Long-term SLOs for reclaimed cloud computing resources

Marcus Carvalho
Universidade Federal de Campina Grande, Brazil
marcus@lsd.ufcg.edu.br

Walfredo Cirne
Google Inc., USA
walfredo@google.com

Francisco Brasileiro
Universidade Federal de Campina Grande, Brazil
fubica@dsc.ufcg.edu.br

John Wilkes
Google Inc., USA
johnwilkes@google.com

Abstract

The elasticity promised by cloud computing does not come for free. Providers need to reserve resources to allow users to scale on demand, and cope with workload variations, which results in low utilization. The current response to this low utilization is to re-sell unused resources with no Service Level Objectives (SLOs) for availability. In this paper, we show how to make some of these reclaimable resources more valuable by providing strong, long-term availability SLOs for them. These SLOs are based on forecasts of how many resources will remain unused during multi-month periods, so users can do capacity planning for their long-running services. By using confidence levels for the predictions, we give service providers control over the risk of violating the availability SLOs, and allow them trade increased risk for more resources to make available. We evaluated our approach using 45 months of workload data from 6 production clusters at Google, and show that 6–17% of the resources can be re-offered with a long-term availability of 98.9% or better. A conservative analysis shows that doing so may increase the profitability of selling reclaimed resources by 22–60%.

Categories and Subject Descriptors D.4.8 [*Operating Systems*]: Performance—Modeling and prediction; K.6.2 [*Management of Computing and Information Systems*]: Installation management—Performance and usage measurement, Pricing and resource allocation

General Terms Management, Performance, Measurement

SoCC '14, 3-5 Nov. 2014, Seattle, Washington, USA.

ACM 978-1-4503-3252-1/14/11.
http://dx.doi.org/10.1145/2670979.2670999

Keywords Cloud computing, Capacity planning, Quality of Service

1. Introduction

Although marketing campaigns for cloud Infrastructure as a Service (IaaS) praise its "infinite elasticity", this clearly cannot be the case: a provider's resources are finite. In practice, at any point in time cloud providers impose a *resource ceiling* for each user to bound the maximum amount of resources that the user will be granted. Users typically get a small ceiling by default and can request or negotiate increases with the service provider.

Provisioning enough capacity to guarantee that all users can be granted their ceilings at all times is both expensive and wasteful: for most applications, the average resource requirement is much lower than the peak [24]. A few users may wish to pay to have dedicated resources always standing by, but most can accept weaker promises in return for lower prices. As long as all the users' loads do not peak at the same time, cloud providers can benefit from statistical multiplexing to provision fewer resources than the sum of all users' ceilings [21], thereby saving money – but some spare capacity is still required to ensure they rarely deny requests for resources [9]. Spare capacity is also required to handle failures, expected market growth, and the seasonal load fluctuations that affect many applications (e.g., during November and December in the USA).

A common way of increasing data center utilization (and thus service provider profitability) is to offer this temporarily-unused capacity in an *opportunistic* way, with essentially no Service Level Objectives (SLO) [20]. Unfortunately, this absence of SLOs restricts the applications that can benefit from these reclaimable resources, and so resources offered opportunistically are usually sold at lower prices than regular ones (e.g., Amazon Spot Instances [3]). Such resources are acceptable for short-lived batch jobs that can often be deferred or restarted, and it is not critical for

them to have unknown (and occasionally long) unavailability periods.

However, the primary target audience for the work we describe here is cloud users that run services that stay up for weeks or months at a time [24]. Such services are common: most interactions on the web are with them. Without long-term availability SLOs, such users could not be sure resources would be available when they were needed – e.g., to match expected workload growth over future months or to quickly restart VMs in eventual service disruption – so they would need to pay for higher-quality SLOs. Our goal is to improve the availability SLOs of reclaimed resources to the point where they become useful for long-running service jobs, especially ones that can accept slightly degraded resources but still need moderate availability guarantees – e.g., non-user facing pipelines, as web indexing and video transcoding.

In this paper, we show how to provide such strong, long-term availability SLOs for a portion of the unused resources through a new *economy* service class. Even though resources offered via this class can be revoked if necessary, the chance of this happening is low enough to make them useful – even over a multi-month duration. We do this by predicting (estimating) the fraction of unused resources that will almost certainly not be required in the future, based on historical usage patterns. We use confidence levels to deal with uncertainty in these predictions and make appropriately-conservative predictions. Different confidence levels permit trade-offs between the risk of SLO violations and the amount of resources that can be offered this way.

We evaluated our method using workloads from 6 production clusters at Google, gathered over 45 months. Our results show that we can re-offer up to 17% of resources with 6-month availability SLOs of 98.9%. Moreover, by doing this, we show the IaaS provider can increase its profitability from selling these reclaimed, unused resources by up to 60% compared to offering them with no availability SLOs.

We make the following contributions in this paper:

- Analyze a set of real-world cloud workloads for ways to improve resource utilization.
- Introduce a novel Economy class service that is backed by reclaimed resources.
- Describe a prediction process that balances the amount of resources that can be offered via this new service class against the risks of SLO violations.
- Test the scheme by instantiating it with well-known prediction techniques, and assess its performance using data from production systems.
- Demonstrate that profitability can be increased by modeling the effects of offering economy class resources.

