
Towards energy-aware scheduling in data centers using machine learning

Josep Lluís Berral, Íñigo Goiri, Ramon Nou,
Ferran Julià, Jordi Guitart, Ricard Gavaldà, and
Jordi Torres

Universitat Politècnica de Catalunya
BSC-CNS, Barcelona Supercomputing Center

eEnergy'10 - April 2010



Context: Energy, Autonomic Computing and Machine Learning

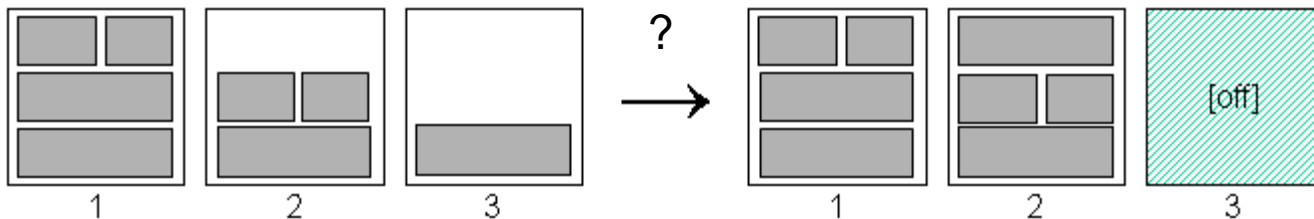
- Keywords:
 - Autonomic Computing (AC): Automation of management
 - Machine Learning (ML): Learning patterns and predict them
- Applying AC and ML to energy control:
 - **Self-management** must include **energy policies**
 - **Optimization** mechanisms are becoming more **complex**
 - ... and they can be improved through **automation** and **adaption**
- Challenges for autonomic energetic management:
 - Datacenters policies require adaption towards constant optimization
 - Complexity can be saved through modeling and learning
 - If a system follows any pattern, maybe ML can find an accurate model to help the decision makers and improve policies

Introduction

- Self-management looking towards Energy Saving:
 - Apply the well-known consolidation strategy
- Consolidation strategy:
 - Reduce the turned on machines grouping tasks in less machines
 - Turn off as many IDLE machines as possible (but not all!)
- Main Contributions
 - Consolidate tasks in a datacenter environment
 - Predict information *a priori* to solve uncertainty and “play it safe”
 - Design adequate metrics to compare consolidation solutions
 - Turn on/off machines from SLA vs. Power trade-off method

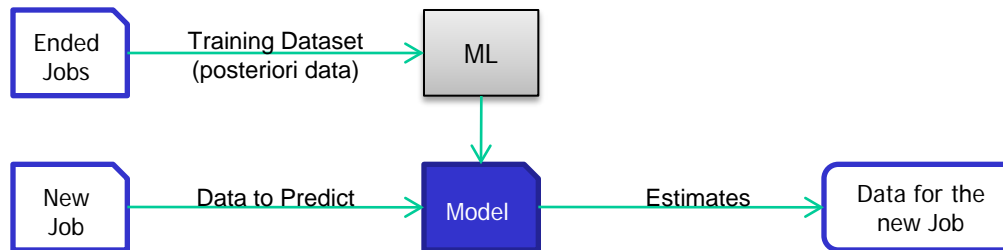
Energy Aware Scheduling

- Consolidation
 - Execute all tasks with the minimum amount of machines
 - Unused machines are turned off
 - Known policies: Random, Greedy policies, (Dynamic) Backfilling
- Policies and Constraints
 - SLA fulfillments must not degrade excessively
 - Operations must reduce or maintain energy consumption
 - Turn off as many machines as possible



EAS: Machine Learning application (I)

- Prediction *a priori* :
 - Deal with uncertainty
 - Anticipate *future* information
- Applying Machine Learning:
 - Relevant variables for decision making only available *a posteriori*
 - ML creates a model from past examples



- Desired information *a priori* :
 - SLA fulfillment level: i.e. we don't know the exact finish time per task
 - Consumption: i.e. we don't know the consumption before placing a task
- Learn a model to induce:
 - <Info. Running tasks, Info. Host> → <SLA fulfillment, Power Consumption>

EAS: Machine Learning application (II)

