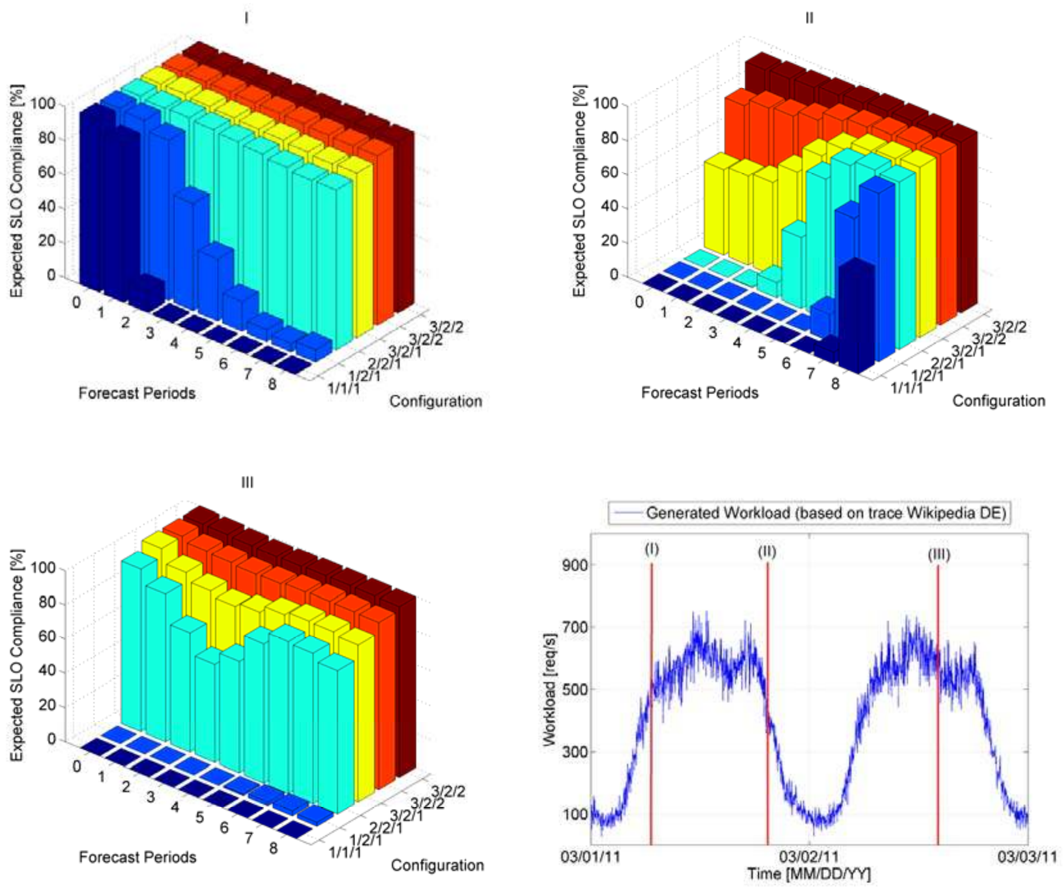


Figure 10: Decision matrix at different points in time

To emphasize the functionality of the predictive operation mode, Figure 10 shows three states of the decision matrix . In figure (I), the system is configured to the smallest configuration. Though this configuration still satisfies the current user demand, the system needs to scale up. Depending on the delay time, the Adaptation Engine will switch to the configuration (1/2/1) or, in case of a longer delay time, directly to (2/2/1). The second figure shows a scale down scenario. Depending on the lead time and the current configuration, the figure (III) shows multiple scenarios. If the system is in configuration (2/2/1) and the lead time is short, the system needs to scale up. If the system is in configuration (3/2/2) and the lead time is long, the system size cannot be reduced, as the stronger configuration will be necessary again in the near future.

The predictive controller aims to provide an efficient configuration at any point in time. In the usual operation (i.e., the workload process can be accurately forecasted) this allows managing the system with very high resource efficiency.

3.6.5. Lifetime optimization operation mode

Let α be the target compliance defined in the SLA, β be the status of current SLA compliance, and γ be the required SLA compliance until the end of the SLA lifetime to meet the SLA. Furthermore, we define δ as the percentage of the elapsed time of the SLA lifetime.

$$\alpha = \delta\beta + (1-\delta)\gamma \Leftrightarrow \gamma = \frac{\alpha - \delta\beta}{1-\delta} \qquad (23)$$

The target probability of the SLA must be equal to the weighted current SLA compliance and the SLA compliance in the remaining lifetime of the SLA. Solving this equation for γ allows the calculation of the required SLA compliance for the remaining SLA lifetime. We can now redefine our predictive operation mode to the lifetime optimizing operation mode as follows

$$S^{LO}(D,\gamma,c^*) = \begin{cases} D_{c^*\Delta} \leq \gamma & \text{scale-up} \\ D_{(c^*-1)\mathrm{d}} \geq \gamma, \forall_{d=0,\ldots,\Delta} & \text{scale-down} \\ else & \emptyset \end{cases} \qquad (24)$$

3.6.6. Lifetime optimization operation mode with long term forecast

The lifetime optimization operation mode accounts for all periods up to the delay time for reconfiguration. The lifetime optimization operation mode with long term forecasts also utilizes workload forecasts until the end of the SLA. For the formal definition of this advanced operation mode, we need to define the whole time series of the SLA life time as $t = t_1, \ldots, t_k \in T$. Subsequently, we use the workload forecast model to derive the long-term forecast. Based on the absolute prediction function $p(\tau)$, we can derive the long-term forecast time series $W^{lt} =< p(1), \ldots, p(k-l) >$.

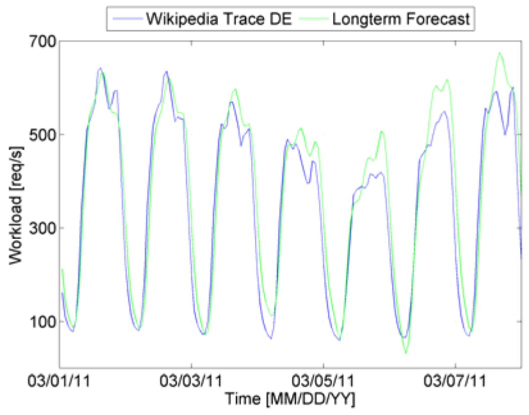


Figure 11: Long term forecast

Figure 11 shows the result of the long-term forecast function for a whole week on the Wikipedia DE trace. Obviously, at the end of the week, the accuracy decreases, but still captures the main characteristics of the process. Using the required SLA compliance until the end of the SLA lifetime to meet the SLA, γ allows us to formulate the optimization problem

$$\min_{\mathrm{c}_{\mathrm{l}+\Delta},\dots,\mathrm{c}_{\mathrm{k}}} c = \sum_{t=t_{l+\Delta}}^{t_k} c_t \qquad (25)$$

$$\text{subjected to} \sum_{t=t_{l+\Delta}}^{t_k} SLO(c_t, w_t^{lt}) \geq \gamma$$

Apparently, we minimize the sum of all resources over the whole remaining lifetime of the SLA as the objective function, so that the sum of all SLO exceeds the required SLA compliance until the end of the SLA lifetime.

$$\gamma' = SLO\left(c_{l+\Delta}, w_{l+\Delta}^{lt}\right) \quad (26)$$

$$S^{LO}(D,\gamma',c^*) = \begin{cases} D_{c^*\Delta} \leq \gamma' & \text{scale-up} \\ D_{(c^*-1)\mathrm{d}} \geq \gamma', \forall_{d=0,\dots,\Delta} & \text{scale-down} \\ else & \emptyset \end{cases} \quad (27)$$

Having solved this simple linear optimization problem, we determine the planned SLO compliance level γ' for the next operation period. Based on this new target value, we update the previous operation mode S^{LO} to determine the actual configuration for the next provisioning period. Essentially, S^{LO} will return the same configuration decision as the optimization problem for $c_{l+\Delta}$. However, as the long term forecast might be inaccurate, the update of γ' mitigates the risk of SLA violations, as the provisioning decision is based on the accurate decision matrix D.

4. Evaluation

Having in mind that the aim of this paper is the design of a practicable green operation model – the adaptation engine framework – that optimizes IT system operation based on a holistic evaluation of all key aspects of the business value chain, we need to confront our model with reality and validate how the adaptation engine can optimize the IT system operation. Essentially, our evaluation will address the following four aspects that underlie the design problem statement:

- **Practicability:** Can the adaptation engine framework be implemented for current IT systems?
- **Optimization of IT system operation:** Under what conditions can the adaptation engine optimize the IT system operation?
- **Effective Green IS:** Can the ecological footprint be reduced by relying on the adaptation engine?
- **Business value chain orientation:** Can the adaptation engine integrate the business value chain in the IT system operation?

Practicability of the adaptation engine framework is guaranteed by a fully-fledged implementation as proof-of-concept. The prototype will be used to create a test environment that forms the basis for the validation of the three aspects empirically.

4.1. Evaluation Environment

The evaluation of our design artifact - the Adaptation Engine - is facilitated by the fact that we implemented the adaptation engine and integrated it into a test infrastructure. The intuition behind our cumbersome prototype building process stems from the insight that the high complexity of the enterprise system environments does not allow demonstrating the validity solely on data analysis alone. Instead, the validity must be proven empirically by collecting the data from a running installation of the system. The test system we used for evaluation purposes consists of an elastic benchmark application, an experiment controller, a basic cloud environment and the implemented Adaptation Engine.

The benchmark application is an updated implementation of the RUBBoS Benchmark (http://jmob.ow2.org). This benchmark refers to an n-tier e-commerce application modeled on a bulletin board news sites similar to Slashdot (http://slashdot.org/). The original implementation consists of 24 different interactions involving all tiers such as register user, view story, and post comments. We ported the Java implementation of RUBBoS to the Microsoft .NET environment and modified the benchmark to run six native RUBBoS interactions. The .NET code was deployed in an enterprise class Microsoft environment. The frontend and the application server tier were deployed on Microsoft Internet Information Services 6.0 (IIS) servers. As a backend, Microsoft SQL Server 2005 was used with the schema and large dataset provided by the RUBBoS benchmark. We modified the RUBBoS benchmark workload generator to use an open model with a freely manageable number of concurrent user interactions (i.e., no client thread sleep time). Furthermore, we implemented a generative workload model. Based on an input time series, the workload generator continuously changes the workload level. With the help of spline interpolates, additional data points are generated in order to simulate a continuous workload process. This is necessary as the later evaluated Wikipedia traces only have a resolution of one hour. By adjusting the number of interpolates, the run-time can be adjusted. This allows the evaluation of the Adaption Engine above real-time. The interactions are chosen based on an a priori specified probability vector. All benchmark application components were deployed on Microsoft Windows Server 2003 (SP2). A key modification of our RUBBoS implementation is its capability for online reconfiguration. Revising the load balancers facilitates the adding and removing of resources (i.e. virtual servers) from each tier at run-time.

We used a custom local cluster as our cloud infrastructure. We implemented scripts that allow us to obtain and release virtual resources via web services similar to a real IaaS cloud environment. The test cluster was built using hardware virtualized Xen 3.4.3 VM instances. The underlying operating system was Red Hat Enterprise Linux 5 (2.6.18-194.11.1.el5xen kernel) on 12 hardware nodes with 8GB of memory and an Intel Core2 QuadCPU (Q9650) with 3.00GHz each. All VMs were created with one VCPU and 2GB of memory into a pool of idle resources and ramped up upon request.

In order to provide maximal compatibility to the elastic application testbed, the Adaptation Engine prototype is implemented based on a similar software stack as the testbed. Most of the prototype code was written in the Microsoft .NET environment, whereas most mathematical operations (e.g. workload forecast model) are written in Matlab. The different modules of the Adaptation Engine are implemented as web services with an administrative frontend.

4.1.1. Optimization of IT system operation

The adaptation engine framework offers a configuration toolbox for using different operation modes for the resource management. The service provider can use those operation strategy functions according her needs. To support the service provider with her choice, the operation strategies are analyzed with respect to the relevant factors of influence. The relevant factors of influence that matter in this respect are (i) the system characteristics and in particular the lead time that is required to reconfigure the system and (ii) the user behavior that may exhibit workload processes with different degrees of variability. Both factors determine how accurate the different operation strategy functions can optimize the IT system operation. As the operation strategy functions address different aspects, it is necessary to compare operation modes that share the same functionality. Accordingly, we will evaluate the basic reactive and the predictive operation modes on the basis of the lead time and workload process variability. The lead time can be directly controlled in our elastic application, whereas different synthetic workload processes can be fed into the test environment.

A synthetic workload process constitutes a time series composed out of five seasonal components with different frequencies and intensities. The Adaptation Engine executes the operation strategies at each point of the series, whereas the workload generator interpolates additional data points to simulate the users of the system. The workload forecast model is calibrated on a set of 1000 data points and has a ramp up of 200 iterations before the evaluation.

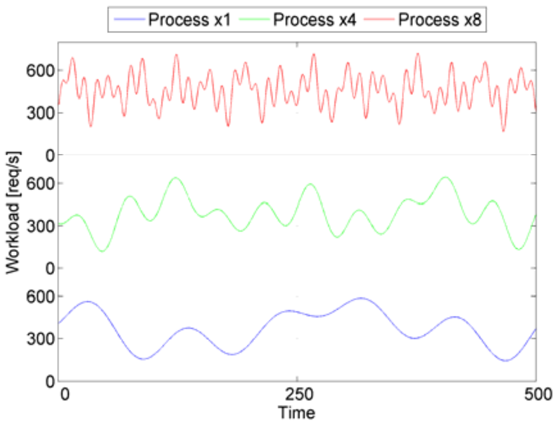
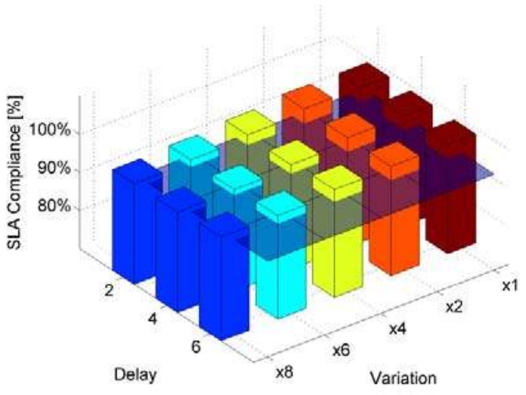


Figure 12: Synthetic workload process

In order to assess the performance of different operation modes exposed to different workload volatilities, we derived four additional processes on the base of the original process, whereas the periodicity of the process is increased by the factors 2, 4, 6 and 8. Figure 12 shows three sample input workload processes, where x1 is the original process, x4 represents the process with periodicity of 4, and x8 with a periodicity of 8 that we use in our evaluation. We assume that the SLO requires a system response time of below 200 ms for 95% of the requests within each second. In addition, we assume that the Service Level Agreement demands compliance with the SLO objective in 95% of the SLA lifetime, which is set to 500 periods ($t_1, \dots, t_{500}$).



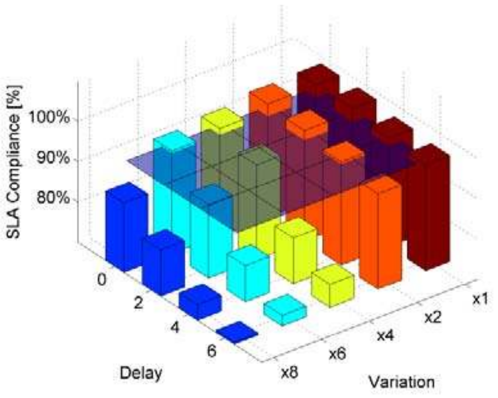


Figure 13: Predictive operation mode **Figure 14: Reactive operation mode**

Figure 13 and 14 present the resulting SLA compliances of 35 different system runs. Each system run uses a different lead time (delay) and process periodicity (variation). The left panel shows the performance of the predictive operation mode, whereas the right panel shows the performance of the reactive mode. X and Y-axes define the different settings for the reconfiguration delay and process periodicity. The shaded plane at the 95 % SLA compliance visualizes the target of the SLA – all scenarios below this plane violate the SLA; equal and above comply with the SLA. Evidently, the predictive operation mode succeeds in complying with the SLA in all scenarios, which we can summarize in result 1:

***Result 1:** The predictive operation mode succeeds well in complying with the SLA when facing a workload process with strong seasonal components.*

The assessment of the basic reactive mode is not as positive as it violates the agreement in half of the scenarios. Essentially, the reactive mode fails in the case of (i) high process variation, which implies the need of frequent reconfiguration, as well as in the case of (ii) long reconfiguration lead times.

Table 1: Variation of processes with lag d

$Q_{0.95}(\nabla)$	d=0	d=2	d=4	d=6
x1	0	20.0	39.9	79,2
x2	0	37,7	74,9	146,0
x4	0	72,9	141,7	253,6
x6	0	96.8	185.3	306.2
x8	0	143.0	256.6	317.8

For a more detailed analysis, we analyze the volatility of the processes in dependence of the reconfiguration lead time lag. By defining $\nabla := \{|w_t - w_{t+d}| \forall t = t_1, \dots, t_{500-d}\}$, we can determine the 95% quantile of the process volatility during the lead time, which corresponds to our SLA definition. Table 1 shows the resulting 95% quantiles; the background of the table is shaded in case of SLA compliance. In this analysis the quantile states that in 5% of the periods between the initiation of the reconfiguration and its availability, the workload level changes by more than the quantile value. Evidently, as soon as the difference exceeds 70 requests per second, the risk of SLA violation increase significantly. Consequently, we have defined a metric that measures the volatility of the workload process. In addition, we can derive a threshold value in terms of requests per second that identifies when basic reactive operation modes are feasible.

Table 2: System characteristics data

	1/1/1=3	1/2/1=4	2/2/1=5	3/2/1=6	3/2/2=7	3/3/2=8
Max. number of users	290	400	470	530	590	660
Additional number of users (Marginal performance gain)		110	70	60	60	70

In order to correlate this result with the system characteristics, we take a closer look on the performance model. In the given SLA setting, the smallest configuration can manage up to 290 requests per second without the risk of an SLA violation. While the first server increases this maximum level by 110 requests per second, each additional server only provides additional performance for up to 60 - 70 users. Based on these numbers, we can derive the second result of our evaluation:

Result 2: *If the workload process variation exceeds the marginal performance gain of an additional server, the basic reactive mode fails to manage the system according to the SLA.*

Result 2 is instructive as it reveals under which circumstances the basic reactive mode can be used to optimize IT system operation. Overall, results 1 and 2 shed light into the question of under what conditions different operation modes can optimize IT system operation.

4.2. *Effective Green IS*

Having demonstrated that the adaptation engine can optimize the IT system operation, we can now tackle the question of whether it can also be effective in reducing the ecological footprint. In this part, the focus is on how many resource savings can be realized by resource adaptive operation modes without compromising the overall system performance. In the absence of disruptions in the workload process (reflecting flash crowd effects) and the performance model (caused by sudden hardware failures or outages), the predictive operation mode and the lifetime optimization operation mode fall together. Thus, this section abstracts from those disruptions – a concise treatment of scenarios with disruption is given in the next subsection. As a consequence, we can compare the static system operation mode, the basic reactive with and without performance reserves and the predictive operation mode.

Different than the previous analysis, we now apply the Adaptation Engine on a real-world workload process. Accordingly, we use the trace of Wikipedia DE (German version) which has a lower base load compared to the largest Wikipedia EN. The trace is characterized by an hourly resolution. Adapting the trace to our test environment, the workload process was scaled to 80% and used the workload generator to interpolate data points. SLA lifetime is set to one week; the adaptation engine's execution frequency is defined to be 15 minutes. During real time execution, this results in more than 600.000 monitoring data points. To reduce the run-time, the resolution of the generated process was reduced to 5376 data points.

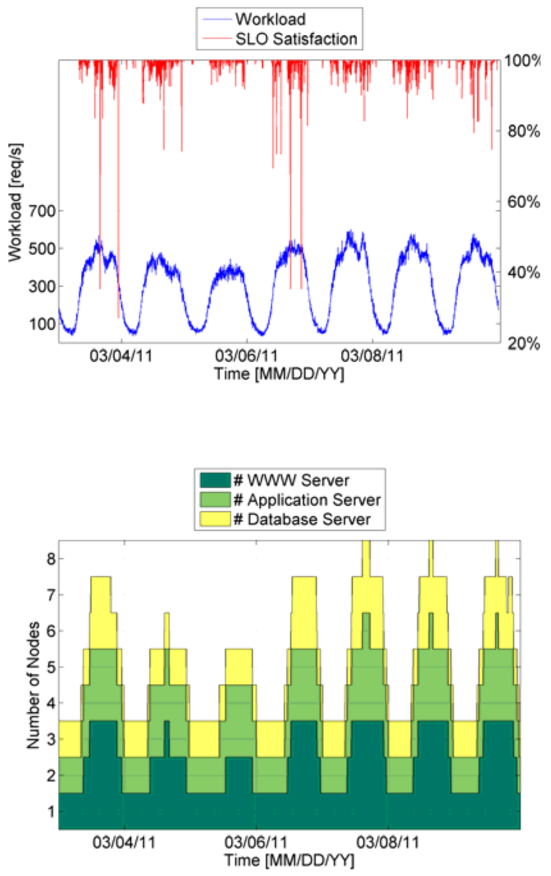


Figure 15: Predictive operation mode

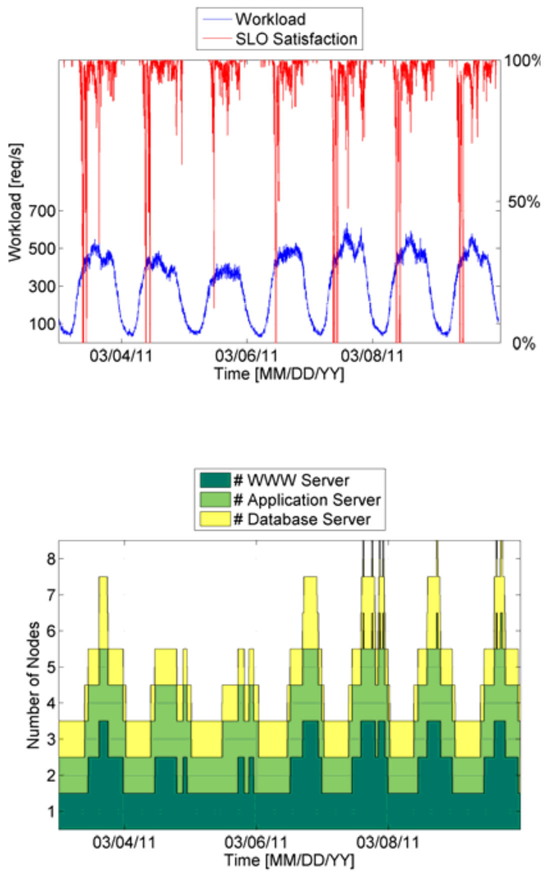


Figure 16: Reactive operation mode

Figures 15 and 16 exemplify the results of a single experiment series for the predictive (left) and basic reactive (right) operation mode. The upper graphs depict the workload process and the SLO compliance whereas the lower graphs show the hardware configuration. The reconfiguration lead time is set to one hour. While the predictive operation mode achieves an SLA compliance of 99.1%, the reactive strategy only achieves a compliance of 94.4%. Having defined the SLA of 95 % the basic reactive mode has violated the SLA. During off peak times, both operation modes achieve an SLO compliance of 100% (i.e. all incoming requests are processed within 200 ms), which results from the fact that the minimal configuration can process up to 290 concurrent requests. During peak times the predictive operation strategy continuously aims at selecting hardware configurations to achieve 95% compliance. As a consequence, during the first four days, the predictive strategy never switches to the strongest system configurations, as the smaller are already sufficient to comply with the SLO. Evidently, during the peak time several SLO compliance monitoring points are below the

95% threshold – refer to the red line in the upper graphs. Essentially, the SLO satisfaction drops considerably when the workload transforms from off peak to peak times. Nevertheless, the predictive operation mode selects configurations, which comply with the SLO on average during each period (15 minute intervals), and hence operates within the defined boundaries of the SLA. Compared to the basic reactive mode, the predictive mode employs less reconfiguration operation, which is a result of the look ahead and thus more sensitive scale-down operation of the predictive operation mode. We can summarize this in result 3:

***Result 3:** The predictive operation mode employs less reconfiguration operations than the basic reactive operation mode.*

Result 3 also implies that the predictive operation mode incurs less of the switching costs that are needed for reconfiguring the hardware. Having described the setup of the experiment series, we can now apply the procedure to different operation modes with different delay time. Table 3 shows the results of different operation strategies and operation modes. Naturally, the static operation mode with the largest hardware configuration achieves 100% SLA compliance, however, at the price of very high resource consumption. We define the static operation mode as our benchmark and compare the different modes in terms of the static operation mode.

Table 3: Results of different operation strategies and reconfiguration lead times

	Delay	Resource Demand	Switching Cost	SLA Compliance	Cost Reduction	Reduction of variable Resources	System in Critical Stage
Unit:	*[minutes]*	*[instance hours]*		*[%]*			
Static		1343	0	100,0%	0%	0%	0,0%
Reactive	0	732	0	98,2%	46%	72%	0,0%
	15	740	8,3	98,1%	45%	73%	0,0%
	30	748	16,5	97,4%	44%	71%	0,2%
	60	765	33	94,4%	43%	69%	2,5%
	120	796	66	87,8%	41%	65%	8,7%
Reactive with Performance Reserve	0	897	0	99,2%	33%	52%	0,0%
	15	904	6,75	99,2%	33%	53%	0,0%
	30	911	13,5	99,1%	32%	52%	0,0%
	60	924	27	99,0%	31%	50%	0,0%
	120	950	54	97,6%	29%	47%	0,5%
Predictive	15	817	8,3	99,3%	39%	63%	0,0%
	30	825	16	99,2%	39%	62%	0,0%
	60	835	29	99,1%	38%	61%	0,0%
	120	868	58	99,2%	35%	57%	0,0%

The basic reactive operation strategy reduces the cost of operation on average by 40%. Considering that 37.5% are continuously required in the base line configuration, the potential of the reactive operation mode is even more evident if we evaluate the saving potential solely on the base of the variable resources. Considerably more than 50% of the variable resources can be saved. Nevertheless, the saving potential decreases with the increasing lead time, as the ramp up phase consumes significant parts of the resource savings. Altogether the basic reactive mode is best in reducing the resources needed to comply with the SLA, which is our result 4.

***Result 4:** The basic reactive mode is the best green operation mode, reducing the operation costs most, which is a proxy for the ecological footprint*

However, the basic reactive operation mode is only feasible for short reconfiguration delay times. In scenarios with higher lead-time, the risk of SLA violation becomes more irrelevant as the system is several times completely overloaded, which leads to a system failure. During this critical state, the SLO compliance drops to zero percent and the system denies the service. The implementation of this strategy for higher reconfiguration delays is thus infeasible.

***Result 5:** The basic reactive operation mode becomes infeasible when facing long delays before reconfiguration actions take place.*

Result 5 suggests that the basic reactive mode cannot be used for elastic applications that use many writing operations on the database. The synchronization of a large database renders the usage of basic reactive modes useless.