The rest of the paper is organized as follows. Section 2 describes our system model. Section 3 analyses a cloud computing workload and shows the potential gains that reclaimed resources could yield, especially if these resources could be offered with long-term SLOs. Section 4 describes the prediction process we used to estimate the amount of cloud capacity that can be offered via Economy class, and Section 5 evaluates it using data from 6 large production clusters. Section 6 discusses related work, and we present our conclusions in Section 7.

2. System model

In this paper we consider the management of resources in a cloud IaaS provider, whether public or private clouds. We begin with describing the system model that underlies our work.

A cloud *resource* means any cloud infrastructure component such as CPU cores, memory space, disk capacity, disk access time, or a combination of these components – typically in a Virtual Machine (VM) or container. The *capacity* of a cloud infrastructure service is the total amount of resources available for users. Users make *requests* for resources to scale up their applications, and (eventually) to scale them down. Each request specifies a *limit* for each resource, which is the maximum amount of the resource that is needed. Many cloud providers only offer pre-defined "bundles" of resource types, such as 4 or 8GB of RAM with 2, 4, or 8 CPU cores – the user can't explicitly specify what they want, but has to map it into the next larger bundle.

Users have pre-defined *ceilings* for each resource type, which is the maximum amount of the resource they can obtain from the cloud at one time. For example, Amazon EC2 defines a default ceiling of 20 VMs per user for On-demand and Reserved instances [1]. A provider will refuse any request that would make the sum of limits for the user higher than the user's ceiling for any resource in the request. If this doesn't happen, the request is *valid*, and the provider *allocates* the requested amount of resources, provided there is enough capacity available. On request, the provider can make *reservations* of resources to a user to increase the chance that a future demand can be met.

Cloud users have diverse Quality of Service (QoS) requirements and budgets for their applications, so cloud providers usually offer different *service classes* that provide resources with different combinations of SLOs and pricing, allowing users to choose what best matches their needs. Failure to meet an SLO will usually provoke financial penalties for the provider, as specified in the provider's Service Level Agreement (SLA). We find it helpful to tease apart the SLOs offered by providers into two parts:

- *Obtainability*: the probability that any request to obtain new resources, within the user's ceilings, made during the validity of the contract between user and provider, will be granted by the service provider.

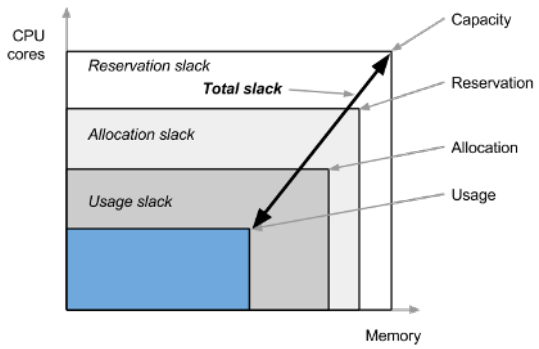

Figure 1. How resources are managed, and the different kinds of slack.

- *Availability*: the probability that a resource that has been granted will be remain operational and accessible during the validity of the contract.

We see three common service classes today, characterized by how the above mentioned SLOs are treated:

- *Reserved*: users will have a very high obtainability SLO ($\approx 100\%$) for resources up to their ceiling, and the obtained resources will have a high availability SLO (e.g., $\geq 99\%$). To meet these, the provider will typically reserve resources for this class, although specific resource instances may not be allocated [13]. Amazon Reserved Instances [3] is an example of this class.
- *On-demand*: users will usually be able to get resources up to their ceilings (e.g., an obtainability SLO $\geq 99\%$), and the acquired resources will have a similar availability to the Reserved class SLO. The provider typically reserves resources to cover only a subset of the sum of users' ceilings for this class. The precise amount is a function of the target availability SLO and its expectation of the aggregated demands of the users. Resources offered in the On-demand class are typically cheaper than those offered in the Reserved one, which can reduce costs for applications that can tolerate some uncertainty. Amazon On-demand Instances [3] are an example of this class of resources.
- *Opportunistic*: these resources are typically offered without obtainability or availability SLOs or SLA penalties (sometimes called "no QoS"): the resources can be preempted at any time to meet Reserved and On-demand requests. The Opportunistic class usually has the lowest price. Amazon Spot Instances [3] is an example of this class.

In a cloud infrastructure that offers only Reserved and On-demand classes, the capacity will not be fully used all the time. These unused resources come from different sources (Figure 1):

- *Reservation slack*: capacity that has not been reserved (i.e., *total capacity* minus sum of *reservations*). Example: spare resources that can be used in the future to admit new users, create or expand reservations, or kept for fault tolerance.
- *Allocation slack*: capacity that has been reserved, but has not yet been allocated to users (i.e., sum of *reservations* minus sum of *limits*). Example: the provider reserves resources to meet SLOs, but a portion of these resources are not assigned to particular VMs. (A temporary reservation is created for every allocation that isn't slotted into a pre-arranged reservation.)
- *Usage slack*: capacity that is not being fully used in the allocated resources (i.e., sum of *limits* minus sum of *usage*). Example: an idle VM that isn't consuming all of the resources allocated to it.

We call the resources left unused from the Reserved and On-demand classes the *total slack*. It is the sum of the three slack types above, which is equal to the difference between the capacity and the usage of resources before any opportunistic or economy work is introduced. The term *resource slack* in this paper will be used to mean the *total slack*, unless we mention a specific slack type.