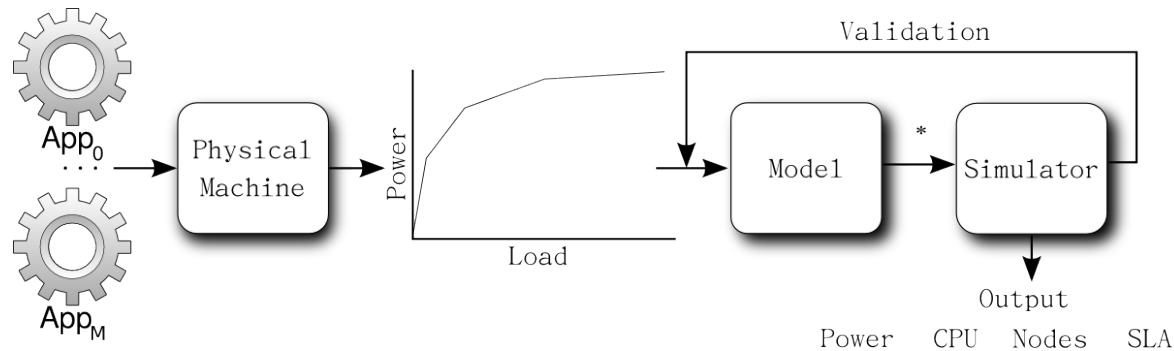
- Information “a posteriori”
 - R_h : Average SLA fulfillment level of jobs in host
 - C_h : Host consumption
 - Finished jobs: Information about ended jobs
 - Host: Information about host capabilities
- Learn a model to induce
 - $\langle \text{Running jobs, Host} \rangle \rightarrow \langle R_h, C_h \rangle$
- Used Variables
 - “Post-mortem” data:
 - Finished Job: $\langle \text{Job}_{\text{Info}}, T_{\text{start}}, T_{\text{end}}, T_{\text{user}}, \text{SLA}_{\text{Fact}} \rangle \rightarrow R_j$
 - Host Consumption: $\langle \text{Usage}_{\text{Res}} \rangle \rightarrow C_h$
 - Available data:
 - Running Job: $\langle \text{CPU}_{\text{Usage}}, T_{\text{start}}, T_{\text{now}}, T_{\text{user}}, \text{SLA}_{\text{Fact}} \rangle \rightarrow R_j$
 - Host Consumption: $\langle \text{CPU}_{\text{Available}} \rangle \rightarrow C_h$
 - Host SLA fulfillment: aggregation of $R_j \rightarrow R_h$

EAS: Machine Learning application (III)

- **Backfilling and Dynamic Backfilling policies:**
 - Purpose: fill turned on hosts before starting off-line ones
 - When a task enters, it is always put on the most fillable host
 - At each scheduling round, move tasks to get more consolidation
- **Applying Machine Learning:**
 - We learn the SLA fulfillment impact and consumption impact, for each past schedule
 - For each possible task allocation $\langle \text{host}, \text{jobs on host} + \text{new job} \rangle$:
 - Estimation of resulting SLA fulfillment
 - Estimation of resulting power consumption
 - If they don't degrade, allocation is viable
 - Dynamic Backfilling: Change the static data by estimated data

Simulation and Metrics

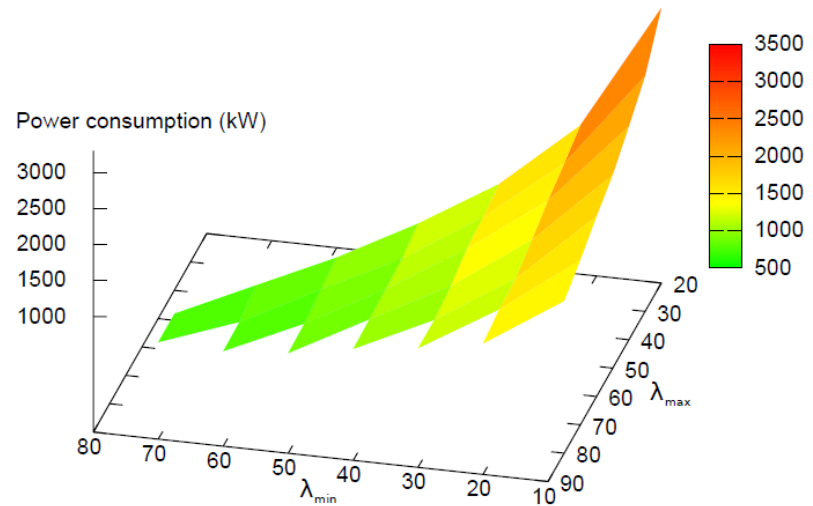
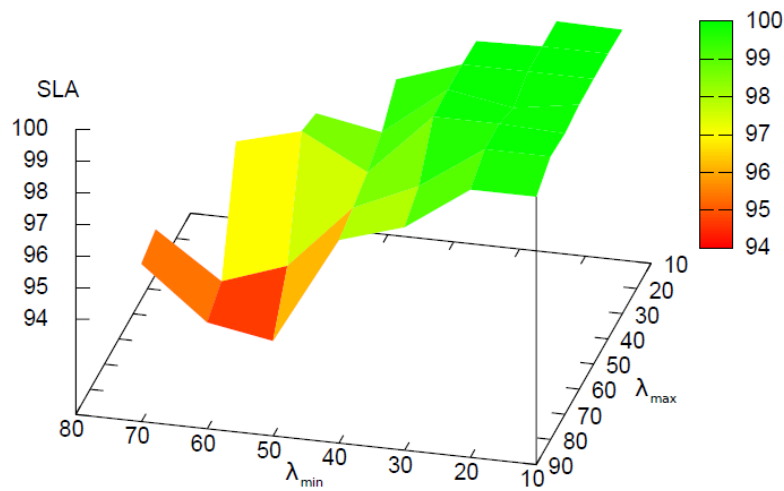
- Self-created simulator:
 - Simulates a data center able to execute tasks according to different scheduling policies
 - Takes into account CPU consumption and energy
 - Able to turn on/off simulated machines



- Metrics:
 - There is no standard approach to compare power efficiency
 - We introduce metrics to compare adaptive solutions:
 - Working nodes, Running nodes, CPU usage, Power consumption, SLA fulfillment level...