The extension of the reactive mode with the performance reserves increases the SLA compliance and prevents the system from entering a critical overload state. Nonetheless, in comparison with the predictive operation mode, the reactive mode with performance reserves cannot attain better SLA compliance. Nevertheless, the operation cost is significantly higher for the reactive mode with performance reserves than the predictive operation mode.

***Result 6:** The predictive operation mode dominates the reactive operation mode with performance reserves in terms of operation costs.*

Essentially, results 4-6 suggest that the reactive operation mode should be used whenever the lead time of reconfiguration actions is very short. If the application is write-intensive, the predictive operation mode is more appropriate.

From the green perspective, the resource adaptive operation allows to shut-down the non-productive servers. Implementing such a policy would directly reduce the carbon footprint of the system by the same magnitude. From the economic perspective and assuming the total cost of operating a server is $1.00 per hour (cost of a small Amazon EC2 instance per hour), the Adaptation engine allows cost reductions of up to $4000 per year already for a small infrastructure with 8 servers.

4.3. Business value chain orientation

Hitherto, our analysis assumed that the operation is free of disturbances and disruptions stemming from the user behavior or from the IT system. This assumption implies that the predictions based on the workload forecast model are very accurate. However, the assumptions may not always hold as sudden spikes in the workload can occur all the time. More prevalent are disturbances in the performance model. If the adaptation engine employs the optimal green configuration, say one web server, three application servers and one database server, but one application server is not properly configured, then detrimental impact on the system performance cannot be avoided.

In this subsection we devote our attention to the question whether a lifetime optimization operation mode can cope with situations where disruptions in the workload of the performance model occur. We hypothesize that the use of our lifetime optimization operation mode allows for the alignment of IT system operation with the business value chain. The lifetime optimization operation mode epitomizes a dynamic SLA operation strategy that aims at satisfying the SLA and fulfills simultaneously its Green IT and Green IS objectives.

In order to show the impact on the system management more clearly, we employ a less stringent SLA that offers flexibility in its execution. Accordingly, the SLO target was set to a value of 90% and the SLA requires 90% compliance to the SLO. The SLA lifetime was set to one week and the reconfiguration lead time is set to one hour. The execution interval of the adaptation engine is set to one hour.

As we are particularly interested in the analysis of the previously mentioned disruptions, we distinguish two different scenarios. The first scenario is dubbed ‘normal scenario’ and is equal to the ones we have discussed in the previous subsections without disruptions; the second is denoted as ‘failure scenario’, which includes disruptions.

The failure scenario temporarily reduces the system performance, which effects a decrease in the SLO compliance rate of 60% . This scenario intends to show how dynamic SLA management not only helps to operate enterprise IT systems more efficiently, but also helps reacting to changes in the environment, mitigating the risk of SLA violations.

Generally, significant SLO violations may ultimately lead to a violation of the SLA. In our failure scenario, we assume that the system is continuously online and still capable of serving all users waiting in a queue, albeit with a lower response time. The effects of disruptions on the system performance will be shown using the predictive operation mode and the lifetime optimization operation modes I and II.

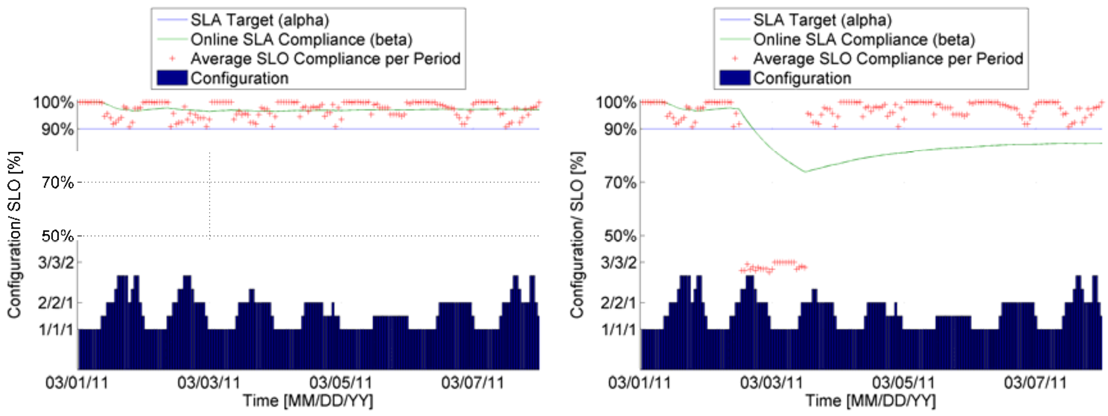


Figure 17: Predictive operation strategy in normal (left) and failure (right) scenario

Figure 17 shows the resource management of the IT system. On the X-axis we have plotted the time when user requests reach the system. The stochastic user requests are indirectly depicted by the average SLO compliance per period, which renders the ratio of processed user requests. The used configuration to process the user requests is depicted by the black surface at the bottom of the graph. On March 1, 2011, the configuration of one web-server, one application server and one database server is used. A couple hours later the configuration was changed adding one more application server. The SLA target is represented by the horizontal line at the 90% SLO compliance. The online SLA compliance per period denotes the status of the current SLA compliance.

Figure 17 illustrates the results for the predictive operation mode facing the two scenarios. The left figure shows the system behavior under the 'normal scenario', whereas the right during the 'failure scenario'. During the 'normal scenario', the system aims at maintaining the SLO compliance in each period above the 90% SLO compliance. As a consequence, the predictive operation mode complies with the SLA at the end of its lifetime. The left figure reveals that the final SLA compliance (Online SLA Compliance) is significantly higher than the specified target, as each period is managed individually. Although the predictive operation mode leads to a higher Quality of Service it results in the over-provisioning of the system and is hence inefficient.

Result 7: *Even in absence of disruptions (e.g. flash crowd effects, temporary hardware failure) a predictive operation mode that aims at maintaining the SLOs at any period results in the over-provisioning of the system.*

In the failure scenario, the Online SLA compliance drops to 73% in reaction to the temporary disruptive shock. Once the system is back to its normal state, the online SLA compliance increases again as the predictive operation mode tends to over-fulfill the SLO (refer to result 7). Nevertheless, as the predictive operation mode still aims at achieving the original SLO per period, the performance limitations due to the disruptive shock cannot be compensated during the rest of the SLA lifetime, which finally leads to a violation of the agreement.

Result 8: *Facing a temporary disruptive shock (e.g. flash crowd effects, temporary hardware failure) a predictive operation mode that aims at maintaining the SLOs at any period results in SLA violation.*

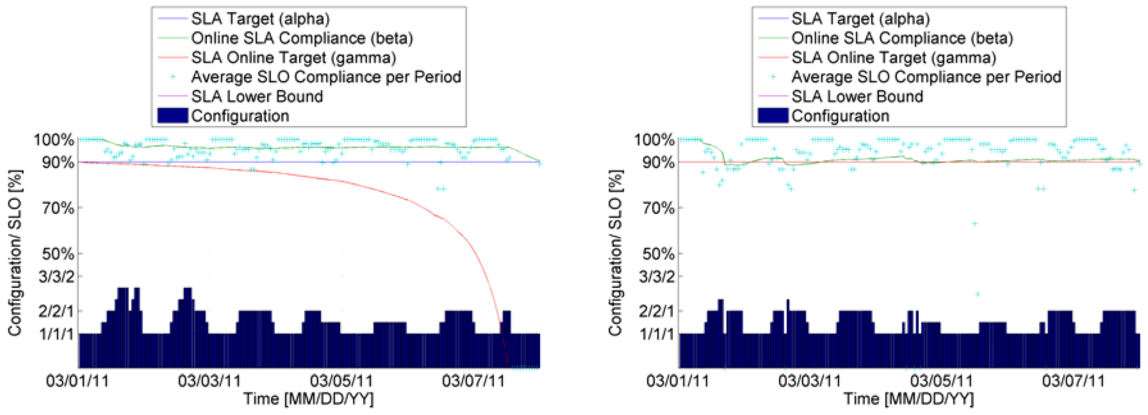


Figure 18: Lifetime optimization strategy without (left) and with (right) long term forecast

As predictive operation modes perform poorly, we turn our attention to lifetime optimization operation modes. The left graph of figure 18 shows the results for a lifetime optimization operation mode, the right for the lifetime optimization with long-term forecast in a 'normal scenario'. Both lifetime operation modes aim at complying exactly with the SLA. Evidently, both modes attain that the online SLO compliance exactly hits the SLA target in any period.

Result 9: *In absence of disruptions lifetime optimization operations meet the SLA without incurring an over-provisioning of the system.*

The lifetime optimization operation mode continuously updates the SLA target (refer to the SLA online target schedule in Figure 18) for each period. Similar to the predictive operation mode, it tends to exceed the SLO goal in the first period. Due to this over-fulfillment of the SLA, the SLA target is gradually decreased to avoid over-provisioning. As a consequence, at the end of the SLA lifetime, the target SLA drops to 0% percent, as the lifetime optimization operation mode has already met the SLA

Result 10: *In absence of disruptions lifetime optimization operations gradually decrease the SLA target up to the point where no request is processed at all.*

The lifetime optimization with long term forecast seeks for the optimal configuration by optimizing the system configuration over the whole SLA lifetime. This global optimization enables the provisioning of weaker configurations throughout the whole lifetime, still complying with the SLA. However, while the lifetime optimization operation mode significantly reduces the number of resources at the end of the SLA lifetime, the lifetime optimization with long-term forecast does not need to reduce the resources as it has already selected configurations with comparably low SLO compliance throughout the whole SLA lifetime.

Results 9 and 10 demonstrate that both lifetime optimization operation modes meet the SLA target without over-provisioning the IT system. However, the both modes refer to inappropriate configurations that are infeasible in enterprise information system operation. To avoid this undesired effect we introduce a lower SLA bound. During operation both modes only select configurations which do not fall below this lower bound.

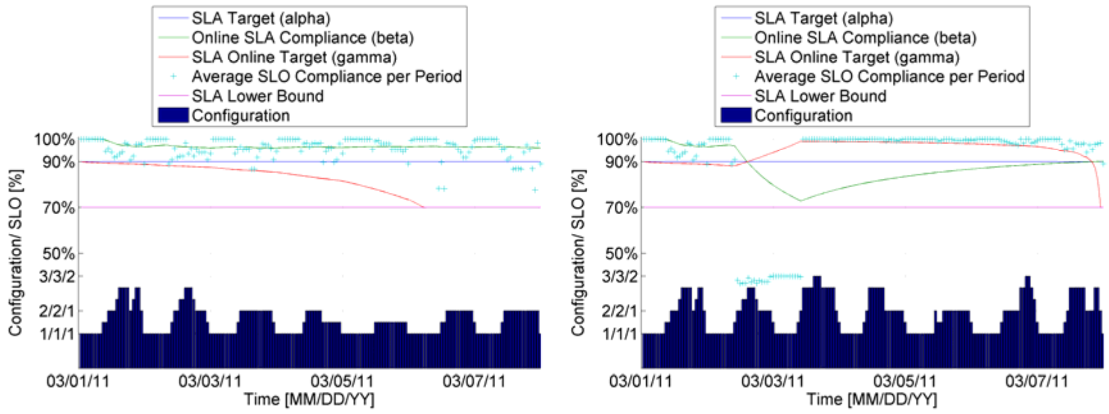


Figure 19: Lifetime optimization strategy in normal (left) and failure (right) scenario

Having introduced the lower SLA bound, the lifetime optimization operation delivers continuously a high QoS. Nevertheless, the lower bound effects an over-fulfillment of the SLA (see left graph in Figure 19).

In the 'failure scenario', the online SLA compliance drops again to 73%. However, during the failure the lifetime optimization operation mode continuously adapts the online SLA target up to a value of 99%. As a consequence, directly upon resolution of the failure, the lifetime optimization operation mode uses a very resource-intensive strategy to compensate for the previous low SLO compliance levels. Close to the end of the SLA lifetime this mode facilitates the system to comply to the SLA again.

Result 11: *The lifetime optimization operation mode with a lower SLA bound over-provisions the IT system in normal conditions, but succeeds in SLA compliance without waste in case of temporary disruptive shocks.*

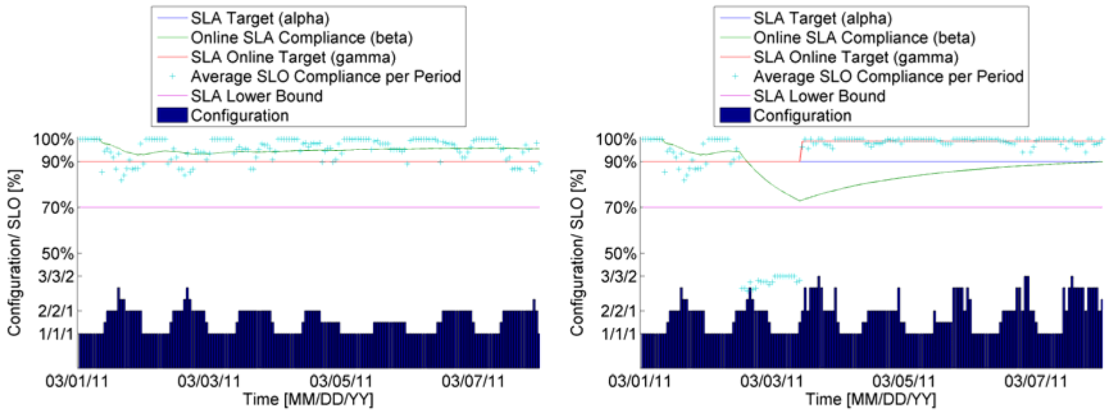


Figure 20: Lifetime optimization in normal (left) and failure (right) scenario

Our SLA lifetime optimization with long term forecast does not only take the near future into its consideration but the state of the system over the whole SLA lifetime. Accordingly, having sustained a temporary system failure, this SLA lifetime optimization mode behaves more moderately by not trying immediately to compensate for the error. Instead the mode aims at leveling out the low performance by dedicating more resources to periods where performance compensations are feasible. Following the failure, the SLA online target is set to 99% and the operation mode aggressively pursues this new target. It should be noted that this mode works well in theory but is very sensitive to long-term forecast.

***Result 12:** The SLA lifetime optimization operation mode with long term forecast succeeds in SLA compliance without waste in case of normal conditions and when faced with temporary disruptive shocks.*

Table 4: Performance of different operation modes

	Normal Situation		**System Failure**	
	SLA Compliance	**Resource Units**	**SLA Compliance**	**Resource Units**
Static SLA Model	97.2%	735	85.1%	735
Lifetime Optimization (LO)	96.1%	707	90.4%	820
LO with lower SLA bound	90.6%	679	/	/
LO with long term forecast	95.6%	692	90.1%	800
LO with both features	90.4%	688	/	/

Table 4 summarizes the final results of the previous graphs. In the ‘normal scenario’, all three operation models comply with the SLA. The predictive optimization operation mode tends to over-fulfill the SLA, whereas both lifetime optimization modes meet the SLA target very precisely. The strict adherence to SLA compliance is infeasible if it suggests ceasing the provisioning as an end game effect. Thus, this mode requires the inclusion of a lower SLA bound. Evidently, this lower bound diminishes the efficiency, but provides better average SLO compliance. Overall, the lifetime optimization operation modes allow a flexible, green management of enterprise information systems. Apparently, the modes align the operation strategy to the business value of the system by strictly managing the system according to the SLA. Second, they can proactively modify the performance goals and thus compensate minor failures.

5. Conclusion

In this article, we presented an adaptation engine framework for sustainable green management of enterprise information systems. Green IS philosophy mandates the integration of SLAs into agile management at runtime and the alignment of SLAs with the business goals (e.g., anticipated future performance of the IT system), which allows our approach to directly harmonize green goals with the business value chain. Moreover, our concrete experiences showed that the integration of the Green IS paradigm into existing IT infrastructure and systems is non-trivial; for instance, volatile workloads as well as significant reconfiguration lead times demand complex operation strategies to meet QoS and green goals simultaneously. In particular, in the case of large enterprise ISs, resource adaptive operation modes are often the only feasible and implantable option.

Conceptually, our adaptation engine framework is a model that includes SLAs in the operation of IT systems in a dynamic and flexible fashion. To accommodate the heterogeneity of the domain, the engine was designed modularly, enabling different configurations according to actual system characteristics. During operation, the adaptation engine systematically analyzes all key factors of influence, such as hardware characteristics and software design, to evaluate all these factors in a compact decision model. Based on the decision model and a strategy function, the adaptation engine manages the enterprise IS. In our evaluation we discussed several different types of such functions and found that flexible and dynamic SLA lifetime optimization proved to be highly effective in aligning the IT operation with the business value chain.

The evaluation of our adaptation engine prototype, based on a real production system workload trace, was carried out in a custom test infrastructure (i.e., cloud testbed, n-tier benchmark application, distributed monitors, and control framework). We systematically investigated practicability, IS optimization potential, green effectiveness, and business value chain orientation, and we showed that the integration of our adaptation engine allows flexible IS operation with up to 46 percent lower cost of operation.

Although this paper focused on the green management of enterprise ISs, the design of the adaptation engine framework can be adapted to various other aspects of enterprise computing such as the allocation of resources in multi tenancy systems or the provisioning of network bandwidth. In the future, we plan to extend the adaptation engine framework to the management of multiple concurrent systems competing for scarce resources. We believe that our dynamic SLA operation strategies pave the way for highly effective resource allocation in enterprise IT environments, which enables further increases in overall efficiency.

6. References

Anon, Wikistats. Available at: http://dammit.lt/wikistats/ [Accessed August 30, 2011].

Ardagna, D., Trubian, M. & Zhang, L., 2007. SLA based resource allocation policies in autonomic environments. *Journal of Parallel and Distributed Computing*, 67(3), pp.259-270. Available at: http://dx.doi.org/10.1016/j.jpdc.2006.10.006 [Accessed July 16, 2010].

Carr, N.G., 2004. IT doesn't matter. *IEEE Engineering Management Review*, 32(1), pp.24-24. Available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1292391 [Accessed April 3, 2011].

Chandra, A., Gong, W. & Shenoy, P., 2003. Dynamic Resource Allocation for Shared Data Centers. *Quality of Service — IWQoS 2003*, Volume 270, pp.381-398. Available at: http://www.springerlink.com/content/h56r570l4u707466.

Cohen, I. et al., 2004. Correlating instrumentation data to system states: a building block for automated diagnosis and control. *Operating Systems Design and Implementation*, p.16. Available at: http://portal.acm.org/citation.cfm?id=1251270 [Accessed September 15, 2010].

Gmach, D. et al., 2009. Resource pool management: Reactive versus proactive or let's be friends. *Computer Networks*, 53(17), pp.2905-2922. Available at: http://dx.doi.org/10.1016/j.comnet.2009.08.011.

Gmach, D. et al., 2007. Workload Analysis and Demand Prediction of Enterprise Data Center Applications. *2007 IEEE 10th International Symposium on Workload Characterization*, pp.171-180. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4362193.

Hevner, A., March, S. & Park, J., 2004. Design Science in Information Systems Research. *Management Infomation Systems Quarterly (MISQ)*, 28(1), pp.75 - 105. Available at: http://www.citeulike.org/group/11373/article/5637737 [Accessed April 3, 2011].

King, A.A. & Lenox, M.J., 2009. Lean and Green? An empirical examination of the relationship between leadn production and environmental performance. *Production and Operations Management*, 10(3), pp.244-256. Available at: http://doi.wiley.com/10.1111/j.1937-5956.2001.tb00373.x [Accessed April 3, 2011].

Koomey, J.G., 2007. Estimating total power consumption by servers in the U.S. and the world. *World*. Available at: http://www.greenbiz.com/research/report/2007/09/12/estimating-total-power-consumption-servers-us-and-world.

Padala, P. et al., 2009. Automated control of multiple virtualized resources. *Proceedings of the fourth ACM european conference on Computer systems - EuroSys '09*, p.13. Available at: http://portal.acm.org/citation.cfm?doid=1519065.1519068.

Powers, R., Goldszmidt, M. & Cohen, I., 2005. Short term performance forecasting in enterprise systems. *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining - KDD '05*, p.801. Available at: http://portal.acm.org/citation.cfm?doid=1081870.1081976.

Schulz, G., 2009. *The Green and Virtual Data Center*, CRC/Auerbach Publications.

Urgaonkar, B. et al., 2008. Agile dynamic provisioning of multi-tier Internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1). Available at: http://portal.acm.org/citation.cfm?id=1342172.

Watson, R.T., 2007. *Information Systems* R. T. Watson, ed., Global Text Project. Available at: http://globaltext.terry.uga.edu/userfiles/pdf/Information Systems.pdf.

CHAPTER IV

DYNAMIC SERVICE LEVEL AGREEMENT MANAGEMENT FOR EFFICIENT OPERATION OF ELASTIC INFORMATION SYSTEMS

This chapter is a revised version of the paper:

Hedwig, Markus; Malkowski, Simon and Neumann, Dirk, "Dynamic Service Level Agreement Management for Efficient Operation of Elastic Information Systems" (2011), International Conference on Information Systems 2011 Proceedings.

Abstract

The growing awareness that effective Information Systems (IS), which contribute to sustainable business processes, secure a long-lasting competitive advantage has increasingly focused corporate transformation efforts on the efficient usage of Information Technology (IT). In this context, we provide a new perspective on the management of enterprise information systems and introduce a novel framework that harmonizes economic and operational goals. Concretely, we target elastic n-tier applications with dynamic on-demand cloud resource provisioning. We design and implement a novel integrated management model for information systems that induces economic influence factors into the operation strategy to adapt the performance goals of an enterprise information system dynamically (i.e., online at runtime). Our framework forecasts future user behavior based on historic data, analyzes the impact of workload on system performance based on a non-linear performance model, analyzes the economic impact of different provisioning strategies, and derives an optimal operation strategy. The evaluation of our prototype, based on a real production system workload trace, is carried out in a custom test infrastructure (i.e., cloud testbed, n-tier benchmark application, distributed monitors, and control framework), which allows us to evaluate our approach in depth, in terms of efficiency along the entire SLA lifetime. Based on our thorough evaluation, we are able to make concise recommendations on how to use our framework effectively in further research and practice.

Keywords: Service Science, Adaptive provisioning, Green IT/IS, Service management, Service Level Agreements, Workload Forecast

1. Introduction

Effective information systems (IS) have increasingly matured to critical success factors in modern enterprises. For instance, the ability to process extensive business data quickly or the operation of innovative and powerful customer portals have become vital necessities in today's dynamic business world. In this sense, effective adaptation to new technology trends as well as agile response to changing customer requirements are key to sustainable business practices. Nonetheless, through a strict focus on functionality, IT operation expenditures have typically been neglected; however, a rising cost of operation and the awareness of its substantial environmental foot print has risen corporate IT out of its "shadowy existence" into a key challenge for the next decades (Koomey 2007).

Current market predictions estimate an annual growth of the server market by 5% (Gartner 2010). Given this trend, information systems will most probably continue to grow in size and complexity, and thus the challenge of the steadily growing consumption of resources persists. Green IT initiatives have already taken up this problem and offer a variety of innovations and best practice to reduce the economic and environmental impact of (T. Velte et al. 2009; Schulz 2009). However, thorough optimization and enhancement efforts are often beyond the expertise of medium-sized and small enterprises, therefore requiring high initial investments.

With the emerging service world, companies have obtained a powerful alternative to their classic in-house IT operation concepts. For instance, cloud computing enables enterprises to outsource parts of their physical IT to service providers and buy back computing power on a pay-as-you-go basis. In contrast to traditional in-house operation, service providers can leverage economy of scale and scope effects. Thus, by offering their services to multiple customers simultaneously and building up expertise, they can provide corporate computing demands more cost effectively. Within the service world, Service Level Agreements (SLAs) have become the common practice to define the terms of business between two parties. On the one hand, they contain the Service Level Objectives (SLO), such as response time requirements; on the other hand, they include the financial arrangements for the service operation. While in the beginning service providers mainly offered their services on a best-effort basis, today strict binding SLA’s are slowly being established in the market (Sahni & Tan 2011; Goolsby 2007).

Nonetheless, the adaptation to the new service technology as well as the sensible utilization of IT will be one of the key challenges of enterprise computing in the next decade. Even though various technological advances as well as new software design paradigms enable highly efficient operation modes today (Bailey 2009), the feasible implementation bears further problems in the real operation. For instance, this includes the relation between the Quality of Service (QoS) and the economic parameters of a service contract (e.g. price and penalty). In this context, we present our new integrated Service Level Agreement (SLA) operation model, which contributes to the sensible and economic utilization of IT. More concretely, we designed and implemented a new approach for the operation of large enterprise information systems. In contrast to existing concepts in the field of green technology, our model does not solely aim at reducing the resource consumption. Instead, our work extends the current state of the art by incorporating both *technical and economical* parameters of SLAs into the system operation strategy. By correlating the economic value of the system (i.e. profit and penalties), cost of operation, user behavior, and performance characteristics, our integrated management approach derives profit optimal operation strategies. Accordingly, our model facilitates highly efficient operation strategies for information systems while at the same time guarantees a continuously high QoS.

In this paper we focus on large web-facing enterprise information systems that provide their services to a large number of concurrent users. The workload of these systems is typically characterized by continuously varying size (i.e., workload) and composition (i.e., workload mix) of simultaneous requests, where each single request only generates a relatively small system load. Common representatives of this group are applications such as e-commerce portals and bulletin board platforms. Inherently, these end-user driven systems often face a highly volatile workload as usage patterns are typically characterized by a strong seasonality component (e.g., time of day or day of week). In addition it is noteworthy that the systems are traditionally over-provisioned because QoS is apparently valued higher than cost savings.

In summary, these enterprise information systems often suffer from an inefficiently low average utilization of computing resources. Furthermore, these systems are typically operated on commodity hardware or on entry level servers in order to reduce the cost of operation (Short et al. 2011). While virtualization and consolidation may help significantly to mitigate some of these problems for relatively small instances of applications, virtualization and consolidation as such do not directly affect the efficient operation of large information systems, which require the computing power of several nodes. One feasible solution for large systems is the use of resource adaptive operation modes, which adapt continuously the system size to the user demand.

However, the implementation of such systems is non-trivial. In fact, if the operated service is provided on the basis of an SLA, the problem becomes particularly hard because efficient operation modes inherently increase the risk of violating performance constraints (i.e., QoS requirements). Even worse, if we consider different SLA configurations for a single service, the problem becomes increasingly complex. For instance, a high-priced service should be managed more conservatively than a low-cost service. In Addition, our work extends the traditional static view on Service Level Objectives (SLOs) towards a fully dynamic notion of SLA compliance. By continuously evaluating the SLA compliance at run time and adapting the SLOs in real-time, the system can react to changes in its environment instantaneously. This dynamic view on SLAs significantly reduces the risk of violating SLAs at runtime based on wrong or outdated operational decisions.

To account for the heterogeneity of IT systems, our integrated management model utilizes a complex system performance model and a workload forecast model. Based on these components, our information system operation model can determine the impact of different operation strategies and find a profit optimal configuration at runtime.