Reclaiming reservation and allocation slacks so they can be used to accommodate other requests is straightforward, as they haven't been allocated to any VM or container. Reclaiming usage slack is harder, because resources must be extracted from a running VM/OS, or container/application. It requires techniques such as balloon drivers [26].

Currently, cloud providers increase the utilization of their data centers by offering some of these reclaimed resources in the Opportunistic service class. In this paper we propose offering a part of these resources in a new *Economy on-demand* class (*Economy* for short). This has strong SLOs that are good for multiple months: the obtainability SLO is comparable to the one provided by the On-demand class, and the long-term availability one is slightly weaker (e.g., 98.9% rather than 99.95%). Economy class can meet the needs of applications (including long-running services) that cannot run on Opportunistic resources, but do not need the full-strength SLOs of On-demand – and it can do so at significant price savings for the user, but noticeable profitability benefits for the provider.

In the next section, we show that the amount of reclaimable resources is substantial in real-world cloud clusters. Then, we propose a prediction scheme that can estimate, over long-term periods, the unused cloud capacity that can be offered in the *Economy* class, as well as their SLOs.

3. Workload analysis

This section presents an analysis of how the slack resources change over time for workloads from 6 real-world, production cloud computing clusters at Google. The data was gath-

ered from December 2012 to November 2013 at 5-minute intervals.

We first analyze how much of the users' resource ceilings are taken up by allocations for each 5-minute interval in our traces. We define the *ceiling utilization* as the sum of the requested limits divided by the ceilings, and we calculate this for both individual users and the cluster as a whole (i.e., the sum of the total requested limits divided by the the sum of the ceilings). Figure 2 shows a Cumulative Distribution Function (CDF) of CPU ceiling utilization for our clusters.

The median per-user CPU ceiling utilization (dashed line) varies from 30% to 63% for the 6 clusters, but we also see extremes: in 38% of the user ceiling measurements the average CPU ceiling utilization is lower than 1%, and yet in 15% of the measurements more than 99% of the CPU ceiling was allocated. Apparently, users do not do a very good job of using their ceilings, or the ceilings were not correctly set for them, or their ceiling utilization varied over time, with high values being reached only occasionally – or some combination of these factors.

At the cluster level (solid line), the median CPU ceiling utilization for the cluster varies from 55% to 75%, but there are fewer extremes: the cluster CPU ceiling utilization is lower than 1% in less than 1% of the measurements, and going higher than 81% is equally rare. Not surprisingly, the cluster-level aggregation decreases the variation in the overall ceiling utilization, which means that we can make resource reservations for groups of users more efficiently than making decisions for individual users. Since it is rare for the aggregated limits to get close to overall ceiling, the provider does not need to reserve resources equal to the sum of ceilings to provide good obtainability SLOs.

Figure 3 shows the CPU resource slacks described in Section 2. On average, total slack accounts for a significant fraction (57%) of the cluster capacity. Interestingly, the reservation and allocation slacks have higher variance over time than the usage slack.

Figure 4 shows the observed distribution of total slack for CPU, memory and disk. In 99% of the measurements, more than 45% of the CPU, 43% of the memory and 89% of the disk capacity are unused. This is the primary motivation for offering an Opportunistic class. We believe that we can offer part of these reclaimable resources in Economy class, thereby improving both the users' and the service provider's experience.

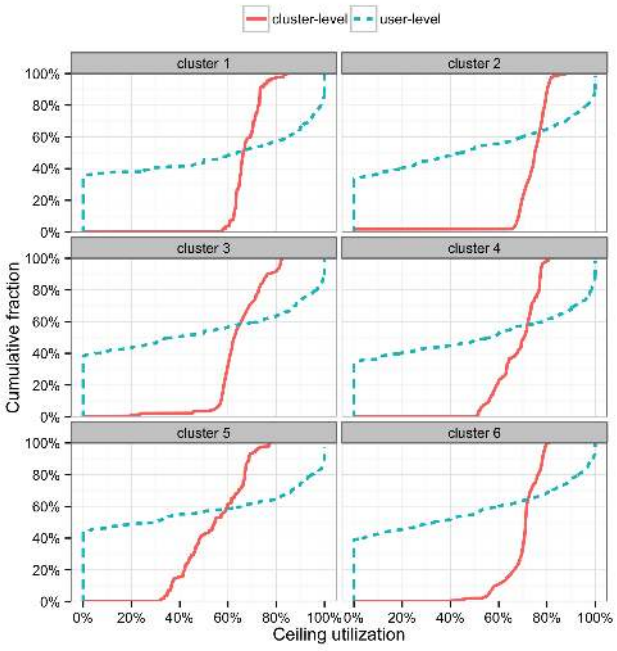

Figure 2. Cumulative Distribution Functions (CDFs) for user and cluster CPU ceiling utilization, measured at 5 minute intervals from December 2012 to November 2013 in 6 separate cloud clusters.