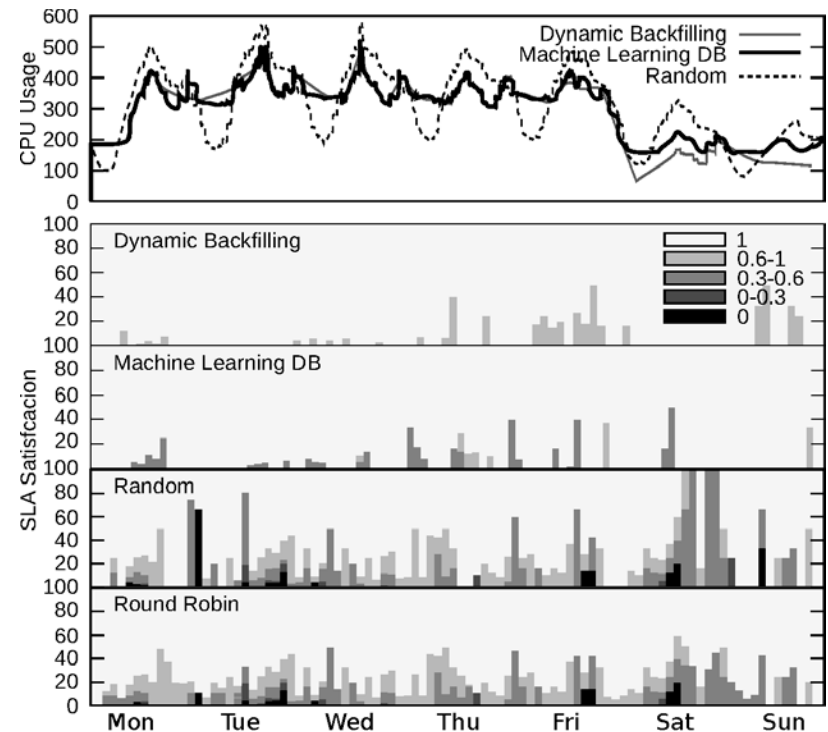
Evaluation (I): Shutting down machines

- Power vs SLA fulfillment trade-off
 - Determine when to shut down IDLE nodes, and turn on new ones
- Find the adequate number of IDLE on machines
 - It depends on the number of running tasks
 - Determine range of IDLE machines (minimum and maximum)
- Trade-off between energy and required resources
 - At what load start off-line machines, or shut down IDLE ones



Evaluation (II): Consolidation

- Experimental Environment
 - Simulated datacenter with 400 hosts (4 CPU per host)
 - Workload: fixed CPU size tasks and variable CPU size tasks
 - Use of Linear Regression and M5P for SLA and Power prediction
- Experimental Results
 - Consolidation techniques perform better than the other techniques:
 - Backfilling & Dynamic BF
 - SLA fulfillment around 99%
 - CPU utilization more stable and lower power consumption



Evaluation (III): Machine Learning

- Experimentation Results (II)

- Dynamic BF + ML performs better, having uncertainty (service and heterogeneous workloads)
- Accuracy around 98.5% on predictions
- Detail: Values with highest estimation always had highest accuracy

	Working nodes (avg)	Running nodes (avg)	Power (kwh)	SLA (%)
Grid workload				
Round Robin	16.11	41.37	1696.66	85.99
Dynamic Backfilling	9.91	26.46	1118.86	100.00
Machine Learning DB	15.04	37.92	1574.78	99.69
Service workload				
Round Robin	290.99	400.00	19761.54	100.00
Dynamic Backfilling	108.79	352.88	16229.22	100.00
Machine Learning DB	99.61	270.50	13673.71	100.00
Heterogeneous workload				
Round Robin	260.66	400.00	19713.72	94.20
Dynamic Backfilling	111.03	329.07	16214.49	99.59
Machine Learning DB	124.20	307.89	15110.33	98.63

Conclusions and Future Work

- **Challenge and Contribution**
 - Vertical and “intelligent” consolidation methodology
 - Metrics to evaluate different consolidation approaches
 - Predict application SLA timings and power consumption to decide scheduling
- **Experimentation Results**
 - Consolidation aware techniques:
 - Improve power efficiency
 - Compare backfilling with “standard” techniques
 - Machine Learning method:
 - Close to consolidation techniques
 - Better when information is inaccurate
- **Current and Future Work**
 - More complex SLA fulfillment (response time, throughput, ...)
 - More complex Resource elements (CPU, memory, I/O elements)
 - More elaborated Policy optimization (utility functions)
 - Addition of virtualization overheads

Thank you for your attention