The high complexity of enterprise system environments entails that any attempt trying to prove the validity of a novel approach to management that bases on data analysis alone is not reliable due to the non-linearities in system operation. Instead, the only way to validitate the goodness of a novel management approach is by relying on experiments on a real-life testbed. Accordingly, we developed a test environment to provide a reliable evaluation. Our test system comprehends of a cloud infrastructure, a benchmark application, extensive monitoring and control software, and the integrated, SLA aware management model itself. In order to provide a real-world evaluation scenario, we used a real production system workload process to generate our test workload. Our evaluations indicate that the application of our new resource management system allows reducing resource consumption of the IT systems under test conditions by up to 40 percent.

The main contribution of this work is threefold.

- We provide a new perspective on the management of enterprise information systems and introduce a novel framework that harmonizes economic and operational goals.
- Our novel dynamic SLA management model, which induces economic influence factors into the operation strategy, adapts performance goals of an enterprise information system dynamically (i.e., online at runtime).
- The integration of our model into a test environment allows us to evaluate our concept in depth, in terms of efficiency along the entire SLA lifetime. Based on our thorough evaluation, we are able to make concise recommendations on how to use our framework effectively.

The remainder of this paper is structured as follows. The next section discusses the design problem statement followed by the presentation of the related work. Subsequently, we discuss the foundation of the integrated information system management model, followed by a detailed presentation of the model. Finally, in the succeeding section, the model is evaluated. The last section summarizes the paper and provides a conclusion and an outlook on our future work.

2. Design Problem Statement

Keeping pace with continuous IT innovations and product life cycle enhancements has become extremely difficult and cost intensive for enterprises. As a consequence, IT landscapes often suffer from physical server sprawl, over provisioned information systems, and a lack of integrated management tools and service management frameworks (Bailey 2009). The efficient and QoS aware operation of elastic enterprise information systems requires a thorough understanding about all factors of influence that affect the overall performance of the system. Even though the idea of resource-adaptive operation of large applications seems to be very tempting, its application is absolutely complex. This complexity stems from the fact that the overall performance of systems depends on various, interdependent factors; changes in each factor can lead to critical performance limitations or – in the worst case – cause total system failure. In order to manage a system efficiently, the factors of influence need to be well understood.

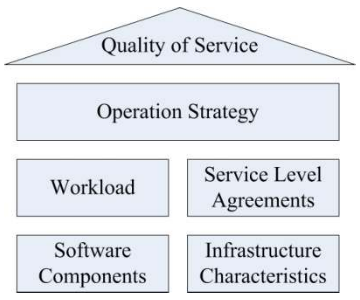


Figure 1: Factors of QoS Influence

Accordingly, we can categorize the following factors of influence (Figure 1) into the groups' workload, SLA, software components, as well as the infrastructure characteristics. Based on these factors, we have to derive an **operation strategy** or operation mode respectively. In classic system design, this would be the provisioning and configuration of an appropriate infrastructure. In modern service design, the operation strategy is more complex and can be adjusted to the preferences of the provider or customer. Furthermore, depending on the focus, there might be more dominant factors, such as data quality. Nonetheless, in this work we focus on the factors determining the system performance.

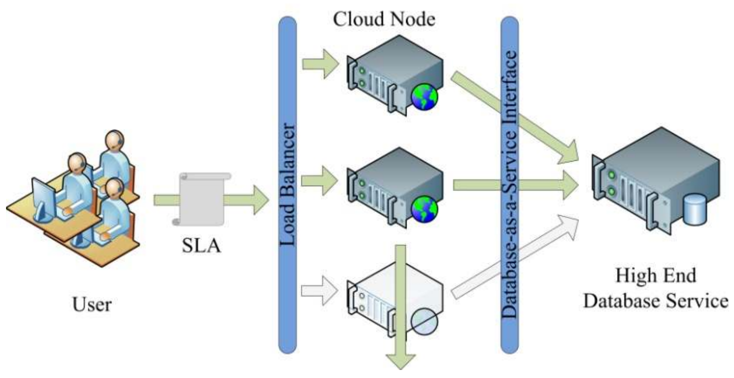


Figure 2: Elastic Application in a Cloud Computing Environment

The **infrastructure characteristics** and **software components** determine the performance characteristics of an information system. In our work, we use a state-of-the-art design for information systems in clouds (Figure 2). Elastic applications have emerged as the next generation of distributed systems. Compared to classic design concepts, they provide extensions enabling resource adaptive operation modes. In particular, this design is highly beneficial for cloud environments as it allows for utilizing the capability of the cloud to instantaneously allocate resources to systems increasing the scalability tremendously (RightScale 2011). Additionally, we utilize a database-as-a-service instead of a traditional database to avoid the problems of data synchronization. Due to their complexity, large information systems often reveal a complex performance behavior. In order to provision the optimal infrastructure configuration for any given workload, the characteristics of the software and the infrastructure must be known a-priori. System performance models are one approach to determine the expected QoS of a certain system configuration facing a certain workload level.

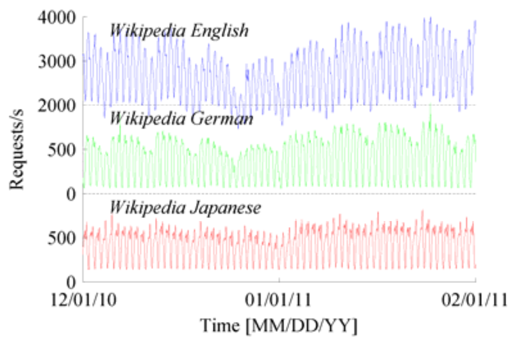


Figure 3: Wikipedia Workload Trace

Next to the system performance characteristics, the **workload** process is a key factor of influence determining whether or not a system is able to meet QoS objectives. Many end-user applications face highly volatile workloads, as the workload traces of Wikipedia (Figure 3) demonstrate. This volatility leads to low average utilizations in static system designs (i.e., the number of resources remains constant). By dynamically adding and removing resources (e.g. servers) from the system, the average utilization of the system can be significantly increased.

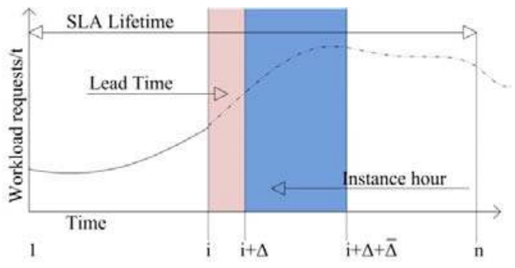


Figure 4: Reconfiguration Lead-time

Nonetheless, the optimal adaptation faces many challenges. More specifically, one key challenge is to overcome the reconfiguration lead time (Figure 4). Usually, the hardware configuration cannot be adapted instantaneously; rather there occurs a delay between the initiation of reconfiguration and its availability due to configuration of the resources, synchronization and reconfiguration of the load balancers. However, due to significant lead-times in the reconfiguration, resource adaptations need to be initiated in advance. The concrete lead time depends on the level of reconfiguration. Nonetheless for the purpose of our model, we assume that the lead time is constant. Thus, in order to mitigate the risk of SLA violations or system crashes, the systems cannot be managed on the basis of the current state, but rather require near-future workload predictions. At heart, workload processes (i.e. particular workload generated by a large number of users) are stochastic processes consisting of trend and seasonal factors as well as unpredictable anomalies such as sudden appearances of users (flash-crowd effect). Consequently, workload forecast mechanisms need to be customized for the individual characteristics of the workload process.

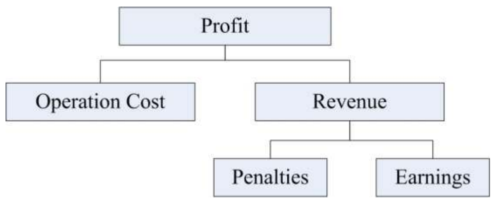


Figure 5: Provider Revenue Model

The final factor of influence is the **Service Level Agreement** itself. It defines the revenue and penalties as well as the cost of operation in our scenario. Figure 5 shows the structure of our economic model. The profit is defined as the provider revenue of the system less the infrastructure cost for providing the service. The ultimate goal of our information system operation model is to increase the resource efficiency of an enterprise information system by aligning the operation strategy to these economic parameters. To facilitate this alignment, we consider the industry scenario of a Software-as-a-Service (SaaS) provider, operating her application on the infrastructure of a third party Infrastructure-as-a-Service (IaaS) provider. Clearly, the application of our SLA model is not limited to this scenario, but for the purpose of this paper, this scenario enables a full cost assessment of the impact of resource adaptive operation modes. In the later evaluation, this helps us to determine the saving potential of our model in sevice operation.

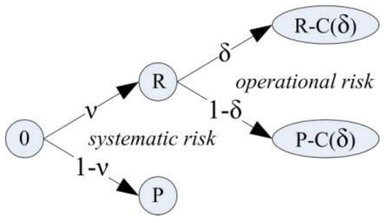


Figure 6: Risk view on SaaS Operation

Figure 6 illustrates the risk perspective of adaptive **operation strategies**. In static operation modes, the major risks are infrastructure failures, such as power outages. This systematic risk $(1-\vartheta)$ of IT system operation can mainly be countered by using high quality equipment. During the Service Level Agreement design phase, this systematic risk is typically included in the price for the service offer. In addition to this systematic risk, resource adaptive operation modes exhibit an operational risk factor. For instance, sudden peaks in the workload process can cause sharp drops in performance causing SLA violations. Those situations could have been mitigated using larger configurations. In this sense, we define $(1-\delta)$ as the risk that a certain strategy violates the SLA. By using more resources, this risk can be reduced or even eliminated: this comes however, at the price of higher operation cost $C(\delta)$. Theoretically, the provider can select a risk level δ which maximizes his profit. Evidently, this implies that operation strategies which embody a certain risk of SLA violation might be beneficial for the provider.

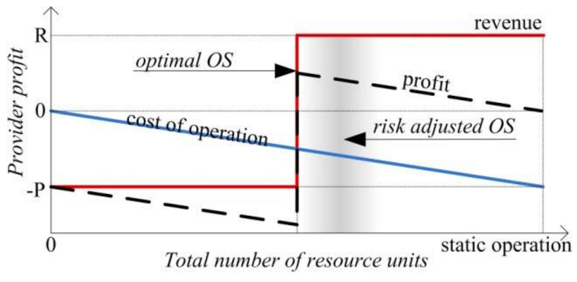


Figure 7: Profit depending resource usage

Having in mind that the optimal operation strategy is not risk free, Figure 7 depicts the economic situation of the service provider. The redline represents revenue and penalty defined in the SLA, the blue line the cost of operation and the black dashed line the resulting profit. Obviously, if the provider maintains too few resources, the SLA is ultimately violated, which entails the payment of a penalty. On the other hand, if the provider adds too many resources to the system, the corresponding costs of operation increase, reducing overall profit. Assuming the provider knows the optimal operation strategy a priori (i.e. the optimal infrastructure size at any given point during the SLA lifetime), he can choose the cost minimal strategy of the optimal point of operation. Albeit, this optimal strategy would maximize the provider’s profit, this strategy would induce a high risk, as minimal performance violations would directly lead to a negative profit. Given the uncertainty during operation, a rational (at least risk neutral or risk averse) provider would choose a less risky operation strategy in the gray shaded area in order to build up some performance reserves to compensate for the operational risk of sudden changes in the environment.

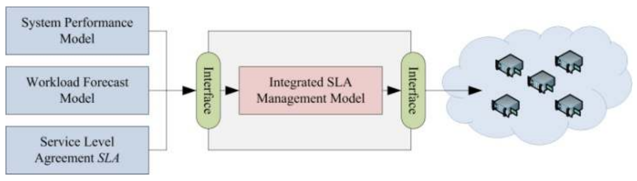


Figure 8: Integrated SLA Management Model Design

Based on the aforementioned factors of influence, we developed our integrated, SLA aware, information system management model (Figure 8). The goal of our model is to derive a cost effective operation strategy. Based on the revenue and penalty defined in the SLA and the cost of operation, our model aims to find a profit optimizing tradeoff between the cost of operation and the risk of performance violations. Our model manages the system on the base of a workload forecast model, a system performance model as well as on the specification of the SLA. The output of the aforementioned model is converted into the data format of our dynamic and integrated SLA management model and processed. Due to this modular design, our management tool supports a variety of these models (c.f. Related Work section). Additionally, our model encompasses actuator interfaces, which initiate, based on the real-time evaluation of the data, the reconfiguration of the infrastructure (i.e. add and remove servers) as well as the reconfiguration of the application (i.e. configure load balancers). In our later evaluation we will use a Fourier Transformation based forecast model and an empirical system performance model developed in our recent research.

3. Related Work

Our integrated, SLA aware management model combines various different research threads into an interdisciplinary model. In the related work section, we will discuss the different aspects of performance modeling, workload forecasting, and SLA management and present the current state of the art in research.

Performance analysis of large, distributed systems is a very active research field and a variety of models have been developed. Famous representatives are for instance (Urgaonkar et al. 2008), who used queuing models for automated resource allocation in information systems or (Cohen et al. 2004) who employed machine learning to model the performance characteristics. Both models are an alternative to our empirical system performance model presented in this paper. Most of the newer contributions in this field extend the performance models to dynamic research management systems. For instance, the authors in (Gmach et al. 2009) developed a reactive migration controller for virtualized environments. However, compared to our concept, their approach is only designed for basic single-layered systems. The paper (Chandra et al. 2003) introduces a resource allocation model for shared datacenters based on a queuing network performance model and a time series workload forecast mechanism. However, they do not consider SLA's in the provisioning process. Another concept (Padala et al. 2009) is an automated control model for virtual resources. The model manages the varying resource demands by dynamically allocating resources to or migrating virtual machines. Nevertheless, in modern cloud environments this migration approach is usually not supported. In (Ardagna et al. 2007) a model has been developed to manage the resource demand of multiple concurrent systems. In contrast to our model, it optimizes the system only for a single point in time, rather than incorporating the state of the SLA's. In the paper (Lassettre et al. 2003) the authors developed a surge protection model for dynamic resource allocation. The authors in (Lim et al. 2010) developed an autonomic control model to scale elastic storage systems based on the utilization of the system.

Most of the aforementioned work only subsidiarily discusses the impact of a varying workload. However, particularly in environments with highly volatile end-user workload, this factor appears to be the most decisive factors. The book (Feitelson 2011) provides a detailed overview of workload and workload modeling. In the paper (Urgaonkar et al. 2008) ,the authors present a workload forecast model based on an empirical distribution estimation of past workloads. The authors in (Gmach et al. 2007) used Fourier Transformation for data smoothing and applied time series analysis for workload prediction. In (Powers et al. 2005) the authors used time series as well as regression analysis for workload prediction. Nonetheless, workload processes differ strongly in their characteristics and thus forecast methods need to be individually adapted to the process characteristics. Accordingly, all aforementioned approaches are an alternative to our Fourier Transformation based workload forecast.

Service Level Agreements and their different aspects have been extensively discussed by the research community. The following paragraph presents an overview of related and complementing work in this field. The article (Buyya et al. 2009) provides a good overview of SLAs in the field of clouds. The authors in (Yeo & Buyya 2007) developed an integrated risk analysis scheme to analyze the effectiveness of resource management policies. Based on SLAs, they determine whether a system is capable of meeting the required objectives and whether the acceptance of a single job is economically feasible. The paper (Aib & Boutaba 2007) presents an approach to business and policy driven refinement in application hosting environments. Their featured model focuses on QoS objectives and includes a mechanism for runtime adaptation. In contrast to our work, both focus on batch processing and thus do not require coping with dynamic workloads, performance and SLA components. In the article (Hasselmeyer et al. 2006) , the authors introduce a model for the automatic negotiation of the Service Level Agreements prior to the contract start. (Buco et al. 2004) develop a business-objective-based SLA management system over the whole lifecycle of the agreements. Similarly, the paper by (Koller & Schubert 2007) presents architecture for autonomous QoS management based on SLA specifications. The paper (Sahai et al. 2001) sketches a general scheme for Service Level Agreements which allows the autonomic management in services systems. In the same direction, the paper by (Raimondi et al. 2008) presents the implementation of an automated SLA monitoring for services. Although our model does not cover all technical and negotiation aspects of the SLA, a productive version would require such an SLA management concept. In summary, most aspects of our dynamic

SLA management model haven solved individually. However, we haven't found any work combining all aspects into a single integrated model for enterprise information systems.

4. Foundation of the Dynamic SLA Management Model

Having explained the factors of influence and the requirements for cost effective and risk aware operation of enterprise information systems in clouds, we will now present the foundation of our dynamic and integrated SLA management model. More concretely, we will discuss our recent and ongoing work in the fields of SLAs, system performance modeling, workload forecast, and information system operation strategies.

4.1. Service Level Agreements

Service Level Agreements specify all aspects of business relations between the contract partners, such as the rights and duties of each party, contract duration as well as guarantees and warranties. The QoS requirements are distinguished into Service Level Objectives (SLOs), Monitoring Intervals and SLA assertion. SLOs consist of specific measurable characteristics of the system (e.g. availability, throughput, response time) together with a threshold value for this characteristic. Furthermore, an SLA includes monitoring intervals, which define when compliance with the SLO is checked. The SLA assertion can combine several SLOs and specifies under what conditions the SLA is violated. In addition, it defines the lifetime of the SLA as a contract and also controls the penalty payments in case of SLA violation. Different SLAs may lead to very different operation strategies.

In the following presentation, we assume a basic SLA definition. The SLO demands a response time of the information system below $\theta = 500\,ms$ for each request. The SLA between the SaaS provider and consumer specifies that this SLO target must be met for α =95% of all requests on average, measured in intervals of 1 $second$. If the provider fulfills the SLA, he receives a payment of R; if he fails to comply, he has to pay a penalty of P (in the remainder of the paper, the penalty is defined as a negative number, i.e. $P = -100$). The SLA lifetime is set to $T = 24$ hours. Next to the SLA specifications, we additionally define that our controller is applied every $d = 15$ minutes. This allows us to divide the SLA lifetime in $n = 96$ periods. We further define i as the current time index and $w = w_1, \dots, w_m \in W$ as the workload process during the SLA lifetime. Evidently, the workload data of w_j with $j > i$ are unknown at time index i. Furthermore, we assume that the provider operates the information system on the resource of an Infrastructure-as-a-Service (IaaS) provider. The cost for an instance hour is defined as $\overline{K}$.

4.2. System Performance Model

Elastic applications are one solution towards highly efficient operation modes of large enterprise systems in clouds. Recall that the provisioning the optimal hardware configuration for any given workload, a thorough understanding of the system characteristics is necessary. Usually elastic applications do not scale linearly with the amount of hardware resources. In particular distributed systems usually show complex performance characteristics, which are caused by various, non-trivial interdependencies within and between the different layers of the system. In our recent work (Malkowski et al. 2010; Malkowski et al. 2011), we developed an observation based, empirical approach for the performance modeling of large systems. In contrast to other models, our approach solely relies on the observed system behavior during operation. More specifically, our system characteristics model monitors the workload process, system metrics, as well as the SLO relevant metrics (e.g. response time) and saves the data in an operational data store. Based on this recorded data, our performance model can predict the expected degree of SLO compliance of a certain configuration c under a workload level w. For the purpose of this paper, we used a relatively simple application design (Figure 2), which scales along the number of cloud resources $c = c_1, \dots, c_q \in C$. In our scenario, the database service is assumed to deliver constant QoS, independent of the system load. In the following presentation we define lambda $\Lambda(c, w)$ as the system performance model function providing the degree of SLO compliance for a certain workload w and configuration c.

For instance, a system with $c = 3$ resource units facing a workload level of $w = 250$ might be able respond to all requests within a response time $\Theta = 500\text{ ms}$. In this case our degree of SLO compliance is $\Lambda(3{,}250) = 100\%$. If we reduce the system size by one node to $c = 2$, then system most likely reaches an overload state and the degree of SLO compliance consequently drops to $\Lambda(2{,}250) = 50\%$.

Generally, the system performance function $\Lambda(c,w)$ has a value range between 0% and 100% and can be interpreted as a step-wise defined function. If the system is overloaded, the system performance function will tend towards 0%. If the system is only moderately utilized, the system performance function $\Lambda(c,w)$ is near 100%. The degree of SLO compliance will obviously increase with the number of resource units c and decrease with the increasing workload level w.

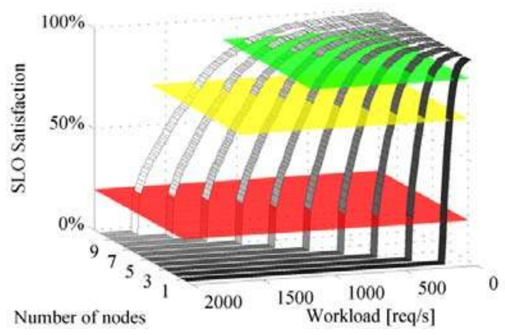


Figure 9: System Performance Curves

Figure 9 shows the performance characteristics of our benchmark application used in the evaluation. The value axis represents the degree of SLO compliance, whereas the description axis presents the workload and the configuration (i.e. number of nodes in the system). The range of high QoS is delimited with the green surface. In the area between the green and the yellow service, the system is still fully functional, although it may exhibit reduced response time. For most applications occasionally operating in the yellow surface (SLO compliance between 95% and 50%) is acceptably. If the workload drops the SLO satisfaction below the yellow surface, the SLA might be compromised. Beneath the red surface, the system will enter a critical, completely overloaded state. If the system enters this area, a total system failure is likely to occur. In our benchmark system, the smallest configuration c_1 is able to achieve a $\Lambda(c_1, 30)$ =100% SLO compliance up to 30 concurrent users. However, depending of on SLO specification, the configuration can account for up to 90 concurrent users (Compliance target $\alpha = 95\%$) or 110 concurrent users ($\alpha = 90\%$), respectively.

4.3. *Workload Forecast*

Modern elastic applications allow resource adaptive operation without compromising system stability. Nevertheless, significant reconfiguration lead times demand to reconfigure the system in advance (Hedwig et al. 2009). In our recent research (Hedwig et al. 2010), we developed a Fourier Transformation based workload forecast model. The core idea of our model is to decompose the workload process with the help of the Discrete Fourier Transformation into its single spectra components. In particular, an end-user generated workload exhibit predominant seasonal factors of influence (e.g. time of day). By identifying and isolating these components, we can predict the near futures of the workload process with high accuracy. Due to space constraints we omit the detailed presentation of the forecast algorithm and define Omega (*1*) as the workload forecast function. Based on the past workload observations , we estimate the near future of the workload process from the current index to the end of the SLA lifetime .

Our forecast model, as well as most mechanisms in literature, usually delivers point estimates for the workload in the near future. However, the later dynamic SLA management model demands distribution estimates in order to facilitate a more robust operation. Thus, we complemented our forecast model with empirical prediction error estimation. Based on the comparison between the forecast and the later observation, the accuracy of the prediction is determined. The prediction error estimation is defined in the form of a matrix (*2*), whereby refers to the forecast horizon of the error estimation. The single elements of the matrix contain the probability to face a workload level in period .

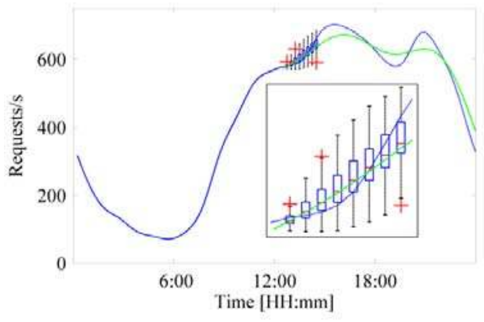


Figure 10: SLA Lifetime Forecast

Figure 10 shows the application of the workload forecast and the error distribution on a Wikipedia workload trace (Mituzas 2011). The blue line presents the observed workload process and the green line the forecast. For the first two hours of the forecast, the prediction error distribution is depicted by the box plots. Without a detailed analysis, we see that the workload forecast provides a good forecast of the process until the end of the SLA lifetime.

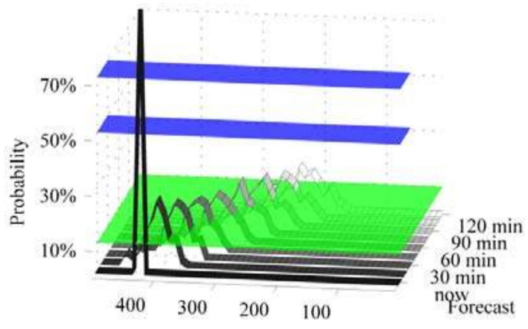


Figure 11: Workload Forecast Error Distribution

Figure 11 depicts the workload forecast error distribution. The first peak represents the current workload level; the following curves show the prediction distribution of the forecast periods. For the near future the prediction is very accurate, though the accuracy decreases with the increasing forecast horizon. Both components (ω and $\dot{W}$) are later used by the dynamic SLA management model to derive efficient operation strategies online.

4.4. *Naïve Operation Strategies*

In this section, we provide a brief overview on common operation strategies in order to show the origin of our dynamic SLA management model. We assume that the respective controllers are executed n times over the SLA lifetime, whereby the index of the current period is defined as i. Based on the most recent monitoring data, the controllers determine an optimal configuration $s \in C$ for the next period. However, as the system may have a significant lead-time Δ (defined as a multiple of the controller execution interval), the reconfiguration decisions will first be available in $i + \Delta$. The following strategies are presented in their most basic version.

$$s_{i+\Delta} = \text{constant} \quad = S^{\text{static}}(\emptyset) \quad (3)$$

The most basic operation strategy is a static system operation mode that keeps the hardware configuration unchanged over the whole SLA lifetime (*3*). The static system operation mode reflects a classical operation strategy that is naturally extremely costly, as the system needs to be adjusted to the peak workload. In the later evaluation, we will use the static operation mode as baseline.

$$s_{i+\Delta} = c = S^{\text{reactive}}(c^*, \alpha, w) = \begin{cases} \text{scale up: } c = c+1 & \Lambda(c^*, w) < \alpha \\ \text{scale down: } c = c-1 & \Lambda(c^*-1, w) \geq \alpha \\ \emptyset & \text{else} \end{cases} \quad (4)$$

The basic reactive operation mode (*4*) follows an observation-based operation strategy. The basic intuition is that the system configuration will be scaled up if the current configuration c^* cannot satisfy the SLO at a desired compliance target of α. This reactive controller reflects the current state of the art of cloud providers. For instance Amazon Simple Queue Service (Amazon 2011b) stores incoming requests in a queue. If the queue exceeds a predefined length, the system is scaled up. Although this controller type significantly reduces the resource consumption, it incorporates the inherent risk of performance violation as systems often cannot be reconfigured ad-hoc and the reconfiguration decision is solely based on the current workload level w.

$$s_{i+\Delta} = c = S^{\text{predictive}}(\alpha, \Delta, \omega) = \underset{c \in C}{\arg\min}\{c \mid \Lambda(\omega_{i+\Delta}, c) \geq \alpha\} \quad (5)$$

While the reactive operation modes base configuration decisions upon the current situation of the system, the predictive operation mode (*5*) uses the workload forecasts ω to make the decision. The intuition is the same as for the basic reactive operation mode. Scaling up becomes necessary if the current configuration c^* cannot meet the SLO target compliance in the near future, during which the configuration is fixed due to the lead time. In those cases it is necessary to increase the system configuration now, such that the advanced configuration at time $i + \Delta$ is available to satisfy the SLA accordingly.