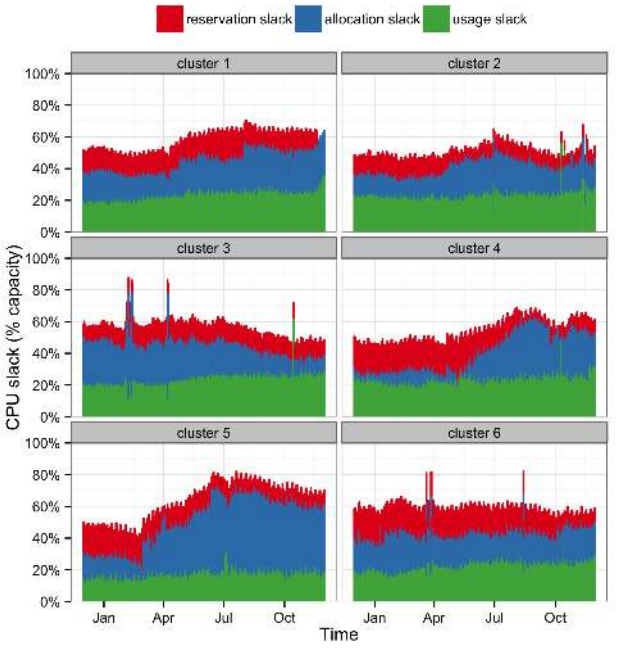

Figure 3. CPU slack over time, normalized to the cluster capacity, at 5 minute intervals from December 2012 to November 2013 in 6 separate cloud clusters.

4. Prediction process

The more reliably we can predict the amount of slack resources, and how unlikely they are to be needed, the better the SLO that can be provided for Economy class. That can lead to higher utilization – and greater revenue. To do this, we propose a prediction process that is comprised of three parts: (i) time-series based forecasting; (ii) predicting confidence intervals; and (iii) using prediction cycles.

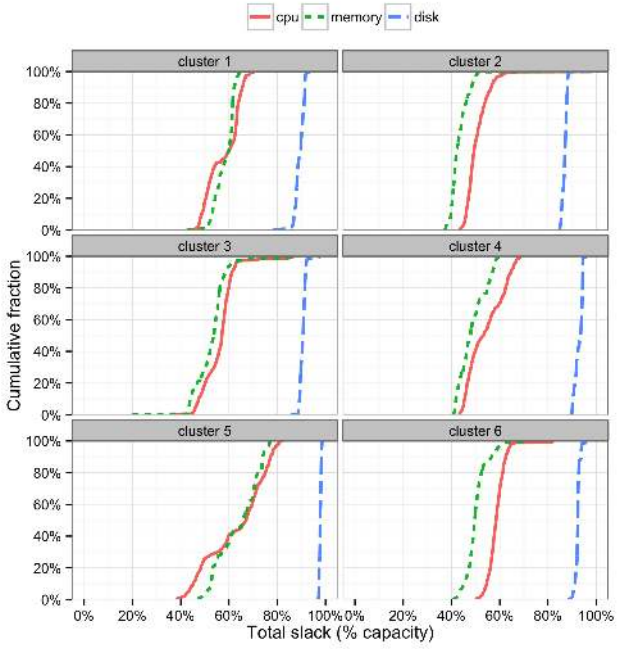

Figure 4. CDF of normalized resource slacks for CPU, memory and disk, at 5 minute intervals from December 2012 to November 2013.

4.1 Time series forecasting

The first step in our process is to predict the minimum value for the total slack in future time periods by deriving a time-series based forecast from historical data. Time is naturally discretized here because slack is measured periodically, so we generate a forecast at time t for a future time $t+h$:

$$\hat{y}_{t+h} = F_h(y_1, y_2, \ldots, y_{t-1}) \quad (1)$$

where h is the prediction horizon, and F_h is the forecasting function that uses as input observations y gathered before time t.

Any forecasting technique (or even a combination of techniques [23]) can be used; the best choice will depend on the data patterns found in the measurements. We tested four widely-used forecasting techniques and picked one that worked well for our workloads. They are:

Mean: the arithmetic mean of the input time series values. This will be over-optimistic in the face of request spikes.

Minimum: the minimum of the input values. This will be overly conservative if there are request spikes.

Auto Regressive Integrated Moving Average (ARIMA): a time series forecasting technique that combines three types of component [7]:

1. A differencing component that removes trends (non-stationarity) from the data by computing the differences between consecutive observations, applied d times.
2. An autoregressive component $AR(p)$, based on a linear regression of the last p observations.
3. A moving average component $MA(q)$, based on a regression model applied to the last q forecast errors.

We used Akaike's Information Criterion (AIC) [5] to pick a good *ARIMA(p, d, q)* model: it estimates the probability of the data arising from each model by using a maximum likelihood estimator, and penalizes models with a large number of parameters [7].

Exponential Smoothing (ETS): a time series forecasting technique that combines models based on the weighted averages of input values, with the weights exponentially decaying for older values. The ETS model has three components: Error correction (E), Trend (T), and Seasonal (S). Each component can be of a certain type: None, Additive, Multiplicative, and other variations. The different combinations of component types results in different ETS methods, where each method have a set of parameters to be estimated. We used the AIC method here, too, to find the best combination of types and parameters [17].

We used the ARIMA and ETS implementations from the R *forecast* package [18].

4.2 Prediction confidence intervals

As part of the prediction, we also generate a *confidence interval*: an estimate of the range of values within which the true value should lie with a certain confidence level (a probability, γ). As our prediction, we pick the smallest amount of resources that fit within the confidence interval for our chosen confidence level. The higher the confidence level, the wider the confidence interval, and the more conservative the predictions. This lets us trade off the quantity of resources that are predicted against their availability: the higher the confidence level, the lower the risk of an availability SLO violation.