5. Dynamic and Integrated SLA Management Model

Based on the background of the previous section, we now present our dynamic and integrated SLA management model. The core idea of our model is to feed (i) the economic parameters of the SLA document (i.e. revenue and penalty) and (ii) the cost of operation into the operation strategy of an enterprise information system. However, as the criterion of complying (violating, respectively) with an SLA is binary, the inclusion of the economic parameters into the online management strategy is difficult to achieve. In order to identify the optimal tradeoff between the cost of operation and the risk of violating the SLA, we developed a pricing and valuation scheme for the different cost factors on the period level i. This allows us to assess the economic impact of different configuration options in real time. This in turn allows us to derive the efficient configuration for any period i in real time given the recent and anticipated future performance. In this section, we will first define our management model. Based upon this dynamic model, we introduce the core of our SLA management model. Subsequently, we will discuss different configuration options with respect to their applicability.

5.1. Dynamic SLA Model

As mentioned in the related work section, SLAs have become the standard for defining the terms of operation in IT infrastructures. SLAs specify the revenue for maintaining the services, penalties for QoS violation, monitoring metrics, and corresponding thresholds. However, today SLAs are predominantly treated in a static way. Accordingly, SLAs are translated into a static set of requirements at design time of the IT services. During runtime, the system is provisioned according to these requirements. Thus, the SLOs, defined in the SLA, are merely used for monitoring the system and determining if and if so, when the SLA is violated. The previous operation modes aim at satisfying the SLO compliance target α, defined in the SLA, at any point in time. By continuously providing a performance level above α, SLA violations are avoided. However, SLAs are typically giving more flexibility to providers, as they only require fulfilling the SLO target over the entire SLA lifetime. For instance, if the initial target α is not met at the beginning of the SLA lifetime, system operation should aim at providing better QoS during the remaining lifetime such that the SLA is still satisfied. For instance, if the SLA requires a response time below one second in $\alpha = 95\%$ of all cases, the SLA can still be met if the response time of the system was initially below 95 % (say 94 %) by offering a better response time during the remaining lifetime (say 96%). In this case the SLO target is set to 96% in order to achieve a 95% SLO compliance on average and thus comply with the SLA.

$$\alpha = \delta\beta_i + (1 - \delta_i)\gamma_i \Leftrightarrow \gamma = \frac{(\alpha - \delta\beta_i)}{1 - \delta_i} \quad (6)$$

Our SLA model exploits this idea by optimizing over the SLA lifetime. Based on the specified SLO compliance target α, our model continuously evaluates the current degree of compliance β (i.e. the average SLO compliance up to the current period) and determines an updated SLO target γ for the period i online. If the system complies with the new target, the SLA will be fulfilled. Equation (*6*) provides the formal definition: Essentially, we define $\delta = i/n$ as the fraction of the elapsed SLA lifetime. The target SLO compliance α must be equal to the weighted current SLO compliance β and the SLO compliance in the remaining lifetime γ. Solving this equation for γ gives us the required SLO compliance for the remaining SLA lifetime. This approach facilitates updating the performance target during operation continuously succeeding in a more fine grained and sensible operation of the system.

$$\bar{\bar{\beta}} = \frac{\beta * (i-1) + \sum_{j=i}^{i+\Delta-1} \Lambda\left(c_j, \omega_j\right)}{i+\Delta-1} \quad (7)$$

$$\bar{\bar{\gamma}} = \frac{\alpha - \delta' * \bar{\bar{\beta}}}{1-\delta'}, \delta' = \frac{i+\Delta-1}{n} \quad (8)$$

$$\hat{\beta}^{c,w} = \frac{\bar{\bar{\beta}} * (i+\Delta-1) + \Lambda\,(c,w)}{i+\Delta} \quad (9)$$

$$\hat{\gamma}^{c,w} = \frac{\alpha - \delta'\hat{\beta}^{c,w}}{1-\delta''}, \delta'' = \frac{i+\Delta}{n} \quad (10)$$

Nonetheless, reconfiguration decisions need to be initiated in advance and thus we cannot only rely on the current state of the SLO compliance. Instead we have to estimate the state of the system in period $i+\Delta$. Again, we assume a constant reconfiguration lead time. Equation (*7*) estimates the expected degree of SLO compliance based on the current degree of compliance β and the expected degree of compliance $\Lambda(c_j, \omega_j)$ in the time between i and $i+\Delta$. Similar to equation (*6*), the expected required SLO compliance for the remaining SLA lifetime $\bar{\bar{\gamma}}$ is calculated with the relative elapsed time in $i+\Delta-1$. The goal of the our SLA model is to assess the impact of different configuration decisions. Thus, in equation (9) and (10), the impact of different configurations c for the decision period $i+\Delta$ is determined. The variable $\hat{\beta}^{c,w}$ provides the expected degree of SLA compliance in period $i+\Delta$ for the configuration c if the system faces a workload level w. The corresponding $\hat{\gamma}^{c,w}$ value is calculated in equation (*9*) with the elapsed time of the period $i+\Delta$. If $\hat{\gamma}^{c,w}$ is larger than 100%, the corresponding system configuration would have a fatal impact on the SLA, as the new

SLO target could not be satisfied at all. In this case our dynamic SLA management model assumes a non avertable SLA violation.

5.2. *Online Economic Assessment of the SLA State*

As previously mentioned, the online derivation of a cost efficient and risk minimal operation strategy is a non-trivial problem. Usually, the assessment whether the SLA is fulfilled or violated is binary being determined at the end of the SLA lifetime. However, in order to ensure an economic-aware operation strategy, the value of the different configuration options must be evaluated online in order to derive a feasible operation strategy. Thus, our dynamic SLA management model evaluates the impact of the different configuration decisions in real time on the basis of past performance and on anticipated future user behavior. Furthermore, we designed a scheme to distribute the revenue R and penalty P among the single periods i of the SLA lifetime. This procedure allows us to derive a profit optimal operation strategy which is aware of the risk of violating the contract. By calculating the expected profit of all configuration options, weighted by the probability of facing a workload level w, we can determine the profit optimal and risk adjusted configuration for the decision period $i + \Delta$. The model is designed in a way, that it employs larger configurations (in terms of resource units) for high penalties and high revenues contracts to avoid SLA violations. In contrast, if the resources are expensive or the penalty is low, the dynamic SLA management model chooses very inexpensive operation strategies. Clearly these operation strategies entail a higher risk of SLA violations but more importantly increase the expected profit in return.

The core idea of our SLA management concept is to distribute the total revenue R on the single periods n equally. Furthermore, we define that the full revenue R/n of a period is earned if the system complies with the SLO target α. Nevertheless, as slight performance violations might be reasonable from an economic point of view, we permit that the configurations may only partially fulfill their goal in single periods. As a consequence, these periods only earn a corresponding fraction of the revenue. Conversely, we also allow that a period can earn more than the assigned revenue by overachieving the SLA target. This approach incentivizes the system to build up performance reserves.

$$\hat{R} = R\left(1 - \delta' * \frac{\bar{\bar{\beta}}}{\alpha}\right) \ , \delta' = \frac{i + \Delta - 1}{n} (11)$$

$$r^{c,w} = \frac{\hat{R}}{n - i - \Delta} * \frac{\Lambda(c,w)}{\bar{\bar{\gamma}}} \quad (12)$$

In order to account for the uncertainties in the system, we continuously update the distributable revenue among the remaining periods with the help of the expected SLA compliance $\bar{\bar{\beta}}$ before the decision period. By determining the ratio between the achieved compliance and target and multiplying this value with the elapsed SLA lifetime, we can derive the distributable revenue $\hat{R}$ for the remaining lifetime (*11*). In order to determine the revenue share of the period $i + \Delta$, we divide the remaining revenue by the number of remaining periods *(12)*. In contrast to our original formulation, we define that the full revenue of the period is earned, if the system complies with the online target $\bar{\bar{\gamma}}$. By multiplying this value with the expected SLO compliance $\Lambda(c, w)$ of a configuration c and a workload w, the expected revenue can be calculated.

$$\Pi = -\frac{P}{n}(i + \Delta) \quad (13)$$

$$\pi^{c,w} = -\frac{P}{n}\frac{\left(\hat{\gamma}^{c,w} - \hat{\beta}^{c,w}\right)}{1 - \hat{\gamma}^{c,w}} * i \quad (14)$$

$$p^{c,w} = \begin{cases} 0 & \Pi > \pi^{c,w} \\ \Pi - \pi^{c,w} & \text{else} \end{cases} \quad (15)$$

The valuation of the penalty is different from the revenue valuation. In our penalty valuation model, we assume that the system should achieve a 100% SLO compliance in each period. All deviations are, similar to the revenue valuation, priced by a fractional penalty. However, as the SLO target α accounts for limited performance shortcomings, each period is given a violation allowance of $-P/n$. If the performance allowance is not required it can be saved for later periods. Equation (*13*) defines the performance violation allowance buffer. Equation (*14*) presents the penalty caused by the system up to the current period i including the online SLO impact $\hat{\gamma}^{c,w}$ of the current decision period. Finally, equation (*15*) states that only positive penalties are relevant for the configuration decisions. This prevents, that the system eventually attempts to increase the profit of a configuration by “earning” negative penalties.

In order to derive the profit optimal configuration, the cost of operation for the period of consideration must be included in the decision process. However, cloud resources are commonly priced on an hourly base and hence we use the attributable price of the cloud resource for the period $K = \bar{K}/\bar{\Delta}$ by dividing the cost of an instance hour $\bar{K}$ by the number of periods the instance hour can be used: $\bar{\Delta} = 1/\Delta$. By multiplying this cost factor with the number of resources c, we can derive the cost impact of a configuration.

$$\text{Target Function:} \quad \vec{s'} =< \mathrm{s'}_{\mathrm{i}}, \dots, \mathrm{s'}_{\mathrm{n}} \geq = \Phi(\gamma, \mathrm{j}, \mathrm{n}) = \underset{c_i,\dots,c_t \in C}{\operatorname{argmin}} \sum_{t=j}^{n} c_i \quad (16a)$$

$$\text{Subjected to:} \quad \frac{1}{n-j} \sum_{t=j}^{n} \Lambda(c_t, \omega_t) \geq \gamma \quad (16b)$$

$$\text{Optional:} \quad \Lambda(c_t, \omega_t) > \alpha^{\mathrm{lb}}, \forall_{t=j,\dots,n} \quad (16c)$$

In our resource adaptive operation mode together with our dynamic SLA management model, the provisioning decision in one period affects the online performance goal γ of the remaining period. In order to include this factor into our operation strategy, we forecast the workload process until the end of the SLA lifetime and select an appropriate configuration for all future periods. Equations (*16a-c*) provide the formal definitions of our provisioning model. The target function (*16a*) minimizes the total number of resource during the remaining periods $n - j$ of the SLA lifetime. The first constraint (*16b*) ensures that the resulting configuration vector is able to satisfy the online SLA target γ. The optional constraint (16*c*) allows defining a lower bound SLA target α^{lb}. This way, the dynamic SLA management model will not select a system configuration which provides less performance than the lower bound. This optimization problem can be solved in linear time utilizing a basic heuristic. Starting with the largest configuration $c^{\max}$, we remove one resource unit in each iteration from the period with the lowest relative SLO performance loss. Thus, we require at most $c^{\max} * n$ iterations to solve the optimization problem.

$$\bar{\Gamma}^{\mathrm{c,w}} = K * (|\Theta(\hat{\gamma}^{c,w}, i+1, n)| - |\Theta(\bar{\bar{\gamma}}, i+1, n)|) \quad (17)$$

$$\Gamma^{\mathrm{c,w}} = \begin{cases} \bar{\Gamma}^{\mathrm{c,w}} & \text{else} \\ 0 & \bar{\Gamma}^{\mathrm{c,w}} < 0 \\ P & \bar{\Gamma}^{\mathrm{c,w}} = \emptyset \\ P & \bar{\Gamma}^{\mathrm{c,w}} > R + P \end{cases} \quad (18)$$

By calculating the difference between the original performance goal $\bar{\bar{\gamma}}$ and the new performance goal $\gamma^{w,c}$, we can estimate the operation cost impact on the future periods (*17*). By multiplying the resulting resource difference with the resource cost K, we obtain the economic impact $\bar{\Gamma}^{\mathrm{c,w}}$ of the configuration c for the future periods. Nonetheless, depending on the resulting Γ, case differentiations may become necessary (*18*). The second condition defines that cost saving effects may not be included in the decision problem. The third constraint controls that in those cases where there is no feasible solution (i.e. no operation strategy is able to achieve the SLA), the future impact is priced with the penalty P. Furthermore, we included a feasibility constraint: If the cost of future operation exceeds the revenue penalty span, continuing the operation is economical infeasible.

$$d(c) = \sum_{w \in W} \left(\dot{\mathrm{W}}_{\mathrm{w,i+\Delta}} * (r^{c,w} + p^{c,w} + \Gamma^{\mathrm{c,w}}) \right) + k^c \quad (19)$$

$$s_{i+\Delta} = \underset{c}{\operatorname{argmax}}\ d(c) \quad (20)$$

Similar to the naïve operation strategies, the optimal configuration is selected on the base of the valuation of the current system state. First, for all configurations c and all workload levels w with a positive probability of occurrence, the expected profit is calculated on the base of the expected revenue, penalty, and cost of operation as well as their impact on the future periods (*19*). Since the workload is uncertain, the outcome of each workload level is weighted with its probability. The configuration with the highest expected profit is selected for the decision period $i + \Delta$ (*20*). Nonetheless, due to unforeseen changes and the systematic risk of performance violations, the continuation of the information system operation might be economically infeasible. In case all configurations c have an expected profit of P (i.e. there is no feasible way of fulfilling the contract), the dynamic SLA management model assumes that the SLA cannot be saved at all regardless of any activities. In this case the system switches to the smallest configuration c_1 to reduce the loss by saving the cost of operation.

5.3. *Modifications*

The previous section presented the dynamic SLA management model in detail. Although our model is designed for a risk neutral, rational operation strategy, there are several options to adapt the model to the individual needs of the service provider. The first option is the inclusion of a lower bound SLO threshold. The presented version of the integrative SLA model is designed to switch to the smallest configuration c_1 facing a certain SLA violation. Although this might be an economically rational decision, this behavior may contradict other soft factors such as reputation of the provider. Thus, our first proposed modification is to switch to the predictive operation mode (*5*) once the SLA is ultimately violated. This is attained by setting the SLA target to the lower bound target α^{lb}. This guarantees an acceptable QoS but reduces the cost of operation for the doomed contract.

$$h'(t,c) = \begin{cases} 1 & \sum_{w \in W} \Lambda(w,c) W_{w,t} < \gamma \\ 0 & \text{else} \end{cases} \quad (21)$$

$$h(c) = \sum_{t=\mathrm{i}+\Delta}^{\mathrm{i}+\Delta+\bar{\Delta}-1} h'(t,c) \quad (22)$$

$$\bar{\bar{k}}(c) = K * \left(\sum_{c'=c_1}^{c} \frac{\bar{\Delta}}{h(c')} \right) \quad (23)$$

Today's cloud providers usually charge their resources on an hourly base. However, our model is usually executed in smaller time intervals and determines the expected optimal configuration in accordance with the lead time Δ. As a consequence, our SLA model might add additional resources to the system, which are only required for a limited time (i.e. significantly lower than the length of an instance hour). To compensate this effect, our model can be complemented with an economic feasibility assessment function of the provisioning decision. The core idea is that the cost for each instance hour is shared between the periods using this resource (i.e. the periods $i + \Delta$ to $i + \Delta + \bar{\Delta}$). For example, if one additional resource unit is only required in one period i, this period is charged with the price of the full instance hour $\bar{K}$. If the resource is feasible for all periods of the instance hour, each period is charged with the fractional cost K. Hence we have to determine whether the new configuration is not only feasible for the predicted period, but also for the following periods during the lifetime of the instance hours. Based on the workload forecast distribution, equation (*21*) verifies if the configuration c can sustain the workload in period t. Subsequently, equation (*22*) determines the number of periods, during instance hour lifetime $\bar{\Delta}$, this resource c is accessible. Finally, equation (*23*) defines the fractional resource costs of the current period. This valuation of the different configurations allows the model to identify the optimal configuration and significantly reduces the number of configuration changes.

$$\bar{\bar{P}} = P - K \sum_{t=1}^{i} |c_t| \quad (24)$$

$$\Lambda'(c,w) = \begin{cases} \lambda & \lambda \leq \gamma \\ \gamma & \lambda \geq \gamma \end{cases}, \lambda = \Lambda(c,w) \quad (25)$$

$$\Lambda'(c,w) = \begin{cases} \lambda & \lambda \leq \gamma \\ \lambda + (\lambda - \gamma) * \dfrac{n-i}{n} & \lambda \geq \gamma \end{cases}, \lambda = \Lambda(c,w) \quad (26)$$

The profit of the provider is defined as his revenue less the cost of operation. In particular in scenarios with low penalties and unforeseen high workload, the dynamic SLA management model may decide to give up on the SLA target. Nevertheless, if this happens in the middle of the SLA lifetime, the deliberate violation of the SLA might be an unfavorable option as the previous efforts to comply with the SLA already caused significant cost of operations. To incorporate this aspect into the management strategy, the penalty is continuously increased by the expenditures used for additional resources (*24*). This procedure avoids abortions during the SLA lifetime.

The last modification of our model deals with the fractional revenue determination of each period. In its initial design, the model allows to earn additional revenue by over-fulfilling the SLA target in each period. In some scenarios this behavior might be unwanted. Equation (*25*) provides an alternative option by limiting the maximum revenue to the revenue assigned to the period by redefining the bounds of the degree of SLO compliance function Λ. The second option (*26*) provides a moderate alternative as it continuously shifts from the original version to the modified version (*25*) during the SLA lifetime. These incentivize the SLA model to build up performance reserves at the beginning and switch to a more cost effective operation mode at the end of the SLA lifetime.

6. Evaluation

The evaluation of our dynamic SLA management model, based on real production system workload traces, was carried out in a self-developed test infrastructure (i.e., cloud testbed, elastic application, distributed monitors, and control framework) (Malkowski et al. 2011). In the evaluation, we systemically analyzed the impact of different SLA properties (e.g. revenue, penalty, QoS requirements) and the various configuration options.

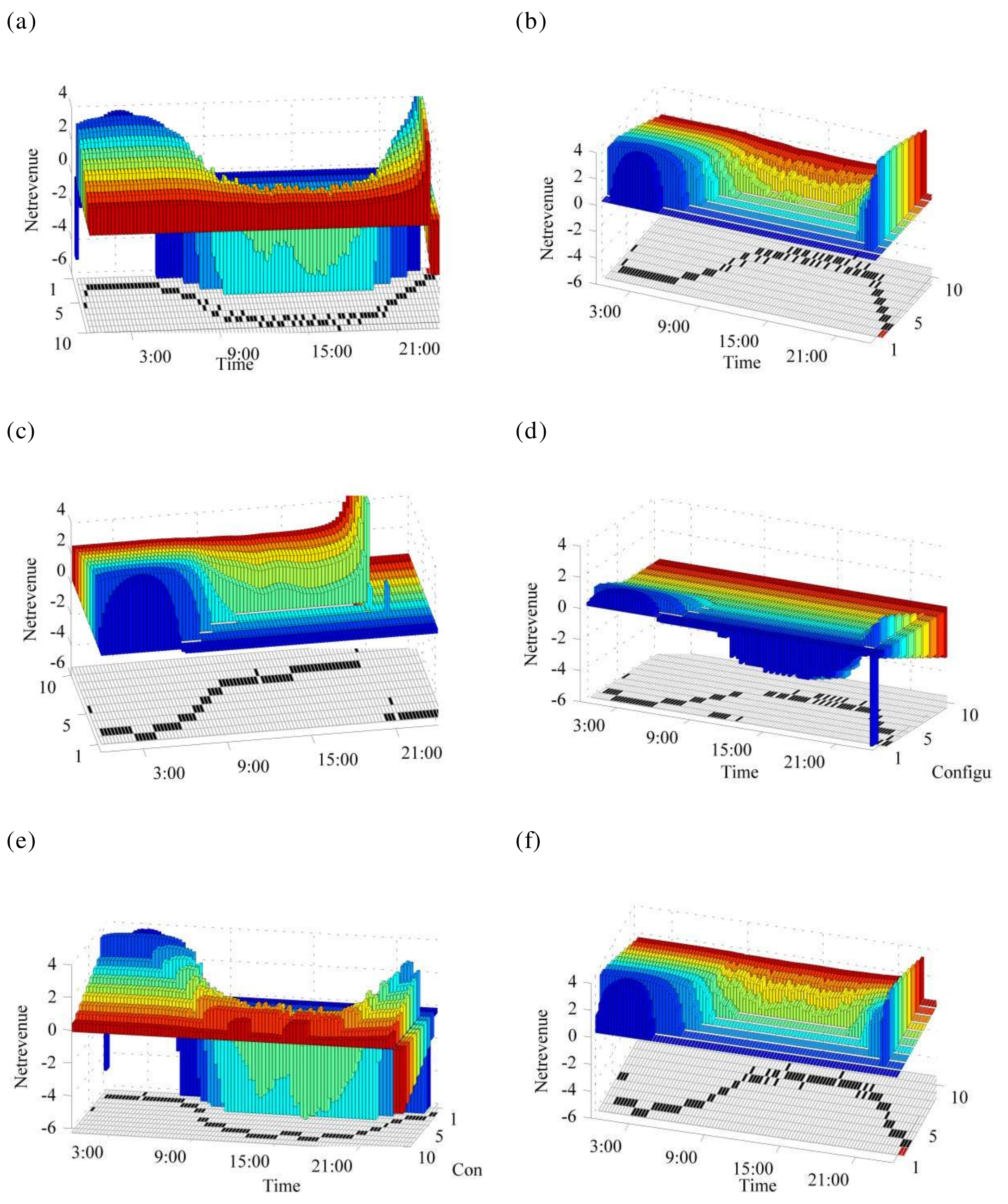


Figure 12: Dynamic SLA Management Model Configuration Options

Figure 12 provides insights into the decision logic of our dynamic SLA management model. While the bars show the expected profit of the different configurations over time, the lower surface depicts the provisioned configuration. In this scenario, we use the workload process of a single day of Wikipedia Germany (Figure 10). The revenue R and the penalty P are set to \$300. The infrastructure cost $\overline{K}$ is \$1.2 and the maximum infrastructure size is limited to $c^{\max} = 10$. The cost of operation is oriented at the common price for mid-range servers of public cloud provider offers. The SLA has a lifetime T of 24 hours. The reconfiguration lead time is set to two periods or 30 minutes respectively. Furthermore the SLA target α is set to 95%. In the following presentation, we use the original model formulation with the lower bound SLA control modification ($\alpha^{\text{lb}} = 70\%$).

Figures (a) and (b) show the characteristics of our base line scenario, where the negative components are trunctuated in figure (b). In this moderate scenario, we can study the behavior of the dynamic SLA management model in a *normal* scenario. Figure (a) shows that during peak times the weaker configurations would violate the SLA and hence are negative. The green mid range configuration exhibits only minor performance limitations and is thus only valued slightly negative. Figure (b) shows the same graphs from another perspective. We can clearly see how the weaker configurations become infeasible during daytime (higher workload) and later become feasible again with the decreasing workload.

In figure (c) and (d), we reduced the SLA target to $\alpha = 80\%$ and deactivated the lower bound control. Due to the relatively high revenue compared to the QoS requirements, the SLA is successfully fulfilled after 85% of the SLA lifetime. In figure (d), we additionally reduced the revenue to $R = 120$ and the penalty to $P = 0$. The SLA specification in this scenario is close to the economic infeasibility as the costs of operation nearly consume the total revenue. However, in absence of the lower bound control, our rational model decides not to provision resources during the expensive peak time. Due to the weak SLA definition, the dynamic SLA management model is still able to meet the SLA target and operate profitably. Nonetheless in this case the system is operated with a minimum amount of resources during the peak time. Evidently, this is no realistic scenario, but facilities the potential of our model.

In figure (e), we activated the node feasibility assessment modification. Evidently, this helps to significantly reduce the fast change of configurations. Furthermore, we can see the profit impact of this modification. As soon as the node is used for multiple periods, the resource costs per period decreases (as the node cost per instance hour is shared over multiple periods) and the specific configurations become more profitable. Finally, figure (f) shows the base line scenario with a longer reconfiguration lead-time ($\Delta = 4$). The dynamic SLA management model tends to provision more resources in the single periods. For instance, we can see this effect in the middle of the SLA lifetime. The configuration c_5 is valuated higher than in the baseline scenario (b). This behavior is caused by the lower confidence of workload forecast and the difficulties to maintain the QoS.

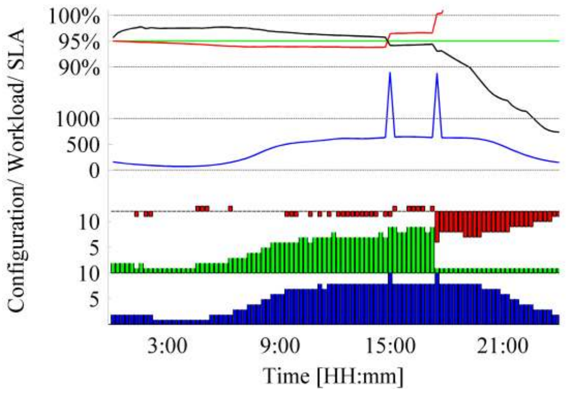


Figure 13: Low-cost Scenario with Disturbance

In the following, we present the behavior of the dynamic SLA management model in the case of an unforeseen, but regularly, recurring disturbance in the workload process. Hence, this disturbance is included in the workload forecast error distribution as a minor outlier. Figure 13 presents a low-cost scenario with a revenue and penalty of 150.

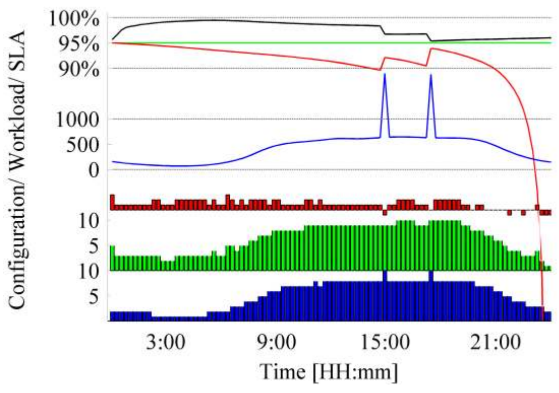


Figure 14: High Revenue Scenario with Disturbance

Figure 14 depicts the high revenue scenario with a revenue and a penalty of 300. The upper red line represents γ and the black line β, whereas the green line is the original performance target α. The blue line below depicts the workload process. The blue graph depicts the optimal configuration determined a priori, the green bar the provisioning of our model and the red the differences between the both.

Both figures illustrate the behavior of our SLA management model. At the beginning in both the scenarios, the SLA is overachieved, which can be seen by the decreasing value of β.

Nevertheless, in the high price scenario more resources are provisioned and β decreases at a fast rate. During both peaks, the online target γ increases. Subsequent to both violations the amount of provisioned resources increases. However, following the second peak, the SLA of the lost-cost scenario is non recoverable and violated, whereas the SLA in the high price scenario can still be met, by overachieving the SLA in the remainder of the SLA lifetime. As our model is configured to behave strictly profit optimizing, it switches the system to a minimal configuration. In summary, this scenario shows the potential of our dynamic SLA management model. While the low cost service is operated with a weak and cost effective infrastructure, the high price service is operated with a more powerful configuration, which can sustain unforeseen workload peaks.

Table 1: Results of the dynamic SLA management model on four weeks of operation

Wikipedia Version	Revenue	Penalty	Alpha	Degree SLA Compliance	SLO Minimum	SLO Maximum	Average Resources	Profit	Successful SLA
German	$300	$-300	0.95	96.3%	73.6%	99.1%	6.07	$125.12	28
	$3000	$-3000	0.95	97.9%	737%	100%	8.69	$2749.73	28
	$300	$-300	0.80	93.5%	69.24%	100%	5.24	$149.09	28
	$300	$-300	static	97.7%	90.2%	100%	10	$12	28
English	$300	$-300	0.95	95.2%	76.0%	98.6%	8.52	$54.62	28
	$3000	$-3000	0.95	96.3%	76.1%	99.6%	9.74	$2719.49	28
	$300	$-300	0.80	91.8%	73.5%	97.8%	7.48	$84.58	28
	$300	$-300	static	96.7%	92.0%	99.6%	10	$12	28
Japanese	$300	$-300	0.95	95.9%	83.8%	100%	6.40	$115.67	28
	$3000	$-3000	0.95	97.4%	77.5%	100%	8.53	$143.61	28
	$300	$-300	0.80	92.8%	71.7%	99.6%	5.43	$2843.61	28
	$300	$-300	static	98.5%	100%	100%	10	$12	28

Table 1 shows a broad range of results of the dynamic SLA management model with different configurations on different workload traces. As the German, English and Japanese Workload strongly differ in their average workload levels, all processes have been normalized such that the maximum workload does not exceed 1000 requests per second. As benchmark, we included the results of a static operation. The cost for one cloud resource per hour was set to $1.20 and the delay time was set to 30 minutes. The first scenario is balanced between the economic parameters and the performance goal. The second scenario is a high revenue scenario and the third has a low performance target. The table presents the results of 28 days from the mid of February to the mid of March in 2011. Our model successfully fulfills the SLA in all scenarios in every run. In particular in the low cost scenarios, our model helps to significantly reduce the cost of operation. The lower SLO minimum originates from the switch to the lower bound control on fulfillment of the SLA. In the high revenue scenario, the SLA management model provisions significantly more resources to the system, as the costs of operation are minor. The saving on the English trace is significantly lower, as the system traces have a higher base load.

7. Conclusion

In this paper we presented a novel dynamic SLA management model for the sustainable and efficient operation of elastic information systems in cloud environments. Based on a system performance model and workload forecast model, our new management concept enables highly efficient operation modes. Our model extends the current state of the art by not only managing the system based on the QoS specifications of the SLA, but also according to economic parameters, such as the revenue, penalties, and the cost of cloud resources. In contrast to (most) other SLA management concepts, our model does not necessarily aim to comply with the SLA at all times, but may instead choose a strategy that maximizes profit in the long-term. Furthermore, the dynamic character of our SLA model allows the flexible adaptation of performance goals at runtime, therefore mitigating the risk of performance violations. In summary our model bridges the gap between cloud technology and the economic value of a service. It provides a methodology to automatically manage service offers according to their value and is, therefore, particularly useful for services offered in different QoS classes with different price models.

Conceptually, our model is an effort with the aim of integrating all aspects of a Service Level Agreements (e.g., monitoring metrics and economic parameters) with runtime monitoring data. To accommodate the heterogeneity of enterprise information systems, our model is designed modularly, enabling different configurations according to the properties of the system. During operation, our model systematically processes and analyzes all factors of influence such as the performance and workload data as well as the current SLA state. In particular, for critical high-price services, our novel SLA model has proven to outperform other basic concepts. While basic controllers are conceptually able to provide any cost-effective operation mode, our model is able to adapt the operation strategy automatically based on the true economic value of the system. Thus, depending on the economic situation, it mitigates the risk of performance violations compared to rigorous cost-driven adaptive operation modes. We showed that the our model allows flexible information system operation with up to a 40 percent lower cost of operation compared to static operation modes and a significantly lower risk of SLA violations compared to other adaptive resource management systems.

We planned various extensions for our SLA management model. Recently, Amazon Web Services LLC introduced a new spot pricing scheme for cloud resources (Amazon 2011a). In our future work, we plan to extend our work to reflect these recent developments by supporting dynamic resource prices. More concretely, this enables service providers to operate the system in times with lower resource cost more risk-aware and take higher operational risks during peak time. Furthermore, we intend to extend our model to manage the resource requirements of multiple competitive systems. Based on the current SLA state of different services and their economic parameters, this extension should optimally allocate resources to the different services. For instance, if we have two services, each requiring 4 nodes, and 9 nodes in total, the management model should automatically assign the residual node to the service with the higher economic risk. In its current implementation the management model is risk neutral, aiming to maximize the expected profit. In our future work, we plan to integrate an individual risk function in order to let the operator self-select his risk-affinity level. Furthermore, the revenue and penalty parameters have only been arbitrarily specified. In our future work, we plan to use the dynamic management concept to estimate the expected cost of operation and the risk of an SLA violation and thus determine the optimal and risk adjusted price and penalty combination for a service.

8. References

Aib, I. & Boutaba, R., 2007. On Leveraging Policy-Based Management for Maximizing Business Profit. *IEEE Transactions on Network and Service Management*, 4(3), pp.25-39. Available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4489642 [Accessed April 30, 2011].

Amazon, 2011a. Amazon EC2 Spot Instances. Available at: http://aws.amazon.com/ec2/spot-instances/ [Accessed April 30, 2011].

Amazon, 2011b. Amazon Simple Queue Service. Available at: http://aws.amazon.com/sqs/ [Accessed April 30, 2011].

Ardagna, D., Trubian, M. & Zhang, L., 2007. SLA based resource allocation policies in autonomic environments. *Journal of Parallel and Distributed Computing*, 67(3), pp.259-270. Available at: http://dx.doi.org/10.1016/j.jpdc.2006.10.006 [Accessed July 16, 2010].

Bailey, M., 2009. *The Economics of Virtualization: Moving Toward an Application-Based Cost Model*, Available at: http://www.vmware.com/files/pdf/Virtualization-application-based-cost-model-WP-EN.pdf.

Buco, M.J. et al., 2004. Utility computing SLA management based upon business objectives. *IBM Systems Journal*, 43(1), pp.159-178. Available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5386770 [Accessed April 30, 2011].

Buyya, R. et al., 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6), pp.599-616. Available at: http://dx.doi.org/10.1016/j.future.2008.12.001 [Accessed April 30, 2011].

Chandra, A., Gong, W. & Shenoy, P., 2003. Dynamic Resource Allocation for Shared Data Centers. *Quality of Service — IWQoS 2003*, Volume 270, pp.381-398. Available at: http://www.springerlink.com/content/h56r57014u707466.

Cohen, I. et al., 2004. Correlating instrumentation data to system states: a building block for automated diagnosis and control. *Operating Systems Design and Implementation*, p.16. Available at: http://portal.acm.org/citation.cfm?id=1251270 [Accessed September 15, 2010].

Feitelson, D.G., 2011. *Workload Characterization and Modeling Book*, Available at: http://www.cs.huji.ac.il/~feit/wlmod/.

Gartner, 2010. Gartner Says Energy-Related Costs Account for Approximately 12 Percent of Overall Data Center Expenditures. Available at: http://www.gartner.com/it/page.jsp?id=1442113 [Accessed April 30, 2011].

Gmach, D. et al., 2009. Resource pool management: Reactive versus proactive or let's be friends. *Computer Networks*, 53(17), pp.2905-2922. Available at: http://dx.doi.org/10.1016/j.comnet.2009.08.011.

Gmach, D. et al., 2007. Workload Analysis and Demand Prediction of Enterprise Data Center Applications. *2007 IEEE 10th International Symposium on Workload Characterization*, pp.171-180. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4362193.

Goolsby, K., 2007. How to rebound from a failed outsourcing relationship. *Outsourcing Journal*. Available at: www.outsourcing-journal.com/jan2007-rebound.html.

Hasselmeyer, P. et al., 2006. Towards Autonomous Brokered SLA Negotiation. *Exploiting the Knowledge Economy Issues Applications Case Studies*, 3. Available at: http://www.mendeley.com/research/towards-autonomous-brokered-sla-negotiation/ [Accessed April 30, 2011].

Hedwig, M., Malkowski, S. & Neumann, D., 2009. Taming Energy Costs of Large Enterprise Systems Through Adaptive Provisioning. In *ICIS '09: Proceedings of the Eight IEEE/ACIS International Conference on Computer and Information Science*. Phoenix, AZ, USA: IEEE Computer Society.

Hedwig, M., Malkowski, S. & Neumann, D., 2010. *Towards Autonomic Cost-Aware Allocation of Cloud Resources*, Available at: http://aisel.aisnet.org/icis2010_submissions/180 [Accessed January 31, 2011].

Koller, B. & Schubert, L., 2007. Towards autonomous SLA management using a proxy-like approach. *International Journal of Multiagent and Grid Systems*, 3(3), pp.313-325. Available at: http://www.mendeley.com/research/towards-autonomous-sla-management-using-a-proxylike-approach/ [Accessed April 30, 2011].

Koomey, J.G., 2007. Estimating total power consumption by servers in the U.S. and the world. *World*. Available at: http://www.greenbiz.com/research/report/2007/09/12/estimating-total-power-consumption-servers-us-and-world.

Lassettre, E. et al., 2003. Dynamic Surge Protection: An Approach to Handling Unexpected Workload Surges with Resource Actions that Have Lead Times. *Self-Managing Distributed Systems*, pp.33-46.

Lim, H.C., Babu, S. & Chase, J.S., 2010. Automated control for elastic storage. In *Proceeding of the 7th international conference on Autonomic computing*. New York, NY, USA: ACM, pp. 1-10. Available at: http://doi.acm.org/10.1145/1809049.1809051.

Malkowski, S. et al., 2010. CloudXplor: a tool for configuration planning in clouds based on empirical data. *Symposium on Applied Computing*, pp.391-398. Available at: http://portal.acm.org/citation.cfm?id=1774172 [Accessed September 14, 2010].

Malkowski, S. et al., 2011. Automated control for elastic n-tier workloads based on empirical modeling. In *Proceedings of the 8th ACM international conference on Autonomic computing - ICAC '11*. New York, New York, USA: ACM Press, p. 131. Available at: http://dl.acm.org/citation.cfm?id=1998582.1998604 [Accessed August 29, 2011].

Mituzas, D., 2011. Wikistats. Available at: http://dammit.lt/wikistats/ [Accessed August 30, 2011].

Padala, P. et al., 2009. Automated control of multiple virtualized resources. *Proceedings of the fourth ACM european conference on Computer systems - EuroSys '09*, p.13. Available at: http://portal.acm.org/citation.cfm?doid=1519065.1519068.

Powers, R., Goldszmidt, M. & Cohen, I., 2005. Short term performance forecasting in enterprise systems. *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining - KDD '05*, p.801. Available at: http://portal.acm.org/citation.cfm?doid=1081870.1081976.

Raimondi, F., Skene, J. & Emmerich, W., 2008. *Efficient online monitoring of web-service SLAs*, New York, New York, USA: ACM Press. Available at: http://dl.acm.org/citation.cfm?id=1453101.1453125 [Accessed June 23, 2011].

RightScale, 2011. RightScale. Available at: http://www.rightscale.com/ [Accessed April 30, 2011].

Sahai, A., Durante, A. & Machiraju, V., 2001. Towards Automated SLA Management for Web Services. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.7978 [Accessed April 30, 2011].

Sahni, M. & Tan, L.F., 2011. *APEJ CIOs will be the Rising Stars in the Corporate Hierarchy as they Drive ROI-Led Transformation Initiatives in 2011, says IDC*, Available at: http://www.idc.com/AP/pressrelease.jsp?containerId=prSG22698011.

Schulz, G., 2009. *The Green and Virtual Data Center*, Boston, MA, USA: Auerbach Publications.

Short, J.E., Bohn, R.E. & Baru, C., 2011. *How Much Information - Report on Enterprise Server Information*, Available at: How Much Information? 2010.

Urgaonkar, B. et al., 2008. Agile dynamic provisioning of multi-tier Internet applications. *ACM Transactions on Autonomous and Adaptive Systems*, 3(1), pp.1-39. Available at: http://portal.acm.org/citation.cfm?doid=1342171.1342172.

Velte, T., Velte, A. & Elsenpeter, R.C., 2009. *Green IT: Reduce Your Information System's Environmental Impact While Adding to the Bottom Line*, New York, NY, USA: McGraw-Hill, Inc.

Yeo, C.S. & Buyya, R., 2007. Integrated Risk Analysis for a Commercial Computing Service. Available at: http://www.computer.org/portal/web/csdl/doi/10.1109/IPDPS.2007.370241 [Accessed April 30, 2011].

CHAPTER V

DATACENTER INVESTMENT SUPPORT SYSTEM (DAISY)

This chapter is a revised version of the paper:

Hedwig, Markus; Malkowski, Simon; Bodenstein, Christian and Neumann, Dirk, "Datacenter Investment Support System (DAISY)" (2010), 43rd Hawaii International Conference on System Sciences Proceedings, Copyright 2010 IEEE, Digital Object Identifier 10.1109/HICSS.2010.129.

Abstract

While the cloud computing paradigm provides the technological foundation for previously unmatched efficiency in information systems, its economic implications are complex and significant. Bringing the benefits of this state-of-the-art technology to corporate datacenters requires a systematic cost analysis of current best-practice in conjunction with the new possibilities. We address this challenge and present a novel resource management model based on marginal cost of computing. DAISY (Datacenter Investment Support System) aids in deriving the optimal investment strategy for large enterprise computing infrastructures taking into account all buy and rental options. Our model-based framework is evaluated through the analysis of different time discrete and continuous scenarios. Our main contribution is the thorough economic analysis of the dependencies between static and adaptive modes of operation for modern computing infrastructure. Our results clearly show the ambivalent economic character of computing as a commodity and that further research is necessary to guarantee sustainable economic decisions.

Keywords: Datacenter Management, Cloud Computing, Infrastructure Investments, Infrastructure as a Service, Workload

1. Introduction

Nowadays, the operation of IT systems has become a highly significant cost factor for modern enterprises. While the ever increasing complexity of business processes causes a rapid growth of computer resource demand, profit margins continue to shrink under market pressure. Consequently, cost reduction is a top priority task for IT decision makers. Recent technological advances such as cloud computing have enabled innovative modes of operation as an alternative to classical computing paradigms. In contrast to owning large IT infrastructures, computing resources are considered a commodity that may be flexibly rented according to actual demand. Despite the premise of non-idling computing resources through adaptive system sizing (Infrastructure-as-a-Service(Assuncao et al. 2008; Konstantinou et al. 2009)) simple calculation reveals that outsourcing entire infrastructures cannot be sustainable in the long-term. On the other hand, static infrastructures sizes result in periodic underutilization by definition, given volatile workload processes. This insight dictates that in most cases the cost optimal mode of operation has to incorporate static and adaptive elements in order to harness the benefits of both worlds. However, there is currently little or no support for making model based decisions on efficient modes of operation for computing infrastructure.

This paper explicitly addresses the aforementioned challenge and provides a novel resource management model based on marginal cost of computing. We introduce a theoretic model utilizing state-of-the-art concepts from the field of service science. More concretely, this work presents the Datacenter Investment Support System (DAISY), a framework to assess and interpret the cost involved in the operation of large enterprise systems. DAISY may be used as both a Decision Support System (DSS) for investment and a tool for examining the cost impact of policy changes.

The main contribution of this paper is the insights that DAISY provides into the complex dependencies between static and adaptive infrastructure modes of operation for modern computing infrastructure. We systematically evaluate the economic benefits of computing resources as a service and corroborate the growing awareness of the complexity that accompany the new technical possibilities. This work provides model-based decision support for choosing optimal infrastructure size along with an intuitive formalization.

This work is structured as follows. Section 2 discusses related work in the field of enterprise computing and IT investment. In Section 3 we introduce the formal foundation of the DAISY framework and discuss suitable interpretation approaches. Section 4 provides illustrative examples for fixed point and continuous model application scenarios. In Section 5 we restate DAISY in a discrete problem setting and derive its practical implications for economical decisions on optimal investment strategies. Section 6 concludes the paper with a brief summary and an outlook into current and future work.

2. Related Work

DAISY combines both economic as well as computer science aspects in an interdisciplinary framework. Hence, this section is arranged accordingly. We discuss related research topics from the field of IT outsourcing, economically motivated resource management concepts, and computer science models that provide the technological foundation for DAISY.

2.1. IT Outsourcing

DAISY suggests minimizing the costs of operation by renting peak-demand resource from a third party provider using the IaaS-approach. The idea of renting computing resources (Lee et al. 2003) originated in the 1960s, and it was commonly applied when computing requirements were beyond the financial capabilities of single enterprises. Today, with advances in technology, allowing the transfer of data online, as well as decreasing prices of IT equipment, the scope and intensity of IT resource rental has changed. Several models have previously been developed to assess the feasibility and implications of IT outsourcing in different contexts (Gilley & Rasheed 2000).

In general, IT outsourcing is to decouple tasks to third party providers which are actually able to conduct the tasks more efficiently by economy of scale. However, since their beginnings IT resources have established themselves as standard products. As a result, an economy of scale effect is virtually non-existent (Gray 2008). From this point-of-view, outsourcing IT systems and employing IaaS solutions, potentially results in economically infeasible options. Risch et al. (Risch & Altmann 2008) prove this assumption is empirically by examining the charring models of cloud providers as well as determining the operation costs of owned infrastructure. This comparison shows that established enterprises can usually provide their computing demand themselves at lower costs. In contrast to these previous research efforts, DAISY adopts a fundamentally different approach. Instead of simple binary buy or rental decisions, our model is based on a hybrid approach (i.e., composition of purchasing and renting resources).

The outsourcing aspect and its impact on the cost of operation is only one side of the coin. Focusing on the optimal size of IT infrastructures, Jagannathan et al. derived an optimal infrastructure size based on the analysis of their customer contracts and past user behavior evaluations (Jagannathan et al. 2003). As a result, they were able to forecast system utilization and derive a cost optimal investment plan for their system architecture. However, the authors solely optimized their investment decisions to satisfy peak demands. Despite similar goals, DAISY specifically addresses efficient investment strategies through incorporating IaaS concepts. Therefore, our model does not require accurate resource demand predictions because unforeseen shifts in the resource demand can be immediately compensated with IaaS resources.

2.2. System Management

Resource management systems (RMS's), for the optimal resource utilization in grids and clouds, are an extensively discussed research topic. Several theoretic models have been developed to address utilization optimization, though the scope of the different approaches strongly differs. While most RMS's have strong technical focus, there are also approaches that specifically incorporate economic considerations (Krauter et al. 2002). As RMS's optimize available resources online, they do not take future development into account. Therefore, their methodologies, do not directly apply to the problem setting of DAISY. The scope of DAISY is to find a total cost of operation minimal strategy by explicitly modifying the owned hardware infrastructure.

The economical reasonable inclusion of remote resources in production systems has been made possible through technological advancements made in recent years. Service-oriented architectures constitute the first feasible software design concept to efficiently distribute applications over several servers. Enough abstraction from the hardware is provided to dynamically add and remove resources from the system (Channabasavaiah et al. 2004; Assuncao et al. 2008). Cloud and grid computing contributed to the standardization and availability of such remote resources (Buyya et al. 2008). Virtualization provides the necessary hardware abstraction and enables fast deployment and migration of encapsulated software systems. Furthermore, virtualization enables the fine-grained management of server resources(Padala et al. 2007). For instance, Chen et al. developed strategies to reduce the cost of operation in resource competitive environments by optimally allocating server resource to applications according to their demand(Chen et al. 2005). This enables the operation of multiple systems with fewer hardware resources and reduces the total cost of operation. However, in contrast to DAISY, this concept is designed to work on a fixed installed base.

The idea of automatically scaling a system by including IaaS has been previously implemented(Ragusa et al. 2008). In this work the authors continuously increase the workload on a server beyond its performance capabilities. Their load balancer continuously monitors the server and activates remote resources upon detecting a potential system overload. Although they showed the technical practicability of using IaaS to complement owned resources, economic aspects, the main focus of DAISY, were not explicitly addressed.

Several works have previously investigated various business models in the field of grid and cloud computing as well as the current state of the industry (Altmann et al. 2007). Although there are some early efforts to complement owned systems with IaaS models, there is no actual implementation of these concepts yet. To date, remote resources have been primarily used to perform isolated computing tasks.

3. The DAISY Framework

This section introduces the time continuous formulation of the DAISY framework. This is a difficult task due to the different factors of influence on the total cost of operation. By explicitly formalizing these various factors, DAISY provides an intuitive view of the cost dynamics of IT architectures. Subsection 3.1 elaborates on the resource demand inherent in modern infrastructures. Subsection 3.2 formally describes DAISY, and the formal model is interpreted in Subsection 3.3. Finally, Section 3.4 concludes the framework with a mathematical derivation of an optimal investment plan.

3.1. Resource Demand Function

End user IT systems generally show strong seasonal variations in their utilization over time, which may be due to factors such as time of day, day of week, and other periodic dependencies in user behavior.

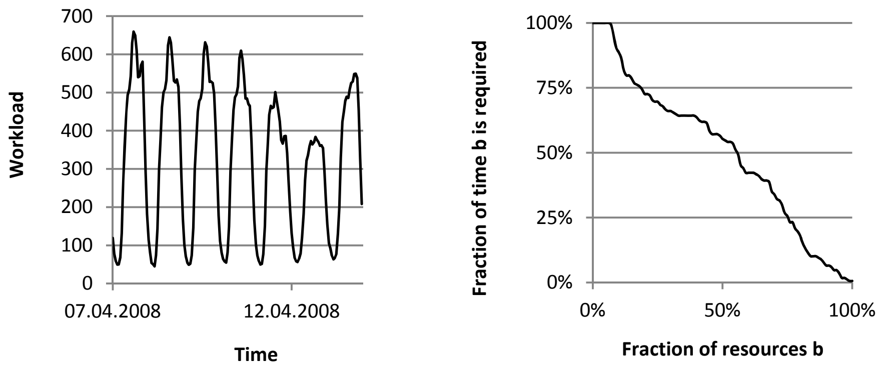


Figure 1: Resource demand and utilization

Figure 1 (left) illustrates this behavior. The graph shows the average requests rate of Wikipedia Germany (Mituzas 2011). Evidently there are strong variations in the usage intensity.

One way to minimize the cost of operation is to dynamically adapt the hardware configuration to the current demand. To provide this capability, various models have been developed that characterize the performance behavior of IT Systems (Malkowski et al. 2007; Urgaonkar et al. 2008; Cohen et al. 2004; Anon n.d.). Load balancers incorporating this technique are able to concentrate the workload on a minimal set of machines as well as to add and remove servers from the infrastructure during operation. This provides a high utilization of the active machines and reduces the cost of operation since unused machines can be switched off. Nonetheless, this approach does not reduce the fixed cost (e.g., amortization costs or rack space charges).

The concept of DAISY is broader. It extends the concept of higher utilization degrees through the total cost of ownership (TCO) including the costs of unutilized servers. DAISY optimizes the owned infrastructure size based on TCO. The latter are based on resource utilization and other cost factors including investment and operation costs as well as the charges of renting remote resources. In order to balance these different cost factors, DAISY introduces a marginal resource utilization function. This function describes the percentage of computing infrastructure that is required in each time interval. As aforementioned, the resource utilization may be highly volatile over time. To derive a representative utilization distribution, longer timeframes (in the magnitude of weeks) need to be inspected. Based on the marginal utilization of the system and the cost functions for owned and rented resources, the cost for providing the service with a distinct infrastructure size can be calculated.

If $w(t)$ describes the level of workload over time, the function $b = i(w) \rightarrow [0\%, 100\%]$ determines the fraction of hardware resource needed to satisfy the demand w. $b = i(w(t))$ determines the fraction of resources b needed to satisfy the demand in t. In this notation the resource demand is assumed completely divisible. This relaxation will be removed later on in the extensions of the model.

In order to determine the marginal utilization distribution, the server utilization has to be examined for time intervals. This is required in order to capture a sufficient amount of factors of influence. The length of the time interval is defined as a.

Share of resource required in t: $$q(t, b) = 1_{i(w(t))>b} \rightarrow \{0,1\} \quad (1)$$

Marginal resource utilization: $$d'(t, b) = \frac{1}{a}\int_{t-\frac{a}{2}}^{t+\frac{a}{2}} q(t, b)dt \quad (2)$$

3.2. Formal Introduction of DAISY

This subsection presents the times continuous version of DAISY. We define x as the percentage of owned infrastructure based on the infrastructure size necessary to satisfy the peak demand. Hence $(1 - x)$ is the percentage of IaaS resources needed to supply the required computing power. Based on the marginal utilization function $d'(t, b)$, the utilization of owned infrastructure as well as the amount of additionally required resources to be rented can be derived.

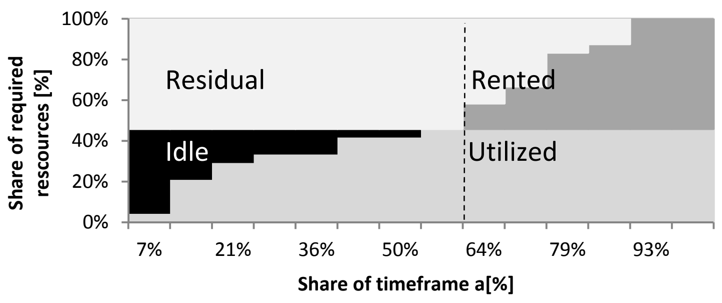


Figure 2: Resource demand composition

Figure 2 illustrates this relationship. For any given point in time, the owned infrastructure is able to satisfy workloads of up to 45% of the peak demand. This is sufficient to provide services for 57% of the investigated time interval. To maintain the service quality for the remainder of the period, additional resources need to be rented. Based on the marginal resource utilization the amount of resources, which is need to be rented, the amount of utilized owned resources, as well as the amount of idle owned resources can be determined. With the help of $d'(t,b)$, each quadrant can be formally defined:

Owned utilized infrastructure: $$d_o(t,x) = \int_0^x d'^{(t,b)} db$$

Owned idle infrastructure: $$d_i(t,x) = 1 - d_o(t,x) \qquad (3)$$

Rented resources: $$d_r(t,x) = \int_x^1 d'(t,b)\, db$$

Residual: $$d_e(t,x) = (1-x) - d_r(t,x)$$

Each quadrant is associated with a cost function per resource unit, representing the total cost of ownership on an hourly basis. For IaaS this is equivalent with the hourly charge; whereas, for owned infrastructure a cost factor for the idle and for the utilized states are required. This is necessary to account for management concepts, which switch-off idle machines in order to save energy, for instance. This cost function also includes the fixed cost (e.g., amortization). For each quadrant a corresponding cost function $c_o(t)$, $c_i(t)$, and $c_r(t)$ can be introduced. To allow variations in the cost factors over time, each function depends on t. The quadrant residual represents hypothetical resources, required for the complete formulation of DAISY. They can be interpreted as computing resources that are rented to other clients in idle periods. As a result, $c_u(t) = 0$.