The confidence interval calculation relies on the variance of the prediction errors and the confidence level γ. We define the significance level $\alpha = 1-\gamma$, and then the prediction interval for a future time $t+h$ is given by

$$\hat{y}_{t+h} \pm \hat{\sigma} \cdot z_{\alpha/2} \quad (2)$$

where $\hat{\sigma}$ is the estimated standard deviation for the prediction errors, and $z_{\alpha/2}$ is the value for the $100 \cdot \alpha/2$ percentile in the standard normal distribution. Although Equation 2 seems to assume that the forecast errors are normally distributed, it also works well for prediction errors that are not normally distributed [7].

For each forecasting technique, the estimated standard deviation for the prediction errors $\hat{\sigma}$ has to be calculated. A general approach to estimating $\hat{\sigma}$ is to calculate the standard deviation from the prediction errors (residuals) when applying the fitted forecast model to the same data used for training. Closed formulas for estimated variance are also avail-

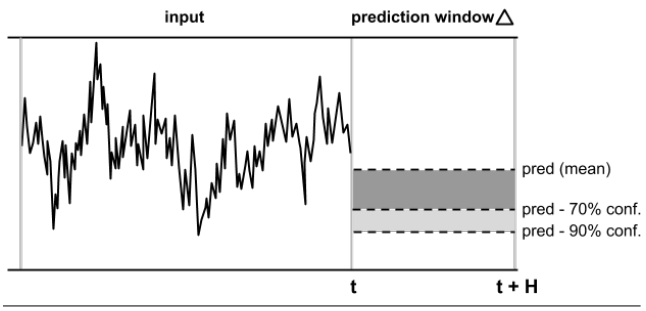

Figure 5. Predictions at time t for the window Δ with different confidence levels.

able for some forecasting techniques and these can produce better estimates.

For the *Mean* method, $\hat{\sigma}$ is estimated as the standard deviation of the input time series. We didn't calculate a confidence interval for the Minimum method, since it's output is already very conservative. We used a closed-formula provided by Box-Jenkins [4] to estimate the variation of prediction errors for the *ARIMA* technique [7], and a formula provided by Hyndman et al. for the *ETS* method [19]. We also applied these calculations as implemented in the R *forecast* package [18].

4.3 Prediction cycles

To provide SLOs for Economy class resources, we need to make predictions for long time windows. But since the capacity and demand for a cloud cluster varies over time there is variance in the amount of slack, so the longer the window, the wider the confidence interval and the smaller the amount of reclaimable resources that will be predicted. Predictions are generated for a time window $\Delta = (t, t+H]$, where t is the time when the prediction is performed and H is the length of the window (the prediction horizon).

After consulting with our customers, we chose to make predictions for a 6-month time window, as this gave them an adequate long-term planning horizon and made a good amount of Economy-class resources available. (Our approach will work for shorter timescales too, but that would not have meet their needs and so was not our focus.) For these experiments, we generated new predictions at the end of each window – i.e., predictions did not overlap in time. We call each one a prediction cycle. The prediction for one such cycle, for the window Δ, is given by

$$\widehat{Y}(\Delta) = \min\left(\hat{y}_{t+h} - \hat{\sigma} \cdot z_{\alpha/2}\right), \quad \forall h \in [1, H] \qquad (3)$$

Notice that we pick the lower bound of the confidence interval as the predicted value in Equation 3. This is shown graphically in Figure 5, which also points out that as the confidence level rises, the width of the confidence interval increases, and the lower bound drops.

5. Evaluation

This section assesses our approach using data from real cloud clusters. It describes the assessment methodology, and then compares the predictors, and analyzes the trade-offs that result from adjusting the confidence levels.

5.1 Methodology

Our evaluation used workload traces from the same clusters at Google that were analyzed in Section 3, but over a longer time period: from April 2010 until December 2013 – almost 4 years. We believe that such long evaluation period, the large scale of the production clusters analyzed, and the diversity of applications running on them make our evaluation representative for cloud computing providers, which can be generalized to other workloads and probably show similar results.

Using 5-minute samples for the predictions would have made the execution time for our experiments prohibitive, so we transformed the original 5-minute trace data into a trace of daily values, by calculating the *minimum* total slack seen in the day for each cluster. This represents the worst case scenario for providing resources in Economy class. The time required to execute all predictions in our evaluation when using daily values was around 2 hours. We believe this coarser granularity does not degrade the prediction quality; as our target is multi-month prediction windows, weekly and monthly fluctuations are more important than daily variations.

We only report total slack measurements for CPU cores in this paper, although we applied the predictions for memory and achieved similar results. We did not generate predictions for disk capacity since it was not a bottleneck resource for these clusters (see Figure 4).

Predictions are executed for every non-overlapping 6-month window in the workload, starting at the 9th month. In each iteration the data is divided into training and test sets, where the test period Δ is always 6 months long and the training data is all the workload data collected up to the start of the test period. The training set is used as input for the forecasting techniques, and the test set is used to evaluate the predictions. After each iteration, the window is advanced 6 months and the previous test set added to the training set. With our 45 months of data, this gives us an initial 9 month training set and 6 prediction cycles with 6-month test sets, except for cluster 3, which has a shorter sample and has only 4 prediction cycles.