Resource Demand Matrix: $$\mathrm{D(t)} = \begin{pmatrix} \mathrm{d_o(t, x(t))} & \mathrm{d_r(t, x(t))} \\ \mathrm{d_i(t, x(t))} & \mathrm{d_e(t, x(t))} \end{pmatrix} \quad (4)$$

Cost Matrix: $$\mathrm{C(t)} = \begin{pmatrix} \mathrm{c_o(t)} & \mathrm{c_r(t)} \\ \mathrm{c_i(t)} & \mathrm{c_e(t)} \end{pmatrix} \quad (5)$$

In order to achieve a compact formulation, the following two-by-two matrices are defined, demand matrix D (4) and a cost matrix C (5). Each entry represents one quadrant. The dependence on t is required to model the changes in the infrastructure size x. The relative infrastructure size x is based on the maximum resource demand of all periods.

3.3. *Interpretation of the DAISY framework*

This subsection discusses suitable interpretations of the DAISY framework. In the following presentation the function parameters are omitted due to legibility constraints.

Total Cost Matrix: $$E = C^T D = \begin{pmatrix} d_o c_o + \mathrm{d_i} c_i & d_r c_o + c_i d_e \\ d_o c_r & d_\mathrm{r} c_\mathrm{r} \end{pmatrix} \quad (6)$$

By multiplying the transposed cost matrix with the demand matrix the total cost matrix (6) is constructed.

Table 1: Matrix *E*

Operation cost	Avoided charges through renting
Cost of renting owned servers	Renting charges

The interpretation of the cost overview matrix E is given in Table 1. The sum of the main diagonal is the total cost of operation. The secondary diagonal shows the avoided cost through renting resources as well as additional cost of migrating completely to IaaS. The sum of the first row is the total cost of operation in the case where the complete infrastructure is owned; whereas the sum of the second row represents the total cost of operation in case of sole use of IaaS. Therefore, this simple matrix provides an intuitive overview of the cost structure.

Further insights can be gained from determining the derivative of E with respect to x. If an increase in the number of servers is planed, the main diagonal of $\nabla_x E$ shows the magnitude of change in the operation costs as well the magnitude of saved rental charges. Especially the changes in the operation cost can be very helpful to determine the economic feasibility of hardware investments. The sum of the diagonal defines the change of the total cost of operation.

Similarly, the derivative of E with respect to t provides valuable insights. The reflection of $\nabla_t E$ shows the cost evolution and aids in understanding the impact of the different pricing factors as well as the behavior of future costs. Both derivatives are later used to find the TCO-minimal infrastructure sizes by finding the global minimum.

3.4. Derivation of the Optimal IT System Size

After defining a compact term for the TCO of a system, the optimal architecture size can be determined. To derive intuitively interpretable results and to provide a convenient representation, either time or cost functions are fixed in the following propositions. The propositions itself can be used to support management decisions. The first proposition derives the TCO-optimal size of an infrastructure for a fixed point in time.

Proposition 1: *The TCO optimal size of owned infrastructure x^* for a fixed point in time $\bar{t}$ is defined as the point in time where the fraction of the idle cost and the sum of the idle and rent cost minus the operation cost are equal to the marginal resource utilization $d'(x^*, \bar{t}) = \frac{c_i}{c_i + c_r - c_o}$.*

The second proposition states that for constant cost functions, the TCO optimal size of the infrastructure solely depends on the characteristics of the workload over time.

Proposition 2: *If all cost functions $c_o(t)$, $c_i(t)$, and $c_r(t)$ are constant (i.e., $k = \frac{c_i}{c_i + c_l - c_o}$ is constant), then the TCO optimal size of an IT system solely depends on the shape and magnitude of the marginal workload distribution over time.*

In the following sections, the two propositions are discussed in detail by providing various examples. Proofs of both propositions have been moved to the Appendix.

4. Continuous Model Application

After the introduction of DAISY in Section 3, this section presents the application of the model.

4.1. Time-Independent Evaluation

The first example analyzes the cost structure of an IT system for a fixed point in time in order to intuitively present the capabilities of DAISY. All cost functions are constant and the workload distribution is static. Depending on the shape of the workload distribution the optimal infrastructure size varies. This section investigates different distributions approximated with basic mathematical functions. More specifically, we investigate the impact of a linear, a heavy-tailed, and a short-tailed marginal resource utilization function.

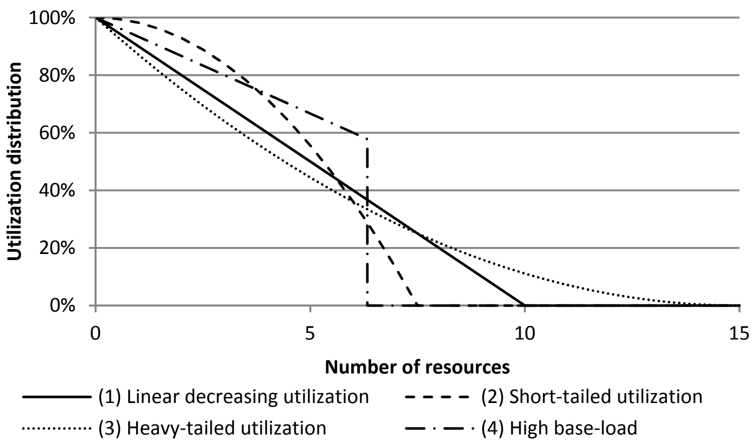


Figure 3: Server utilization scenarios

Figure 3 depicts the distribution characteristics. In order to be able to derive economic conclusions, we define infrastructures for each scenario. To allow for a comparison between the different characteristics, the resource requirements are identical for all scenarios. However, the peak demands of the four distributions differ. All scenarios require five computing units on average, and the peak resource demand for the scenarios are $s_1 = 10$, $s_2 = 7.5$, and $s_3 = 15$. Scenario 4, with a high base load, does not have to be explicitly investigated because it can be reduced to scenario 1.

$$d'_1(t,b) = 1 - b \tag{7a}$$

$$d'_2(t,b) = 1 - \mathrm{b}^2 \tag{7b}$$

$$d'_3(t,b) = 1 - 2\mathrm{b} + \mathrm{b}^2 \tag{7c}$$

The marginal distribution functions d'_1, d'_2, and d'_3 are given in Equation (7b). Using the Propositions one and two we can derive the cost minimal infrastructure size for the three scenarios.

$$s_1^* = (1-k)s_1 \tag{8a}$$

$$s_2^* = \sqrt{(1-k)}s_2 \tag{8b}$$

$$s_3^* = \left(1-\sqrt{k}\right)s_3 \tag{8c}$$

By setting $k = \frac{c_i}{c_i + c_l - c_o}$, we get the TCO-optimal infrastructure sizes s_1^*, s_2^*, and s_3^* (8a). The next step is to include the cost factors for the computing resources. In this case study, they have been approximated based on current market prices. The cost factors can be set to arbitrary values. We assume that the installation of a server including all infrastructure requirements accounts for $6.000. Assuming an expected life-time of 2-years for the equipment, this leads to $0.34 per hour. This fixed cost share is added to the cost of operation. Furthermore, we assume that the operation costs of a server are $0.25 and the idle costs are $0.05. For renting the same server using IaaS resources, $1.00 is charged per hour. As a result, our cost factors are: c_0 = $0.59, c_i = $0.39 and c_l = $1.00. The cost factors are estimated on the base of current market prices, whereby the price per instant hour is based on the cost model of EC2 for a large node. The price for the server is based on the market value of such a node. Operation and idle cost reflect the energy intake of these systems, including the energy demand of the datacenter infrastructure (e.g., for cooling and network equipment).

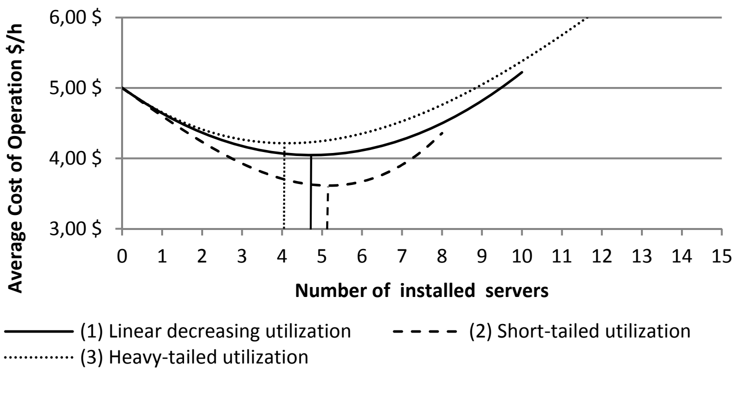


Figure 4: Avg. cost if x servers are installed

Figure 4 shows the TCO on an hourly basis depending on the number of owned servers. The fraction of owned infrastructure x is transformed to the actual number of servers in the graphs of this section. As all three scenarios require five resource units, the costs of operating the systems with IaaS are identical. The results are summarized in Table 2. Overall the combined use of owned and rented resources provides the lowest cost of operation. Heavy-tailed utilization distributions generally require large peak-demand performance. Solely satisfying the latter with owned infrastructures is very costly.

Table 2: Avg. cost of operation

	(1)	(2)	(3)
IaaS	$5.00		
DAISY	$4.04	$3.61	$4.21
Owned IS	$5.22	$4.35	$7.45
s_i^*	4.74	5.16	4.12

Hence, DAISY delivers the strongest results in such scenarios. In contrast, if we consider the short-tailed distribution, the costs of operation are more or less identical. Overall the length of the marginal resource distribution tail determines the feasibility of a mixed operation.

4.2. Analysis of System Management Scenarios

This subsection presents how to take advantage of DAISY in the assessment of management policies. Assume the current state in an IT-System is the static operation of the system. In order to reduce cost the potential of including IaaS in the system has to be evaluated. The operator has three options. The first is to leave idle servers online, resulting in an idle cost equal to the operation costs $c_o = c_i = \$0.59$. The second is to switch servers of, resulting in the idle cost of the last section $c_i = 0.39$. The third alternative is to rent these resources to other companies at a charge of $0.30, resulting total idle cost of $c_i = \$0.59 - \$0.30 = \$0.29$.

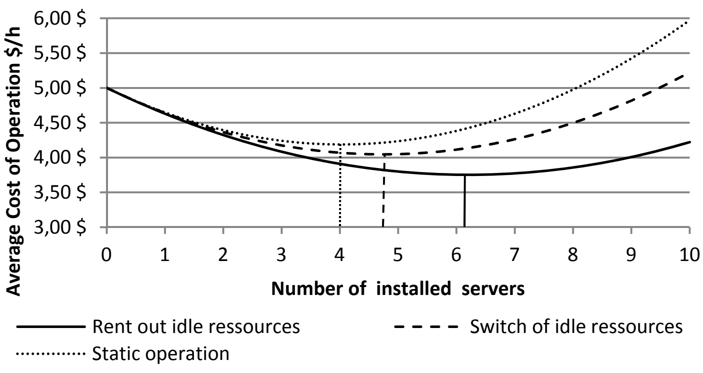


Figure 5: Impact of idle resource management

Figure 5 shows the average cost of operation related to the number of installed servers. Applying cost diminishing management concepts reduces the average cost of operation. More surprisingly, efforts to reduce the idle costs increase the economically feasible infrastructure size. However, recalling Proposition 2 and considering that k is the denominator of the optimum criteria, the result speaks for itself.

4.3. Time-Dependant Evaluations

This subsection presents the application of DAISY in a dynamic, time-dependant environment. We focus on changes in usage behavior of clients, considering the shape of the marginal resource utilization distributions as well as peak-demand. We begin with a basic scenario to continue the argumentation from the previous subsections. For simplicity, we assume the cost functions remain constant over time. First we define the continuous infrastructure function $x(t)$ to derive parameters needed for optimization. We start with a simple linear definition in Equation (9)

$$x(t) = \beta_0 + \beta_1 * t \qquad (9)$$

Consider a 2-year scenario with a linear marginal resource utilization distribution function d'. Further assume that the peak demand linearly doubles in these two years leaving the marginal resource utilization distribution functions unchanged. For the economic evaluation, we set the infrastructure size to handle the peak demand at the beginning to $s_b = 5$ and at the end to $s_e = 10$. Hence, the relative peak infrastructure size is $x^m(0) = 50\%$ and $x^m(24) = 100\%$.

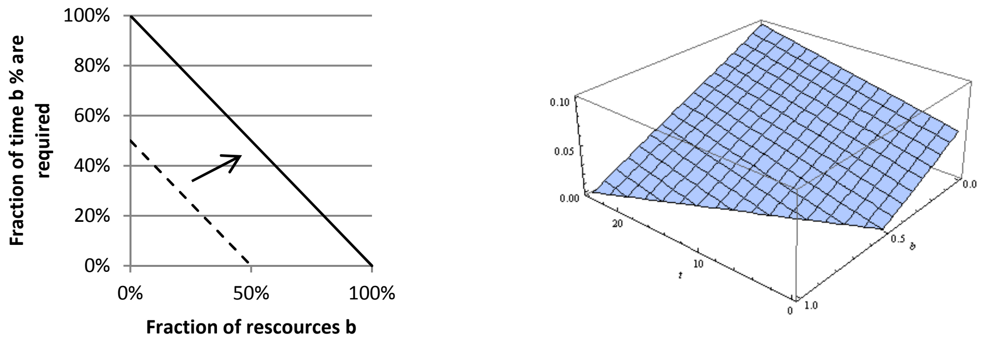


Figure 6: Simple continuous example

Figure 6 (left) shows this scenario. The right side of the figure has the shape of the marginal resource distributions over time. Recalling Proposition 2 and considering linearly increasing resource demand, we can directly derive the optimal infrastructure size (10).

$$x(t) = d^{-1}(\mathrm{t_b}, \mathrm{s_b}) + \mathrm{t}\frac{d^{-1}(\mathrm{t_e}, \mathrm{s_e}) - d^{-1}(\mathrm{t_b}, \mathrm{s_b})}{\mathrm{t_e} - \mathrm{t_b}} \quad (10)$$

Now we turn to a more realistic example with a utilization function subject to seasonal effects.

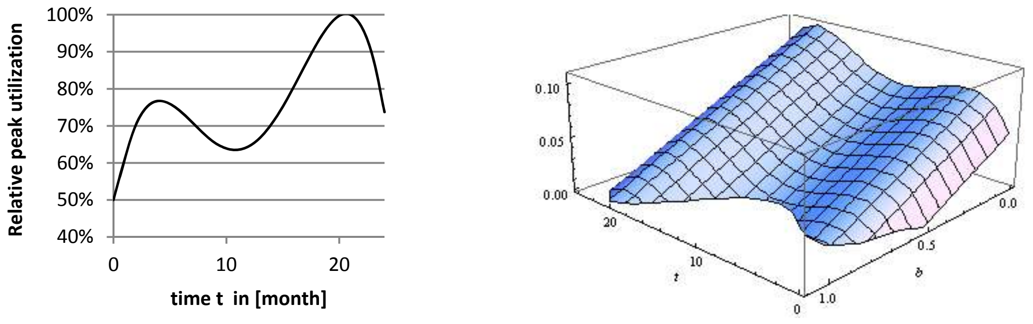


Figure 7: Dynamic utilization function

Figure 7 illustrates this scenario. The structure of the polynomial implies that we have to increase and reduce our infrastructure size continuously. Due to the complexity of the workload process, a linear infrastructure size function $\mathrm{x}(t)$ cannot represent the optimal solution. Therefore, we enhance the optimization $x(t)$ by a fourth-order-polynomial (DAISY P4) as our peak utilization process has three turning points. By applying a higher order polynomial, we can achieve a better fit as shown in Figure 8.

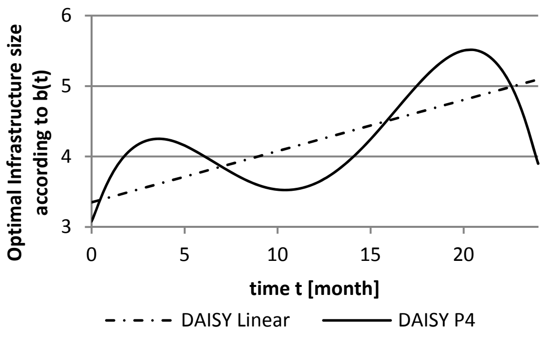


Figure 8: Optimal infrastructure size

Next we assess the cost of this scenario. We use the cost of operating the system only with owned resources and the costs operating only using IaaS as benchmarks. Figure 9 shows the average cost of operation per hour in all four scenarios.

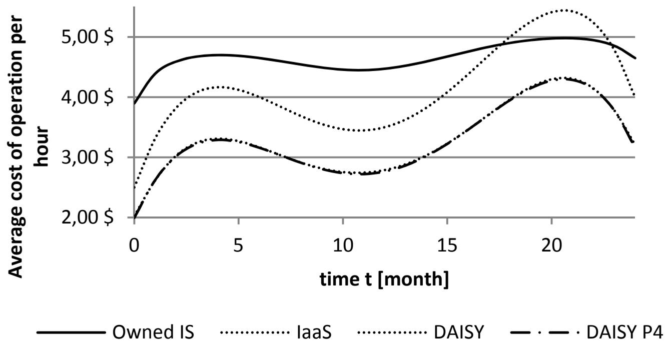


Figure 9: Average cost of operation

Applying DAISY reduces the average cost of operation significantly. As presented in Table 3 and depicted in Figure 9, the application of a higher order polynomial has almost no further economic effect.

Table 3: Avg. cost per hour	
Owned IS	$4.64
IaaS	$4.12
DAISY Linear	$3.27
DAISY P4	$3.26

In summary, DAISY achieves a more efficient solution than the extreme benchmark cases. The savings are realized by utilizing the cheaper owned-resources for the base-load and satisfying the peak demand with remote resources.

5. Discrete Model Application

The continuous version of DAISY assumes that servers are perfectly divisible resources. This section reformulates DAISY as a discrete time and resource problem because computing resources can generally only be purchased or rented as full units. We reduce the continuous time interval to discrete points in time. As a result the marginal resource utilization function is defined for each server during each period. This reduces the complexity of deriving the input data significantly because the marginal resource utilization becomes a relative frequency function of the observed workloads during the examined period.

5.1. *Mathematical Formalization*

Let $T = t_1, \dots, t_n$ be the set of phases and $X^d = x_1^d, \dots, x_n^d$, the number of servers owned in each phase, whereby x_t^d is a natural number between x^{min} and x^{max}. We no longer use the relative share of resources. Inheriting the definitions and elements of DAISY, the discrete version can be formalized as follows.

$$\min_B \sum_t^T \left(c^O(t) + c^R(t)\right)$$

$$c^O(t) = \sum_{x=x_{min}}^{x_t} d'(t,b)c_o(t) + \left(1 - d'(t,b)\right)c_i(t) \qquad (11)$$

$$c^R(t) = \sum_{x=x_t}^{x_{max}} d'(t,b)c_r(t)$$

This formulation permits several extensions. For instance, we may suppress the reduction of the infrastructure over time by including the constraints $b_{t+1} \geq b_t \forall t \in T$. This is a reasonable assumption, since a server that has been bought and added to the system, will typically not be removed from the system anymore.

A second extension is the inclusion of switching cost, which are caused by the adaption of the infrastructure size. By analyzing the original workload process, the number of switches in every period can be derived and transformed into a switching cost function. Interested readers should refer to [9] for a comprehensive model of the dynamics of workload processes and their implications on the infrastructure size.

5.2. Numerical Examples

In the following we investigate the implications of the discrete model. For that purpose, we may reuse the scenarios from the previous section.

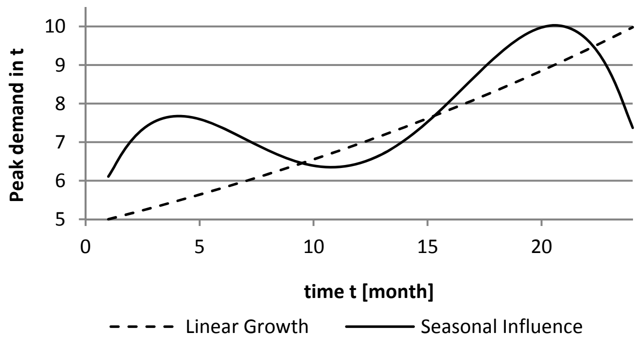


Figure 10: Scenarios

Figure 10 shows the peak resource demand over time. The first scenario constitutes a linear growth process, while the second additional seasonal effects. Similar to the previous cases, both show a steady increase over time. Again, the marginal utilization distribution is assumed to be a linear function.

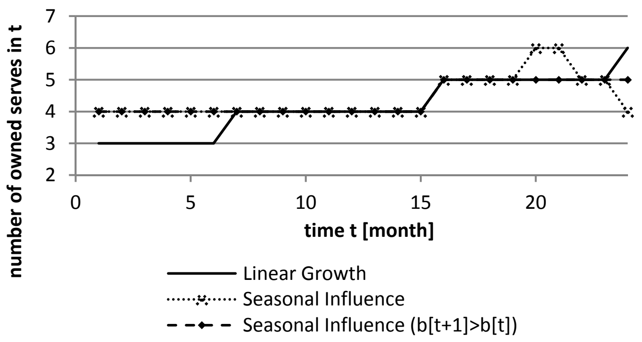


Figure 11: Investment strategies

Figure 11 shows the resulting operation strategy of DAISY. Similar to the continuous version of DAISY, in the linear growth scenario the number of servers is continuously increased. The same applies for the seasonal scenario. However, decreasing the numbers of servers before they reach the end of their life-time is typically economical infeasible. Therefore, the last scenario evaluates the cost minimal strategy including the constraint that the number of servers cannot decrease. Consequently, DAISY does not add another server in period 20 and 21 because it would have only been utilized in these two periods.

Table 4: Avg. cost per hour	
Linear Growth	\$3.14
Seasonal Influence	\$3.34
Ses. Inf. $b_{t+1} \geq b_t$	\$3.36

Table 4 shows the average total cost. The results are similar to the results of the continuous version in the same scenario. This suggests that the discrete version can be used without risking significant loses of precision. However, due to the computational complexity, solving discrete problems is often intractable, which is addressed by the next proposition.

Proposition 3: *DAISY (discrete) is solvable in polynomial time.*

A proof-sketch of this proposition has been moved to the Appendix. In summary, the discrete version of DAISY fulfills the requirements of a Decision Support System because: (i) the input data can be easily generated, (ii) the discrete optimization problem is solvable in polynomial time, and (iii) the loss of precision is insignificant compared to the exact formulation. In fact, the discrete formulation is more robust, which allows the integration of further properties.

6. Summary and Conclusion

Nowadays, enterprises face rapidly increasing cost of IT operation, which makes cost efficiency a top priority. In parallel, recent technological innovations in service science (e.g., cloud computing) have enabled new IT concepts that promise a radical cost reduction. In this light we have investigated the economic implications of combining company owned infrastructure with remote computing resources. In a model-based approach, we have derived a novel framework that assesses the total cost of IT system operation and enables sound investment decisions. Our results clearly revealed the economic benefits of combining company owned infrastructure and IaaS resources. In comparison to each concept by itself, such a hybrid approach can significantly reduce the total cost of operation and therefore enable more efficient investments utilization.

The main contribution of this paper is the systematic economic analysis of the fusion of IT infrastructure ownership with IaaS. Our results showed the close relationship between the dynamics of highly volatile workload processes, system utilization, and the cost of operation. Our framework (i.e., DAISY) addressed these dynamics, which offered novel insights into the cost of operation characteristics of information systems. Moreover, our model allows the derivation of optimal investment strategies based on the total cost of operation. For this purpose we have introduced the notion of marginal cost of computing.

Depending on the analysis scope, DASIY can be formulated in different ways, and we have presented three possible use cases in this paper. First, we have investigated time independent cost structures, which allows scenario analysis of infrastructure size impact and policy changes. Second, we have formulated the model with incorporated time variable to model cost dynamics and usage pattern changes. Third, we have restated the model as a discrete problem, which can be employed to derive cost optimal investment strategies. Despite their fundamental differences, the combined usage of own infrastructure and IaaS has proven beneficial in all of the investigated scenarios. Nonetheless, determining the optimal infrastructure strategy remains a non-trivial task. We have shown that the optimal solution is highly dependent on various (possibly time variant) factors, such as cost models and workload processes.

Our current and future work includes expanding DAISY as a stochastic optimization problem to derive feasible investment strategies under uncertainty. Currently, DAISY minimizes the total cost of operation; however, instead of minimizing the operational expenditures, the optimization of the cash flow might be of higher significance from a venture capital perspective.

7. Appendix

7.1. Proof of Proposition 1

For any fixed point in time $\bar{t}$, we choose an optimal infrastructure size by minimizing the TCO with respect to x. We define $c(x)$ as the TCO given an infrastructure size x. This leads to equation (12).

$$\min_{x} \mathrm{c(x)} = c_o d_o(x) + c_i d_i(x) + c_r d_\mathrm{r}(x) \qquad (12)$$

Rewriting the resource demand functions with their definition, results in equation (10).

$$\mathrm{c(x)} = \int_0^x \left(c_o d'(b) + c_i\left(1 - \mathrm{d'(b)}\right)\right) \mathrm{db} + c_i \int_x^1 d(b)\, db \qquad (13)$$

The extremes of the equation can be found by calculating the first derivative of $c(x)$ and setting the resulting term to zero (14).

$$c'(\mathrm{x}) = c_o d'(x) + c_i\left(1 - d'(\mathrm{x})\right) - c_r d'(\mathrm{x}) = 0 \qquad (14)$$

Solving this equation for $d'(x)$ reveals that the TCO minimal owned infrastructure size x^* is achieved, if the marginal resource utilization is equal to the given fraction of the cost factors.

$$d(x^*) = \frac{c_i}{c_i + c_r - c_o} \qquad (15)$$

Determining the inverse function of the marginal resource utilization function delivers the TCO minimal infrastructure size x^* in equation (16).

$$d'^{-1}\left(\frac{c_i}{c_i + c_r - c_o}\right) = x^* \qquad (16)$$

As a result the TCO minimal size of an IT system for a fixed point in time $\bar{t}$ is a function of marginal resource utilization and the given cost ratio. □

If the minimal amount of servers required by a center is a function of the marginal resource utilization, the same can be formulated in terms of the shape and magnitude of the same.

7.2. *Proof of Proposition 2*

This proposition directly follows from the argumentation of proof 1. The first step is to reformulate the optimum criterion for the continuous time case inserting the constant k to fix the cost.

$$d'^{-1}(\mathrm{t,k}) = x^*(\mathrm{t}) \qquad (17)$$

From the formulation in (17) we can directly see that the cost minimal infrastructure size only depends on the marginal resource distribution.

In conclusion, as long as we hold the relaxation, that resources are completely divisible, we can derive the general optimality function $x^*(t)$, providing the optimal infrastructure size over time. This leads to equation (18):

$$d'^{-1}\left(\mathrm{t}, \frac{c_i(\mathrm{t})}{c_i(\mathrm{t}) + c_r(\mathrm{t}) - c_o(\mathrm{t})}\right) = x^*(\mathrm{t}) \qquad (18)$$

This solution is universal for any given marginal resource utilization function $d'(t, b)$. □

7.3. Proof-Sketch of Complexity

To prove, that the discrete version of DAISY is solvable in polynomial time, we transform it into the shortest path problem. We define a set of vertices V, whereby each vertex represents one valid configuration for every period (19). The upper and lower bound for resources is given by x^{min} and x^{max} where x refers to discrete resources.

$$v_{t,x_t} \forall t \in T, x_t = x^{min}, \dots, x^{max} \qquad (19)$$

The set of edges E between the vertices is defined as the costs to operate an infrastructure of size x in phase t including the charge for potentially required IaaS resources.

$$e := < v_{t,x_t}, v_{t+1,x_t} >, \qquad \forall t \in T,\ b_t = x^{min}, \dots, x^{max} \qquad (20)$$

Finding the minimal cost of operation is an instance of the shortest path problem, as we have to find the path through the network with minimal costs. Therefore DAISY has a polynomial runtime complexity.

8. References

Altmann, J. et al., 2007. A Taxonomy of Grid Business Models. In *Gecon2007, Intl. Workshop on Grid Economics and Business Models*.

Anon, Experimental evaluation of N-tier systems: Observation and analysis of multi-bottlenecks. Available at: http://www.computer.org/portal/web/csdl/doi/10.1109/IISWC.2009.5306791.

Assuncao et al., M.D. d, 2008. Evaluating the cost-benefit of usinf cloud computing to extend the capacity of clusters. In *HPCC '08: Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications*.

Buyya, R., Yeo, C.S. & Venugopal, S., 2008. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In *HPCC '08: Proceedings of the 2008 10th IEEE International Conference on High Performance Computing and Communications*. Washington, DC, USA: IEEE Computer Society, pp. 5-13.

Channabasavaiah, K., Holley, K. & Tuggle, M., 2004. Migrating to a service-oriented architecture. *IBM Developer Works*, pp.G224-7298.

Chen, Y. et al., 2005. Managing server energy and operational costs in hosting centers. *ACM SIGMETRICS Performance Evaluation Review*, 33(1), pp.303-314.

Cohen, I. et al., 2004. Correlating instrumentation data to system states: a building block for automated diagnosis and control. *Operating Systems Design and Implementation*, p.16. Available at: http://portal.acm.org/citation.cfm?id=1251270 [Accessed September 15, 2010].

Gilley, K.M. & Rasheed, A., 2000. Making more by doing less: An analysis of outsourcing and its effects on firm performance. *Journal of Management*, 26(4), pp.763-790.

Gray, J., 2008. Distributed Computing Economics. *Queue*, 6(3), pp.63-68.

Jagannathan, S., Altmann, J. & Rhodes, L., 2003. A Revenue-based Model for Making Resource Investment Decisions in IP Networks. In *IFIPIEEE Eighth International Symposium on Integrated Network Management 2003*. pp. 185-197.

Konstantinou, A.V. et al., 2009. An Architecture for Virtual Solution Composition and Deployment in Infrastructure Clouds Categories and Subject Descriptors. *Environments*, pp.9-17.

Krauter, K., Buyya, R. & Maheswaran, M., 2002. A taxonomy and survey of grid resource management systems for distributed computing. *Softw. Pract. Exper.*, 32(2), pp.135-164.

Lee, J.-N. et al., 2003. IT outsourcing evolution: past, present, and future. *Communications of the ACM*, 46(5), pp.84-89.

Malkowski, S. et al., 2007. Bottleneck Detection Using Statistical Intervention Analysis. In A. Clemm, L. Z. Granville, & R. Stadler, eds. *DSOM '07*. Berlin, Heidelberg: Springer-Verlag, pp. 122-134.

Mituzas, D., 2011. Wikistats. Available at: http://dammit.lt/wikistats/ [Accessed August 30, 2011].

Padala, P. et al., 2007. Adaptive control of virtualized resources in utility computing environments. *ACM SIGOPS Operating Systems Review*, 41(3), p.289.

Ragusa, C., Longo, F. & Puliafito, A., 2008. On the Assessment of the S-Sicilia Infrastructure: A Grid-Based Business System. *Grid Economics and Business Models Lecture Notes in Computer Science*, 5206, pp.113-124.

Risch, M. & Altmann, J., 2008. Cost Analysis of Current Grids and Its Implications for Future Grid Markets. *GECON 08 Proceedings of the 5th international workshop on Grid Economics and Business Models*, pp.13-27.

Urgaonkar, B. et al., 2008. Agile dynamic provisioning of multi-tier Internet applications. *ACM Transactions on Autonomous and Adaptive Systems*, 3(1), pp.1-39. Available at: http://portal.acm.org/citation.cfm?doid=1342171.1342172.

CHAPTER VI

RISK-AWARE SERVICE LEVEL AGREEMENT DESIGN FOR ENTERPRISE INFORMATION SYSTEMS

This chapter is a revised version of the paper:

Hedwig, Markus; Malkowski, Simon and Neumann, Dirk, "Risk-Aware Service Level Agreement Design for Enterprise Information Systems" (2012), 45rd Hawaii International Conference on System Sciences Proceedings, Copyright 2012 IEEE, Digital Object Identifier 10.1109/HICSS.2012.509.

Abstract

Effective information systems have become key to sustainable business practices. However rising costs of operation and a growing system complexity are driving the search for a more efficient delivery of corporate computing. New technologies in the emerging service world such as cloud computing provide powerful alternatives to traditional IT operation concepts. Nonetheless, executive decision makers still have reservations about migrating to this new technology. In addition to security concerns, a key issue is the still prevailing lack of strict SLAs in these service offerings. In fact, service providers hesitate to offer strictly binding SLAs because assessing economic risk exposure is a major challenge. In this paper, we present a novel model for sustainable SLA design for enterprise information systems. Our model combines various state-of-the-art concepts from the field of system management and balances the failure risk with the cost of operation. More concretely, our model helps IT decision makers to understand the relationship between the operation cost and the service quality. Consequently, the minimum economic price of a service and the corresponding operation strategy, given the customer requirements and the infrastructure characteristics, can be determined based on our approach.

Keywords: Service Level Agreements, Operation Strategy, Elastic Application Design

1. Introduction

Effective Information Systems (IS) have become the backbone of modern corporate governance. Both the ability to process extensive information quickly (e.g. data warehousing) as well as the operation of innovative and powerful customer interfaces (e.g. e-commerce applications) have become vital necessities in today's dynamic business world. However, a rising cost of operation as well as the steadily increasing complexity of IS make their sustainable operation a challenging problem (Schulz 2009; Koomey 2007).

The demand for economical and reliable IS solutions as well as various technological advances have favored the emergence of a new business field (KPMG 2007). Today, various IT service providers offer a variety of computing services remotely. Their service portfolio ranges from low-level storage and infrastructure services, to complete enterprise information systems offered as Software-as-a-Service (SaaS) applications over the internet. In contrast to traditional in-house operation, service providers can leverage economy of scale and scope effects. Thus, by offering their services to multiple customers simultaneously and building up expertise, they can provide corporate computing demands more cost effectively (Bailey 2009).

Nonetheless, enterprises still have reservations to migrate their systems to this new technology. Next to security concern, the main reason is that information systems often belong to the critical infrastructure of a company. The risk of failure and loss of business is valued significantly higher than the saving potential. Hence, to consider service providers as an alternative, enterprises demand securities for the proper service operation or compensating equalization payments in case of failure (Deloitte Consulting 2005; Goolsby 2007). However, still today, most service providers offer their services only on a best effort base.

Within the service world, Service Level Agreements (SLAs) have become the common practice to define the terms of business between two parties. They include the Service Level Objectives (SLO), such as the maximum tolerated response time as well as the financial arrangements for the service usage. With the maturing service market, providers are slowly moving towards their customer's requirements and starting to offer their services with strict binding SLA's and substantial penalties in case of non-compliance(Sahni & Tan 2011).

Nonetheless, with more and more competitors entering the market and the establishment of service standards (Thibodeau 2011), the pricing pressure increases (Sullivan 2009). Consequently, competitive as well as economic pricing has become a hard challenge, and providers are forced to operate their systems as efficiently as possible using state-of-the-art technology (Susarla & Barua 2009).

In this context, we present our novel SLA design model, which helps service providers offer their service portfolio for optimal profit. By simultaneously evaluating various technical and economic aspects of service operation, our model allows for optimal operation strategies as well as economic and competitively viable prices to be found. Our model can be used to assess the economic impact of parameter variation in SLAs. For instance, it enables determining the surcharge if the provider agrees to guarantee a certain response time in 99% instead of 95% of the SLA lifetime.

The main contribution of this paper is twofold. First, this paper presents a holistic overview over all aspects of modern service operation and their relations. Second, based on our integrative model, we offer interesting insights into the economics of service operation. For instance, our model derives the relationship between the cost of operation and the probability of SLA compliance. In particular, we will show that the deliberate choice of a higher operational risk may lead to higher provider profits. Furthermore, given a set of Service Level Objectives and a desired penalty, our model can determine the risk-neutral price of the service. Additionally, our model can be used to assess if a client request for services has an expected positive profit, and which is the most suitable infrastructure to operate the service.

The remainder of the paper is structured as follows. The rest of this section presents our scenario and details the economic aspects of our model. The second section provides a brief overview of related and background work, followed by a detailed discussion of our SLA design model. Finally, in section 4 our model is evaluated. Section 5 summarizes the paper and provides a conclusion and an outlook on our future work.

1.1. Service Scenario

For the presentation of our model, we selected the scenario of a Software-as-a-Service (SaaS) provider, offering an enterprise information system (e.g. ERP or CRM) to a client. The SaaS application is operated in a cloud computing environment, which charges on a pay-as-you-go basis for every resource unit.

In order to support highly efficient operation modes and to leverage the potential of the cloud technology, our model is designed for the latest generation of cloud applications. Elastic applications enable resource adaptive operation modes. More concretely, depending on the number of active users, this application design allows adding and removing computing resources (e.g. servers) from the system at runtime. In particular, e-commerce applications may face significantly higher workload levels during day than at night. By continuously adapting the system size to the user demand, the systems can be operated with very high resource efficiency, and hence significantly lower cost of operation.

However, the management of elastic applications is non-trivial. Compared to classic operation modes, effects such as sudden workload peaks or hardware failures accommodated a higher risk of SLA violations. To compensate this risk, service providers may add reserve resources to the system. Though this increases the cost of operation, it reduces the risk of SLA violations significantly. We will refer to provider's handling of resources as the operation strategy (OS).

1.2. Economics of Service Operation

This section discusses the economic perspective of the provider. Depending on the technical and economic parameters, his goal is to operate his system with optimal profits. However, given an accepted SLA, his only setscrew is the operation strategy.

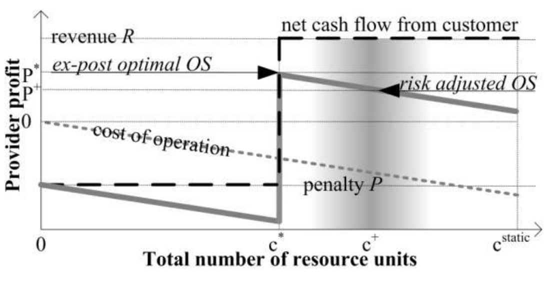


Figure 1: Optimal Operation Strategy

Figure 1 depicts the situation for the service provider. For the successful operation of the service he receives the revenue R, otherwise he has to pay a penalty P (black-dashed line). The gray-dotted line represents the cost of operation depending on the number of used resources according to the operation strategy. As the y-axis shows the profit, the cost of operation has a negative impact. If the provider provisions too few resources, he is not able to fulfill the SLA and has to pay the penalty. On the other hand, if he overprovisions his profit drops (solid gray line). Under full information, the service provider could run the optimal operation strategy with a total amount of resource units c^* and receive the maximum profit P^*. However, this is usually impossible; hence to account for uncertainty during operation, the rational provider would provision reserve resources.

Figure 2 depicts the economic situation of the provider. Ceteris paribus, we examine the expected profit depending on the operation strategy. Evidently, with the increasing amount of provisioned resources, the probability of SLA compliance increases. Simultaneously the cost of operation also increases and hence there is an expected profit optimal operation strategy balancing the cost of operation and the probability of SLA violation. This leads to the central research question of the paper. Given all factors of influence, what is the optimal operation strategy to maximize provider profit?

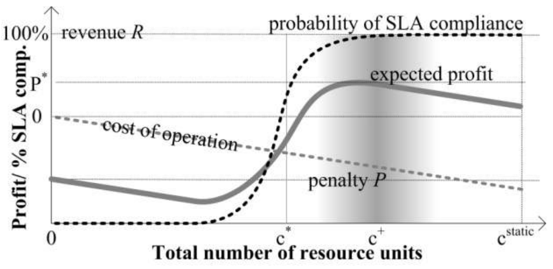


Figure 2: Economics of Service Operation

Figure 2 depicts the economic situation of the provider. Ceteris paribus, we examine the expected profit depending on the operation strategy. Evidently, with the increasing amount of provisioned resources, the probability of SLA compliance increases. Simultaneously the cost of operation also increases and hence there is an expected profit optimal operation strategy balancing the cost of operation and the probability of SLA violation. This leads to the central research question of the paper. Given all factors of influence, what is the optimal operation strategy to maximize provider profit?

In particular, the economic parameters of the SLA have a significant impact on the operation strategy. Rationally, if the provider agreed to a high penalty, he will tend to provision more resources to mitigate the risk of SLA violations. On the other hand, for low cost services it might be economically beneficial to accept occasional SLA violations as in the long-term the saved costs of operation outweigh the penalties.

2. Related Work & Background

The profit optimal and risk-aware operation of services requires a thorough understanding of the whole domain. In this section we provide an overview of our previous work and discuss how our model differs from related approaches.

In contrast to similar concepts in research and practice, our model addresses the question of profit optimal information system operation holistically. More concretely, we combine a system performance model, a workload model as well as an economic SLA model into one integrated model.

To the best of our knowledge, this has not been done before. Generally in literature two research threads can be identified. First, various models and concepts exist for the management of services based on SLAs, or more general on Quality of Service (QoS) requirements. For instance, the paper (Yu & Buyya 2006) presents a model for the scheduling of scientific workload applications based on QoS and economic parameters. In the article (Du et al. 2008), the authors present a storage-provisioning network based on QoS requirements. In a different context, the authors in (Sen et al. 2010) design and evaluate network infrastructure investment strategies for content delivery networks. The article (Skital et al. 2008) presents the application of SLAs in the Grid context for real-time applications.

The second research thread comprises concepts for the efficient online management of IT systems. Generally, the operation of elastic application belongs to the field of dynamic resource management. For instance, the authors in (Gmach et al. 2009) developed a reactive migration controller for virtualized environments. The paper (Chandra et al. 2003) presents a resource allocation model for shared datacenters. Another concept (Padala et al. 2009) is an automated control model for virtual resources. In (Ardagna et al. 2007) a model has been developed to manage the resource demand of multiple concurrent systems. In the paper (Urgaonkar et al. 2007) the authors developed a surge protection model for dynamic resource allocation.

In summary, while the first thread aims to develop sophisticated and domain specific models for the economic and efficient management of services, the second thread targets the implementation challenges. However, we haven't found any work unifying both aspects for information systems in the service world. In the remainder of this section, we present our recent research work in this field and introduce the formal aspects.

2.1. Service Level Agreements

The main objective of elastic application management is to comply with the SLO defined in the SLA. A SLO usually consists of a monitoring metric and a corresponding threshold (e.g. response time less than one second). In our scenario, we further define that α percent of all requests within a short period must comply with this goal (e.g. 90% of requests per second must comply with the SLO). Furthermore, SLAs usually do not demand compliance with the SLO during the whole SLA lifetime, but allow occasional violations. Hence, we define A as the percentage of time the SLO must have achieved to fulfill the SLA.

SLAs and their different aspects have been extensively discussed by the research community. The article (Buyya et al. 2009) provides a good overview of SLAs in the field of clouds. The authors in (Yeo & Buyya 2007) developed an integrated risk analysis scheme to analyze the effectiveness of resource management policies. In the paper (Raimondi et al. 2008), the authors present the implementation of an automated SLA monitoring for services.

2.2. System Performance Model

The resource efficient and SLO aware operation of an elastic application requires a thorough understand of the system, including the infrastructure and software characteristics as well as the user behavior. In our recent research we developed an empirical performance model (Malkowski et al. 2011). Based on the monitored performance during operation, our empirical model is able to determine which configurations of the system are able to sustain a certain workload level.

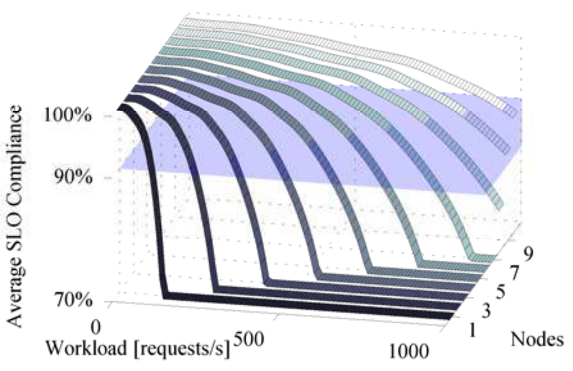


Figure 3: System Performance Model

Figure 3 depicts the performance characteristics of an elastic application. The description axes show the number of resource units c as well as the workload level w. The value axis shows the degree of SLO compliance defined as $s(w, c)$. In this scenario, each additional resource increases the maximum number of concurrent users by approximately $s^{max} = 100$ users.

$$\eta(w) = \min_{c} s(w, c) \geq \alpha \quad (1)$$

Based on the system performance function s, we can provision the minimal sufficient system configuration η according to the workload level and the individual SLO requirements. Generally, empirical performance modeling is only one approach; alternatives are, for instance, queuing models (Urgaonkar et al. 2008) or machine learning (Cohen et al. 2004).

2.3. User Behavior & Workload

The workload is the most important criteria determining whether a system is able to meet QoS requirements. Traditional design concepts scaled systems to the maximum expected workload. However, in particular, web facing information systems face highly volatile workloads, which may cause low average system utilization. Though elastic applications allow resource adaptive operation without compromising system stability, information systems usually have a reconfiguration lead-time. Hence, to avoid performance short-comings, reconfiguration decisions need to be initiated in advance. In our recent research (Hedwig et al. 2010; Hedwig et al. 2009), we developed a workload forecast model which predicts the near future behavior of a workload process. Together with the system performance model and in accordance to the SLA definition, this allows for adapting the system continuously to the user demand.

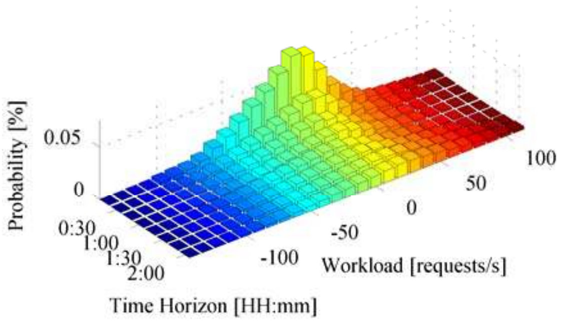


Figure 4: Workload Forecast Error Distribution

Nonetheless, as workload processes are usually stochastic, we face a risk of workload prediction errors. Figure 4 depicts the forecast error distribution of our workload forecast mechanism. If we assume a lead time of 1 hour and that each node in the system is able to handle 100 concurrent users, we have a positive risk of selecting a configuration which is not capable of handling the workload. In this sense, we define $P_{\Delta}^{WLF}(\omega)$ as the probability that the observed workload differs from the predicted work by ω

$$\Omega(\beta) = \frac{1}{s^{max}} \int_{0}^{s^{max}} \int_{s^{max}*(\beta+1)-w}^{\infty} P_{\Delta}^{WLF}(\omega - w) d\omega \, dw \quad (2)$$

To mitigate the risk of SLA violations caused by wrong predictions, we can overprovision the system. However, depending on the volatility of the workload and the characteristics of the system, there is still a risk that the additional resources are also not sufficient to cover the unexpected demand. We define Ω as the probability that the system is able to comply to the SLA if we provision β additional resources. The first integral describes the residual expected performance reserves of a configuration if we predict the workload level w, the second the probability of a wrong workload prediction. We will later use Ω to incorporate the risk of resource adaptive operation in our risk model.

Alternative concepts are, for instance, the use of time series analysis (Powers et al. 2005), empirical workload distribution estimation (Urgaonkar et al. 2008), or a time series mode with combined with Fourier Transformation (Gmach et al. 2007).

2.4. *Reliability*

Another risk factor which may lead to SLA violations are hardware or infrastructure failures. In our model we define δ as the survival probability of a single node and ν as the survival probability of the infrastructure.

Static system designs are usually scaled to handle the maximum expected workload level plus performance reserves. In adaptive operation modes however, with a minimal system size, node failures are more critical as the system would be immediately overloaded. In order to mitigate this risk, the operation strategy can be set to provision additional resources in each period. Though fault nodes can be replaced immediately in cloud environments, we still have the lead-time until the replacement resources are available.

3. SLA Design Model

In this section, we present our SLA design model for the determination of the economic parameters of SLAs based on the desired SLOs, user behavior, and system characteristics. Various factors influence the performance of an elastic application and thus may cause performance short-comings or SLA violations. In this paper, we focus on the risk induced by the infrastructure and node reliability as well as by resource-adaptive operation modes. However, the general idea of our model is not limited to the aforementioned factors and can easily be extended with other factors such as data quality considerations or flash crowds.

3.1. *Modeling of Risk Factors*

Based on the presented concepts in section 2, we now examine their impact on the probability of SLA compliance. In our model, we assume independence between the different factors of influence.

The first factor refers to the resource node reliability δ, which determines the survival probability of a node in one period. Depending on the operation strategy and the expected workload, in each period a certain amount of resources is provisioned. However, as the provider is aware of potential node failures, he may decide to provision reserve resources. For instance, if the system is operated with one additional resource, it can compensate the failure of one node. However, if two nodes fail, the system might violate the SLA. To determine the probability of a critical number of node failures, we calculate the binominal system survival probability given that c resource units are provisioned and $c^{min} \leq c$ is required:

$$\lambda(c, c^{min}) = \sum_{i=c^{min}}^{c} \binom{i}{c} \delta^{i} * (1-\delta)^{c-i} \quad (3)$$

The function λ defines the probability that at least c^{min} nodes are online. Consequently, with the increasing amount of reserve resources $c - c^{min}$, the survival probability of the system tends towards 100%.

The second risk factor refers to infrastructure failure risk. The modeling of failure risk is quite simple as we assume that in each period the system has a positive probability of failure. Hence, we define v as the survival probability of the system in one single period. Thus, the probability that the infrastructure operates without failures for n periods is v^n.

The third risk factor points to elastic application operation risk, which defines the risk of provisioning too few resources. In our model, we assume independence of the prediction accuracy from the time of day and the current workload level, which allows us to aggregate the forecast risk into a single factor.

$$\Omega(c - c^{min}), \text{with } \beta = c - c^{min} \quad (4)$$

The prediction failure probability $\Omega(\beta)$ from the previous section can be directly applied to our model. Similar to node failure risk, the impact of inaccurate workload predictions diminishes with the number of reserve resources.

3.2. System Reliability during SLA Lifetime

Based on the three independent risk components, we can determine the probability that a configuration with c resource units facing a workload level of w is capable of fulfilling the SLO in one period. With increasing number of units c the survival probability likewise increases.

$$r(c, w) = \lambda(c, \eta(w))\Omega(c - \eta(w))v \quad (5)$$

Depending on the compliance probability of a single period given a certain amount of resource units, we can now determine the survival probability for the whole SLA lifetime.

Suppose we demanded 100% SLO compliance in all periods of the SLA lifetime, we obtain the compliance probability by multiplying the survival probability of the single periods.

$$\pi(C) = \max_{c_1,\dots,c_n} \prod_{i=1}^{n} r(w_i, c_i) \quad (6)$$

$$\text{subjected to:} \sum_{i=1}^{n} c_i = C \quad (7)$$

Recalling the operation strategy, we now define $\pi(C)$ as the survival probability of the system during the SLA lifetime if C resource units are provisioned in total. By solving the maximization problem, we can derive the optimal allocation of the resources over time. This optimization problem can be solved in linear time utilizing a basic heuristic. Starting with the smallest configuration c^{min}, we add one resource unit in each iteration to the period with the highest reliability gain.

3.3. Degree of SLO Compliance

In the previous presentation we required 100% SLO compliance in all periods during the SLA lifetime. However, in most cases the contracts will allow that the system temporarily drops below the SLO goals. Apparently, we only have to fulfill the SLO during $A\%$ of the SLA lifetime, which means that we can design our operation strategy to tolerate non-permanent violations. Calculating the probability of the compliance rate A is non-trivial. Thus, we must calculate the probability of all combinations of successful and violating periods to determine the compliance probability during the SLA lifetime. In our work, we approximate this probability by calculating the geometric average of the period survival probabilities $\phi = \sqrt[1/n]{\pi(C)}$ and define $q = \lfloor n * (1 - A) \rfloor$ as the approximated number of periods we allow to violate the SLO.

$$\tilde{\pi}(c) = \sum_{i=0}^{q} \binom{i}{n} (\phi)^{n-1} (1 - \phi)^{i} \quad (8)$$

With the help of the binominal distribution, we can now determine the survival probability $\tilde{\pi}(C)$ of our system in terms of the SLA goal A, given an operation strategy with the total number of resources C.

3.4. *Maximization of the Provider Profit*

Based on the system survival probability function π in dependence of the system configuration c, we can now derive the profit optimal operation strategy based on the price R, the penalty P, and the cost per instance hour K.

$$G(P,R,K) = \max_{C} R * \pi(C) - P\big(1 - \pi(C)\big) - K * C \quad (9)$$

By multiplying the probability of SLA compliance π with the service price R, we get the average expected revenue. Similarly, we obtain the expected average penalty per period. Furthermore, we can calculate the cost of operation by multiplying the number of total resources C with cost per instance and period K. Solving this maximization problem provides us with the expected maximum profit G we can achieve with the risk optimal operation strategy.

$$G(P,R,K) = \max_{C} R * \pi(C) - P^{\theta}\big(1 - \pi(C)\big) - K * C \quad (10)$$

In the previous formulation, the provider has been risk neutral and hence only optimizes his profit. However, in case of high penalties, the provider might consider an SLA violation as more dangerous. In order to extend our model with risk awareness, we include an additional risk parameter θ into the model which allows us to consider the penalty as more harmful and hence choose a less risky operation strategy.

3.5. *Minimum SLA Price Determination*

The minimum price is the price the provider needs to charge in order to have an expected profit of zero so that he at least does not suffer from loses. By setting $G(P,R,K) = 0$ and solving the equation for the revenue R, we obtain the following minimization problem.

$$\overline{R} = \min_{c} \frac{P\big(1 - \pi(c)\big) + K * c}{\pi(c)} \quad (11)$$

The variable $\overline{R}$ defines the minimum price such that the provider is willing to accept the SLA. Conversely, if the service price is lower than this value, the provider should not accept the SLA.