The prediction techniques described in Section 4 were applied to the training data and the results are evaluated against the test data. The prediction errors are calculated by subtracting the minimum total slack predicted at time t for the 6-month prediction window $\Delta = (t, t+H]$, as in Equation 3, from the actual values for the total slack at each day $t+h$ in the window, defined as:

$$\varepsilon_{t+h} = y_{t+h} - \widehat{Y}(\Delta), \quad \forall h \in [1, H] \tag{4}$$

Notice that a window Δ only has a single prediction, but H prediction errors, as we have many observations (days) in the test set.

We distinguish positive and negative errors in the evaluation since they mean different things in our context. Positive errors mean that we under-estimated the amount of resources that were available; it's wasteful, but not harmful. Negative errors mean that we over-estimated what was actually available: if these resources were taken up by customers, their availability SLO would not be met, which could result in financial penalties for the service provider. Throughout the text, positive and negative errors will be called *wastage* and *shortage*, respectively.[1]

We varied the confidence level γ from 50% to 90%, after finding that confidence levels outside this range either saw many resource shortages or predicted few resources would be available.

As a comparison point for the predictions with confidence levels, we show the results for the predicted mean future value (i.e., without considering confidence intervals). We also built an *oracle* predictor, which knows the future values for the time-series data and can give the highest prediction value that will have no shortage (i.e., the minimum future value in the prediction window). Obviously, it cannot be implemented in real life. Since there is a daily variation in the amount of slack resources and the oracle only makes a single prediction for the entire prediction window, the oracle will still have wastage of resources.

The time series data used in the predictions were normalized to help the comparison of results between clusters with different sizes, and to obfuscate confidential data. For each prediction applied at time t, we use as input the *relative total slack*, which is calculated by dividing each total slack measurement in the training set by the resource capacity observed at the prediction time t. The relative total slack is used when calculating the prediction errors with Equation 4. Note that if a cluster grew a lot during a prediction window, the normalized total slack could exceed 100% – i.e., become larger than the cluster's entire capacity at the prediction time.

5.2 Prediction results

Figure 6 shows the CDF for the total slack prediction errors for each of the four techniques. As expected, increasing the confidence level moves the curve to the right: as the predicted values get smaller, there's more wastage and fewer shortages.

[1] An interesting question to ask is how many penalties the service provider should be willing to pay in order to increase average utilization. If the only factor was expected profit, paying some penalties would probably be a good idea, but in practice, factors such as the service provider's reputation also matter.

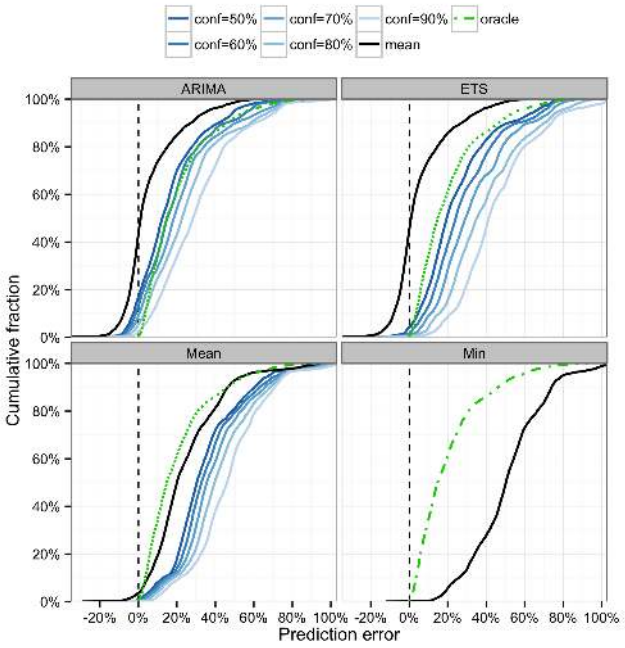

Figure 6. CDF of per-day prediction errors for total CPU slack for 6 month prediction windows, over all clusters. The solid black line is the predicted mean future value (i.e., without considering confidence intervals); lighter lines show the effects of increasing the confidence level. The lighter the line, the higher the confidence level, and consequently the larger the confidence interval (i.e., more conservative predictions). The oracle's predictions are shown as a dashed line; the zero-error case by a vertical line.

Our goal was to pick a forecasting technique that provided a good balance between wastage and shortage for our workloads. Here's what we found.

- Mean presented no resource shortage for most scenarios, but all predictions were too conservative.
- Minimum is simply too conservative.
- ARIMA produced resource shortages even for scenarios with conservative confidence levels.
- ETS provided the best range of options: the resource shortage gradually decrease with increasing confidence level, getting almost no shortage in more conservative scenarios.

Given this data, we chose to use ETS in the rest of our evaluation. Figure 7 shows the relative total slack values and predictions for the ETS predictor over time. We see that the relative total slack presents high variations at some periods. Possible causes for the resource slack peaks could be an increase on the cloud capacity and migration of workloads to other clusters. On the other hand, drops on the total slack could be caused by seasonal demand bursts, new users/services be-

ing assigned to the cluster and a cloud capacity reduction (e.g. machine failure, scheduled repairs, or removing obsolete machines).

Our methodology would continue to work if a different forecasting function gave better predictions for a different workload. A service provider should expect to do the same kind of evaluation that we did here to pick what works best for them. However, we expect that these results are likely to generalize well to other cloud providers because of the length, size, application diversity, and behavior variations of the clusters and their workloads.

Our algorithms are deliberately conservative, in order to deal with uncertainty and avoid the bad consequences of available resources shortage. As a result, the predicted resources shrink with larger variations in the total slack in the training data, longer predictions, and shorter training windows (i.e., fewer samples).

5.3 Capacity planning analysis

What are the trade-offs of using more/less conservative predictions? This section shows how the amount of resources offered as Economy impacts the availability SLOs that could be offered for this class. We use the 6-month prediction of unused resources (total slack) as the amount of reclaimable resources to be offered in Economy class.

We define the *6-month availability* of these resources as the fraction of days in a prediction window with no shortages (i.e., no need to reclaim any resources). This is deliberately conservative: for example, we might not need to reclaim any of the resources offered in Economy class if they were needed on a day when they had not been taken up. Note that we count an entire day as unavailable even if there is only one 5 minute window with resource shortage. Again, this is deliberately conservative because we are targeting users with long-running services. If short outages (e.g., a few minutes) are acceptable, then finer grain measurements would certainly show a higher 6-month availability, making Economy class even more attractive.

The confidence level affects how many resources will be offered in Economy class, and also the 6-month availability for their resources. Figure 8 shows the shape of this trade-off for ETS.

In the most conservative scenario (90% confidence level), the 6-month availability ranged from 98.9% to 100% for the 6 clusters, and the resources offered in Economy class represented 6.7–17.3% of the cluster's capacity. The (not implementable) oracle predictor could reclaim 30.7–52.1% of the cluster's capacity. At the 50% confidence level, the availability was worse (88.7–100%), but the amount of resources that could be offered in Economy class increased to 28.4–40.4%.

For clusters 2, 3, and 6, the availability is almost 100% for all confidence levels. This is a result of the conservative nature of our prediction algorithms, coupled with only moderate variability in the input training data.

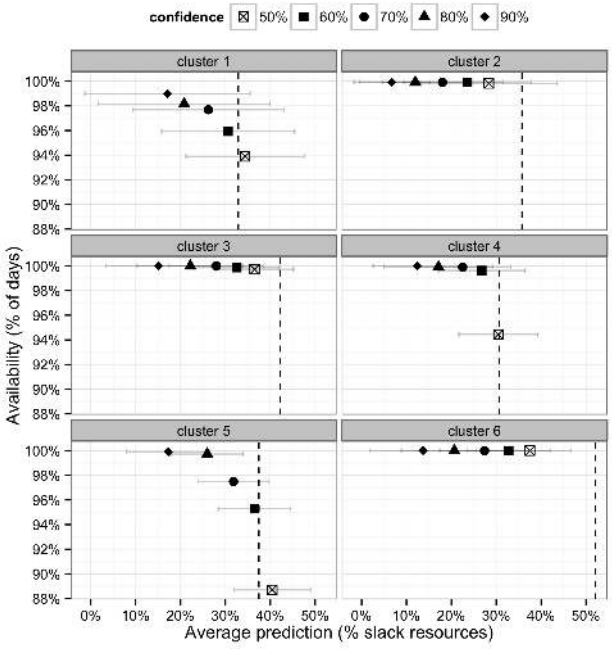

Figure 8. Economy resource quantities predicted by ETS, and their 6-month availability, for different confidence levels. The dashed line is the oracle's prediction, and represents the largest value that could be offered with 100% availability. ETS sometimes predicts more resources than this, which decreases their availability. Error bars show the standard deviations for the predictions.

By comparing the results for the different confidence levels, cloud providers can choose a value that fit their needs, balancing the strength of the availability SLO against the amount of resources to be offered in Economy class.

5.4 Profitability analysis

Would a cloud provider be better off if it sold some of the reclaimable resources in Economy class, or would it be better to sell them all in the Opportunistic one? This section explores several scenarios to answer this question. We first assume that all reclaimable resources offered by the provider are actually consumed by users. Then, we discuss what can make this assumption invalid in practice, and what would be the consequences.

To compare the profits, we first calculate the revenue R_k for a resource class k as:

$$R_k = C \cdot T \cdot f_k \cdot p_k$$

where C is the total resource capacity of the cloud provider, f_k is the fraction of resources allocated for class k, p_k is the price of resources per unit of time for class k, and T is the length of the period the revenue is being calculated in units of time.