This formulation of our model is useful as it allows providers to offer their services to the customer in a very customer friendly way. The provider allows the customer to individually select his SLO settings and the desired penalty he needs in order to compensate his losses in case of an SLA violation. On the basis of this information, the provider can determine the minimum price which he needs to charge to provide the service.

Please note, that the service price $\overline{R}$ covers the operation cost of the violated periods as well as the penalty payments and lost revenue as it is derived from the average expected profit. Therefore we can consider $\overline{R}$ as the average cost of providing the service.

Normally, a provider would not provide the service with an expected profit of zero but instead add a margin. Based on the cost of operation and probability of SLA compliance, we can now derive the average profit of the provider.

$$\overline{G} = (R - \overline{R}) * \pi(C^*) \quad (12)$$

The average profit of the provider $\overline{G}$ is defined as the price R minus the average cost of operation multiplied by the probability of SLA compliance. This simple equation allows the provider to incorporate his margin into the price. Solving this equation for R allows the provider to price in his margin into the service price.

4. Evaluation

In the first part of the evaluation, we present insights into the reliability of elastic applications. Afterwards, we discuss the impact of different SLA parameters on the operation strategy. In the second part of the evaluation, we will show how our model can be used to determine optimal price for services given the desired penalty of the client and his QoS requirements.

Table 1: System Parameters

	Cost per Instance/h (K)	Node MTTF (1/δ)	Infrastructure MTTF (1/v)
Low-end	\$0.15	1d	1M
Normal	\$0.40	2d	6M
High-end	\$0.60	6M	36M

Throughout the evaluation, we will use the following parameter settings. The SLO is set to $\alpha = 90\%$. This objective must be met during the whole SLA lifetime $T = 48$ hrs and is hence defined as $A = 100\%$. The service price R and penalty P are set to \$200. Furthermore, we define three reference systems with individual characteristics (Table 1), whereby we use the normal system as default. The SLO and SLA compliance are evaluated every $\tau = 60$ minutes, which leads to a total of $n = 48$ periods. The provider receives the revenue in the case he complies with the SLA; otherwise he has to pay the penalty.

4.1. *Insights into the SLA Design Model*

Elastic applications support resource adaptive operation modes. However, to leverage their full potential, the configuration of the systems needs to be adapted to the user demand continuously. Continuing the economic situation of the service provider in Section 1.2, Figure 5 depicts the configuration of an elastic information system over time. The black line shows a Wikipedia workload trace from a single day (Mituzas 2011). We will use it as a reference trace through the evaluation. Evidently, different traces would lead to different results. Nonetheless, due to space constraints we focus on the other effects.

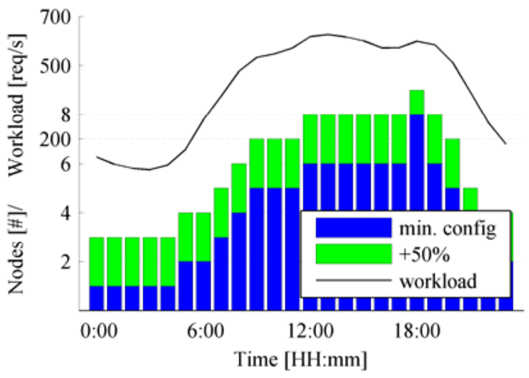


Figure 5: Elastic System Operation

The blue bars represent the minimal required configuration in each period to comply with the SLO goal α.The green bars illustrate the additional required resources in each period to provide the service profit-optimally. Therefore, additional resources are provisioned to mitigate the risk of node failures or workload miss predictions. This leads to the question, how much additional resource should be provisioned to operate profit-optimally?

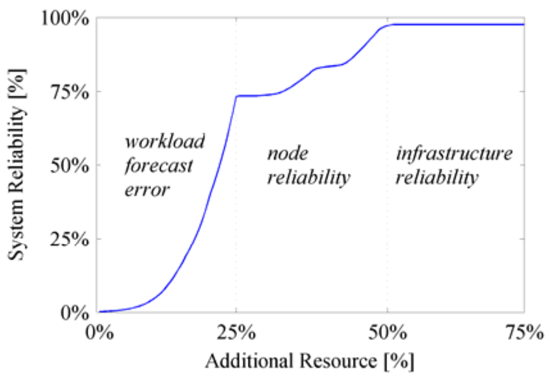


Figure 6: Impact of add. Resources on Reliability

To answer this question for the given scenario, Figure 6 shows the impact of additional resources on the service reliability in terms of the SLA. By definition, the smallest configuration is able to meet the SLO goal in every single period. However, over multiple periods the probability to meet the SLA goal tends towards zero. This striking observation stems from the fact that the minimal configuration probably violates the SLO in at least one period. In this scenario, this causes a violation of the SLA.

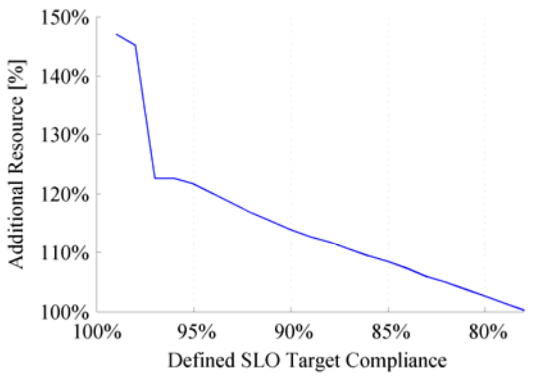


Figure 7: SLA - A Goal against add. Resources

In its minimal configuration the system requires 4.23 resource nodes on average. Therefore by supplying 25% more resources, on average the system obtains approximately one additional resource unit in each period. In our scenario, this additional resource unit primarily compensates the forecast error which has risk of miss prediction the workload of 5% in each period. The probability that the required infrastructure size is two nodes higher than the predicted is already insignificant, which explains the sharp kink at 25%. Given the node reliability of about 98% there is still a significant risk of two node failures within one period. As soon as we provide 50% more resources (i.e. approximately two additional nodes per period), the system reliability increases and tends towards the infrastructure reliability. As our scenario does not provide any options to reduce the infrastructure risk, this is the maximum reliability the system can achieve. In summary, if we target a strict SLO compliance in the SLA, the system requires 50% additional resources to provide this service profit-optimally. Nevertheless, there is still a residual risk of about 2% that the system violates the SLA.

Evidently, aiming at 100% SLO compliance increases the resource demand significantly. Figure 7 presents the relation between the desired SLA compliance A and the required additional resources. The minimal configuration is able to comply with SLA if A = 78%. In order to operate the system with A = 95%, we have to provide approximately 20% more resources. In order to achieve A = 99% SLA, the average system size has to be increased by 47%. As a result, the cost of operation for the aforementioned service qualities increases equivalently.

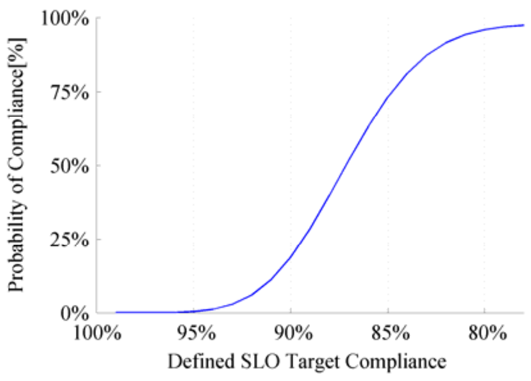


Figure 8: Min. System Reliability depending on A

Figure 8 depicts the probability of SLA compliance for the minimal configuration in dependence of the SLA goal A. Operating an information system with the minimal configuration at any point in time is highly cost efficient. However, this minimal operation mode has almost a certain risk that management and system errors occur. Nonetheless, depending on the SLA definition, a temporary violation of the SLO in one period does not necessarily lead to an agreement violation (A < 100%). In particular for aid-financed web services, this inexpensive operation mode might be useful, if occasional performance shortcomings are acceptable. Therefore, if the SLA goal A is set to 80%, the probability of SLA compliance is nearly 100%.

In summary, the first part of the evaluation demonstrated the impact of the degree of SLO compliance on the cost of operation.

4.2. Determination of Optimal Prices

In the second part of our evaluation, we reuse our previous scenario, but now aim at finding the optimal price for a service given the customer's QoS requirements ($\alpha = 90\%, A = 100\%$), the SLA lifetime, and the desired penalty. This view is particularly interesting as customers can self-select their desired QoS level as well as they can forward their losses to the service provider in case of non-compliance. Based on this information, our model allows determining the minimum price the provider has to charge in order to operate economically. Hence, we determine the price which leads to an expected profit of zero. Again we assume the three aforementioned reference systems and set the SLA lifetime to one week.

In Figure 9, the green, blue, and black lines represent the minimal required price in terms of the desired penalty. Not surprisingly the revenue/ penalty relation differ strongly between the three system types. While for the low-end system, for each additional dollar of penalty the provider has to charge about $0.33, the operation of the high-end system is nearly risk-free, and the provider can basically agree to any penalty.

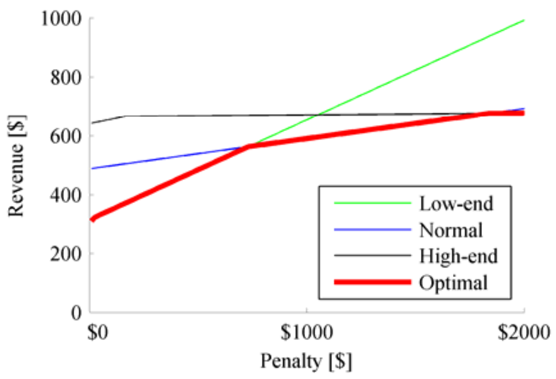


Figure 9: Optimal Revenue against Penalty (I)

The interception with the y-axes represents the required price in the absence of penalties which boils down to the cost of operation divided by the SLA compliance probability (as the provider only receives the revenue in case of SLA compliance, he has to cover his total cost of operation with the earnings of the successful periods). The cost of operating the system with the low-end nodes is the cheapest option in terms of cost of operation. However, only for penalties of up to $1000, is the operation of the low-end system feasible. With the increasing penalty, the provider is forced to switch to the normal system, and at a penalty of about $2000 to the high-end system.

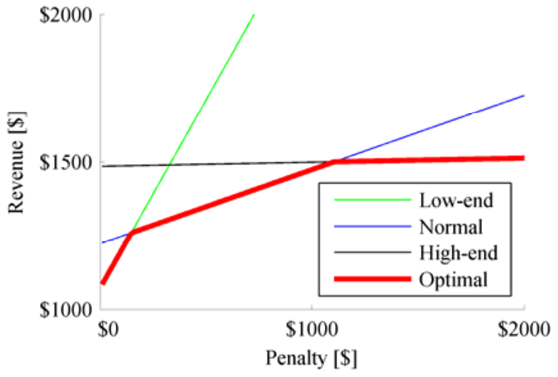


Figure 10: Optimal Revenue against Penalty (II)

Figure 11 shows the same scenario with an increased SLA lifetime of two weeks T = 2w. Evidenlty, the low-end system is only feasible for very low penalties. Furthermore, operating the service on the normal and high-end nodes is already feasible for smaller penalties compared to the previous example.

At first sight, one might assume that the customers should ask for long living SLAs as the provider tends to provide their services on better hardware. However, looking on the cost side, these longer living contracts dramatically increase the cost of operation, as the provider suffers from higher losses due to the exponentially increasing failure risk with the SLA lifetime. While in the first scenario a desired penalty of $1000 leads to a service price of $500, in the second scenario the price is already $1500.

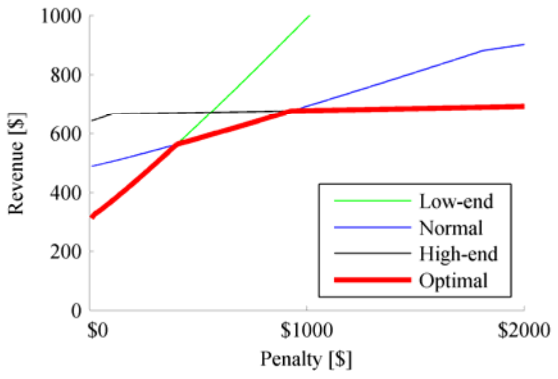


Figure 11: Optimal Revenue against Penalty (III)

In the previous section, we assumed the provider is indifferent to his risk position and hence agrees to any penalty. However, this behavior is unlikely as the provider will probably be very cautious in accepting SLAs with very high penalties. Figure 11 presents the risk-aware modification of our model with a risk aversion parameter θ = 1.1. These parameters increase the valuation of the penalties and hence force the provider to choose less risky operation modes. This risk aversion based on the penalty is indirectly priced into the service price. For low penalties the behavior is similar to Figure 9. However, with the increasing penalty, the provider switches earlier to the better infrastructures and asks for significantly higher prices.

Next to this behavior, throughout the last three figures we could see occasional kinks in the revenue/ penalty figures. They occur if the provider decides to provide more resources to the system in order to achieve better system reliability. However, the additional costs need to be covered by a higher price. As mentioned earlier, this happens if the additional profit based on the improved reliability exceeds the additional cost. Finally, we compare the elastic operation mode with the classic static operation of an information system. In this scenario, we determine the expected profit in dependence of the revenue. Furthermore, we use our previous scenario and define the penalty as the triple price. The SLA lifetime is set to T = 4d. The static operation mode is presented by the red line. Evidently, it is infeasible to operate the service for revenues smaller than $560, as the cost of operation and expected penalties exceed the revenue. In this scenario, the static system consists of 10 resource units.

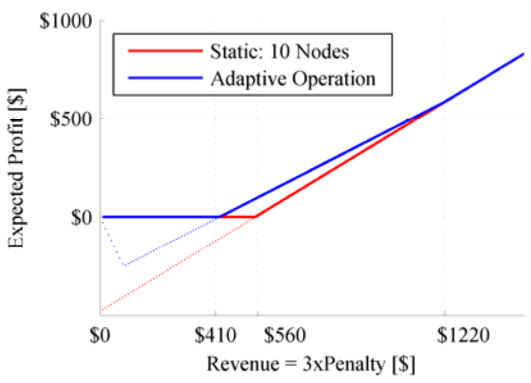


Figure 12: Static vs. Elastic Operation

In the resource adaptive operation mode, the service operation becomes already economical at price of $410. Up to revenue of $1210 and a penalty of $3630, the profit optimal operation strategy is to choose an operation strategy with a higher failure risk than that of the static operation mode. In this area, the system is able to comply with the SLA with a probability of 93.4% percent. If the revenue is beyond this threshold, the reliability goes up to 96.8% and the adaptive and static operation strategies are identical, as from an economic point of view it does not make sense to take the risk of elastic operation, as SLA violations become too costly.

Overall the most important observation of this scenario is that resource adaptive operation modes can provide the same economic performance as the static operation modes at lower costs of operation. In this scenario, the provider can save up to 30% of operation costs by using the more risky adaptive operation mode, however at the price of more frequent SLA violations.

5. Conclusion

In this paper, we presented our novel SLA design model, which provides a new perspective on the management of elastic enterprise information systems in clouds. The model utilizes various low-level concepts from the field of system management and reliability to determine the risk of information system operation. On this base, our model derives the relation between the quality of service and the cost of operation and thus helps service providers to design their service offerings economically.

In the evaluation, we have shown that the deliberate choice of a higher operational risk may lead to higher provider profits. Furthermore, we presented how our model can be used to support the pricing of a service offering and how to determine if a customer-proposed SLA should be accepted. Our model further supports the service provider in selecting the optimal operation environment and strategies for his service offerings.

In one of our ongoing research projects, we developed an automated resource management system for elastic applications. Our next step will be to include our SLA design model into this management framework to automatically choose profit optimal and risk adjusted operation strategies. In the current version, our model only assumes critical failures. In our future work, we also plan to model different failure types with different impacts on the SLA compliance.

6. References

Ardagna, D., Trubian, M. & Zhang, L., 2007. SLA based resource allocation policies in autonomic environments. *Journal of Parallel and Distributed Computing*, 67(3), pp.259-270. Available at: http://dx.doi.org/10.1016/j.jpdc.2006.10.006 [Accessed July 16, 2010].

Bailey, M., 2009. *The Economics of Virtualization: Moving Toward an Application-Based Cost Model*, Available at: http://www.vmware.com/files/pdf/Virtualization-application-based-cost-model-WP-EN.pdf.

Buyya, R. et al., 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6), pp.599-616. Available at: http://dx.doi.org/10.1016/j.future.2008.12.001 [Accessed April 30, 2011].

Chandra, A., Gong, W. & Shenoy, P., 2003. Dynamic Resource Allocation for Shared Data Centers. *Quality of Service — IWQoS 2003*, Volume 270, pp.381-398. Available at: http://www.springerlink.com/content/h56r570l4u707466.

Cohen, I. et al., 2004. Correlating instrumentation data to system states: a building block for automated diagnosis and control. *Operating Systems Design and Implementation*, p.16. Available at: http://portal.acm.org/citation.cfm?id=1251270 [Accessed September 15, 2010].

Deloitte Consulting, 2005. *Calling a Change in the Outsourcing Market*, New York, USA. Available at: http://www.deloitte.com/assets/Dcom-Luxembourg/Local Assets/Documents/Global_brochures/us_outsourcing_callingachange.pdf.

Du, A.Y. et al., 2008. Capacity Provision Networks: Foundations of Markets for Sharable Resources in Distributed Computational Economies. *Information Systems Research*, 19(2), pp.144-160. Available at: http://dl.acm.org/citation.cfm?id=1528684.1528687 [Accessed July 15, 2011].

Gmach, D. et al., 2009. Resource pool management: Reactive versus proactive or let's be friends. *Computer Networks*, 53(17), pp.2905-2922. Available at: http://dx.doi.org/10.1016/j.comnet.2009.08.011.

Gmach, D. et al., 2007. Workload Analysis and Demand Prediction of Enterprise Data Center Applications. *2007 IEEE 10th International Symposium on Workload Characterization*, pp.171-180. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4362193.

Goolsby, K., 2007. How to rebound from a failed outsourcing relationship. *Outsourcing Journal*. Available at: www.outsourcing-journal.com/jan2007-rebound.html.

Hedwig, M., Malkowski, S. & Neumann, D., 2009. *Taming Energy Costs of Large Enterprise Systems Through Adaptive Provisioning*, Available at: http://aisel.aisnet.org/icis2009/140/.

Hedwig, M., Malkowski, S. & Neumann, D., 2010. *Towards Autonomic Cost-Aware Allocation of Cloud Resources*, Available at: http://aisel.aisnet.org/icis2010_submissions/180 [Accessed January 31, 2011].

KPMG, 2007. *Strategic evolution a global survey of sourcing today.*, Sydney, Australia. Available at: http://www.kpmg.com.au/Portals/0/rasita_strategic-evolution200701.pdf.

Koomey, J.G., 2007. Estimating total power consumption by servers in the U.S. and the world. *World*. Available at: http://www.greenbiz.com/research/report/2007/09/12/estimating-total-power-consumption-servers-us-and-world.

Malkowski, S. et al., 2011. Automated control for elastic n-tier workloads based on empirical modeling. In *Proceedings of the 8th ACM international conference on Autonomic computing - ICAC '11*. New York, New York, USA: ACM Press, p. 131. Available at: http://dl.acm.org/citation.cfm?id=1998582.1998604 [Accessed August 29, 2011].

Mituzas, D., 2011. Wikistats. Available at: http://dammit.lt/wikistats/ [Accessed August 30, 2011].

Padala, P. et al., 2009. Automated control of multiple virtualized resources. *Proceedings of the fourth ACM european conference on Computer systems - EuroSys '09*, p.13. Available at: http://portal.acm.org/citation.cfm?doid=1519065.1519068.

Powers, R., Goldszmidt, M. & Cohen, I., 2005. Short term performance forecasting in enterprise systems. *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining - KDD '05*, p.801. Available at: http://portal.acm.org/citation.cfm?doid=1081870.1081976.

Raimondi, F., Skene, J. & Emmerich, W., 2008. *Efficient online monitoring of web-service SLAs*, New York, New York, USA: ACM Press. Available at: http://dl.acm.org/citation.cfm?id=1453101.1453125 [Accessed June 23, 2011].

Sahni, M. & Tan, L.F., 2011. *APEJ CIOs will be the Rising Stars in the Corporate Hierarchy as they Drive ROI-Led Transformation Initiatives in 2011, says IDC*, Available at: http://www.idc.com/AP/pressrelease.jsp?containerId=prSG22698011.

Schulz, G., 2009. *The Green and Virtual Data Center*, Boston, MA, USA: Auerbach Publications.

Sen, S., Raghu, T.S. & Vinze, A., 2010. Demand Information Sharing in Heterogeneous IT Services Environments. *Journal of Management Information Systems*, 26(4), pp.287-316. Available at: http://mesharpe.metapress.com/openurl.asp?genre=article&id=doi:10.2753/MIS0742-1222260410 [Accessed August 28, 2011].

Skitał, Ł. et al., 2008. *Parallel Processing and Applied Mathematics* R. Wyrzykowski et al., eds., Berlin, Heidelberg: Springer Berlin Heidelberg. Available at: http://www.springerlink.com/content/xp280k0v5x5320v8/ [Accessed August 28, 2011].

Sullivan, T. (Gartner), 2009. Gartner: Cloud spending to skyrocket in 2009. Available at: http://www.infoworld.com/t/platforms/gartner-cloud-spending-skyrocket-in-2009-826 [Accessed April 30, 2011].

Susarla, A. & Barua, A., 2009. Contracting Efficiency and New Firm Survival in Markets Enabled by Information Technology. *Information Systems Research*, 22(2), pp.306-324. Available at: http://dl.acm.org/citation.cfm?id=2000434.2000441 [Accessed July 25, 2011].

Thibodeau, P., 2011. The race to cloud standards gets crowded. Available at: http://www.infoworld.com/d/cloud-computing/the-race-cloud-standards-gets-crowded-170524 [Accessed August 24, 2011].

Urgaonkar, B. et al., 2007. Analytic modeling of multitier Internet applications. *ACM Transactions on the Web*, 1(1), p.2-es. Available at: http://portal.acm.org/citation.cfm?doid=1232722.1232724.

Urgaonkar, B. et al., 2008. Agile dynamic provisioning of multi-tier Internet applications. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 3(1). Available at: http://portal.acm.org/citation.cfm?id=1342172.

Yeo, C.S. & Buyya, R., 2007. Integrated Risk Analysis for a Commercial Computing Service. Available at: http://www.computer.org/portal/web/csdl/doi/10.1109/IPDPS.2007.370241 [Accessed April 30, 2011].

Yu, J. & Buyya, R., 2006. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Scientific Programming*, 14(3,4), pp.217-230. Available at: http://dl.acm.org/citation.cfm?id=1376960.1376967 [Accessed August 28, 2011].

VITA

The vita includes personal data, which have been removed from the online version.

Referred Research Publications

Malkowski, S., **Hedwig, M.**, Parehk, J., Pu, C. & Sahai, A., 2007. *Bottleneck Detection Using Statistical Intervention Analysis,* 18th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management Managing Virtualization of Networks and Services (DSOM'07).

Bodenstein, C., Püschel, T., **Hedwig, M**. & Neumann, D., 2009. *DEEP-SaM - Energy-Efficient Provisioning Policies for Computing Environments*. Lecture Notes In Computer Science; Vol. 5745.

Schulz, F. ,Momm, C., Westermann, D., Blau, B., Michalk, W., **Hedwig, M**., Rolli, D., Afaghi, O. K. & Schmidt, A., 2009. *ValueGrids: Using Grids in Dynamic Service Value Networks*. In Proceedings of the Cracow Grid Workshop 2009 (CGW'09).

Hedwig, M., Malkowski, S. & Neumann, D., 2009. *Taming Energy Costs of Large Enterprise Systems Through Adaptive Provisioning*, In the Proceedings of the International Conference on Information Systems (ICIS 2009).

Malkowski, S., **Hedwig, M.**, Jayasinghe, D., Park, J., Kanemasa, Y. & Pu, C., 2009. *A New Perspective on Experimental Analysis of N-tier Systems: Evaluating Database Scalability, Multi-bottlenecks, and Economical Operation*. In the Proceedings of the CollaborateCom'09.

Malkowski, S., **Hedwig, M**. & Pu, C., 2009. *Experimental Evaluation of N-tier Systems: Observation and Analysis of Multi-Bottlenecks*. In the Proceedings of the 2009 IEEE 12th International Symposium on Workload Characterization (IISWC '09).

Hedwig, M, Malkowski, S., Bodenstein, C. & Neumann, D., 2010. *Datacenter Investment Support System (DAISY)*, In Proceedings of the 43rd Annual Hawaii International Conference on System Sciences (HICSS-43).

Malkowski, S., Jayasinghe, D., **Hedwig, M**., Park, J., Kanemasa, Y. & Pu, C., 2010. Empirical Analysis of Database Server Scalability Using an N-tier Benchmark With Read-intensive Workload. In the Proceedings of the 2010 ACM Symposium on Applied Computing (SAC'10).

Malkowski, S., **Hedwig, M**., Jayasinghe, D., Pu, C. & Neumann, D., 2010, CloudXplor*: a tool for configuration planning in clouds based on empirical data*. In the Proceedings of the 2010 ACM Symposium on Applied Computing (SAC'10).

Hedwig, M., Malkowski, S. & Neumann, D., 2010. *Towards Autonomic Cost-Aware Allocation of Cloud Resources*, In the Proceedings of the International Conference on Information Systems (ICIS 2010).

Bodenstein, C., **Hedwig, M.** & Neumann, D., 2011, *Low-energy automated scheduling of computing resources*. In Proceedings of the 1st ACM/IEEE workshop on Autonomic computing in economics (ACE'11).

Malkowski, S., **Hedwig, M.**, Li, J., Pu, C. & Neumann, D., 2011, *Automated control for elastic n-tier workloads based on empirical modeling*. In Proceedings of the 8th ACM international conference on Autonomic computing (ICAC 2011).

Qiu, X., **Hedwig, M.**, Neumann, D., 2011, *SLA Based Dynamic Provisioning of Cloud Resource in OLTP Systems.* 10th Workshop on E-Business (WEB 2011).

Bodenbenner, P., **Hedwig, M.**, Neumann, D., *Are Personalized Recommendations the Savior for Online Content Providers?* 10th Workshop on E-Business (WEB 2011)

Bodenstein, C., **Hedwig, M.** & Neumann, D., 2011, *Strategic Decision Support for Smart-Leasing Infrastructure-as-a-Service.* In the Proceedings of the International Conference on Information Systems (ICIS 2011).

Hedwig, M., Malkowski, S. & Neumann, D., 2011. *Integrated SLA Management for Green Information Systems.* In the Proceedings of the International Conference on Information Systems (ICIS 2011).

Hagenau, M., Liebmann, M., **Hedwig, M.**, & Neumann, D., 2012. *Automated news reading: Stock Price Prediction based on Financial News Using Context-Specific Features.* In Proceedings of the 45th Annual Hawaii International Conference on System Sciences (HICSS-45).

Hedwig, M., Malkowski, S. & Neumann, D., 2012. *Risk-Aware Service Level Agreement Design for Enterprise Information Systems.* In Proceedings of the 45th Annual Hawaii International Conference on System Sciences (HICSS-45).

(Underlined publications are included or partially included in the thesis)