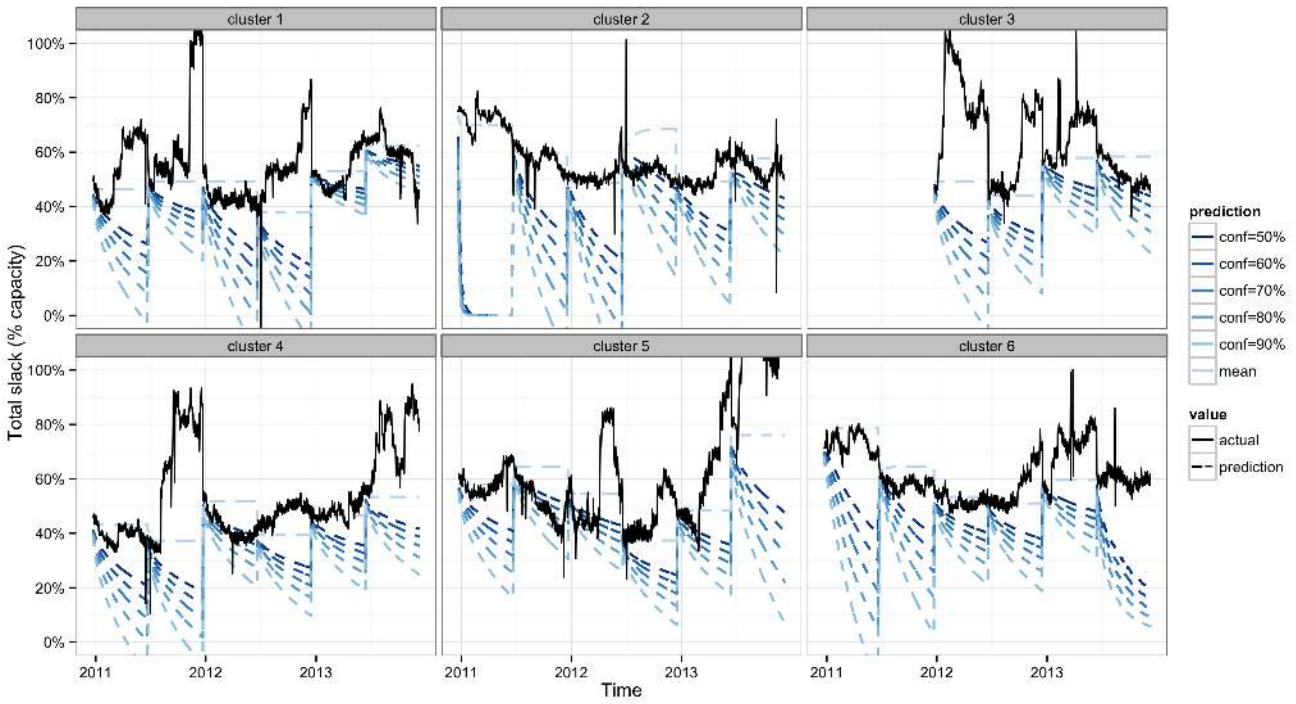

Figure 7. Predicted and actual total slack using ETS. The black (darkest) wiggly solid line shows the actual total slack, and the blue (lighter) dashed lines show the predicted values for the mean and the lower bounds for different confidence levels. The total slack is normalized to the cluster capacity on the day the prediction was applied – the last day of the training set.

If the provider does not offer Economy class, all slack resources will be offered as Opportunistic. The revenue when using only the Opportunistic class for all slack resources is given by:

$$R_{op} = C \cdot T \cdot f_{slack} \cdot p_{op}$$

where f_{slack} is the fraction of slack resources that can be re-offered, and p_{op} is the average price for the Opportunistic resources during the measured period.

When a portion of the slack resources are offered in Economy class, the remaining slack resources are offered in the Opportunistic one. Thus, we calculate the revenue R_e when offering Economy class as:

$$R_e = C \cdot T \cdot [f_e \cdot p_e + (f_{slack} - f_e) \cdot p_{op}]$$

where f_e is the fraction of slack resources to be offered in Economy class, and $(f_{slack} - f_e)$ is the remaining slack resources to be offered in the Opportunistic class.

When offering resources with SLOs, the service provider will pay penalties on SLO violations. For this analysis, we imitate the SLA for Amazon's EC2 cloud service [2]: if service availability is lower than 99% averaged over a month, the service provider pays the user a penalty (the user gets a discount) that is 30% of that month's bill.[2] We define the penalty X_k for the class k as a fraction of the provider's revenue in the measured period. The penalty factor for the Opportunistic class is always zero ($X_{op} = 0$), as the penalty factor X_e for Economy class is calculated as:

$$X_e = \begin{cases} 30\%, & \text{if } a_e \leq 99\% \\ 0, & \text{otherwise} \end{cases}$$

where a_e is the availability measured for this service class.

We define the profit P_k a provider makes with a resource class k as:

$$P_k = R_k \cdot (1 - X_k) - cost$$

Calculating a service provider's costs is a difficult process. However, there is no intrinsic reason to suppose that the costs would be a function of the quality of the resource SLOs, and since the objective of this analysis is to compare the profits when offering different SLO classes, not to measure absolute profits, we simply ignore any such cost differences for the rest of the discussion (i.e., we set $cost = 0$). A more thorough analysis of provider's profit would need to include cost factors such as infrastructure, power, operations staff, marketing, and maintenance.

Limitations in our data cause us to make two very conservative assumptions:

1. The grain of our evaluation data is a single measurement per day, so we assume a worst case scenario in which

[2] The penalties associated with SLO violations for most cloud services are remarkably small.

a single availability violation in a day makes the whole day unavailable. Thus, one single violation in a month would result in 1 day of 30 unavailable, which is already lower than the 99% availability threshold, and would cause penalties for the provider for resources offered in the On-demand or Economy classes.

2. We do not know how many users would be affected by an SLO violation, so we assume that *all* users are affected by *every* unavailability period.

In practice, the actual availability will be significantly better than this, and the penalties would only be paid out to a subset of users. Thus, if finer grain data was available, then the analysis would give higher profitability for Economy class.

We can compare the profit that the provider would have when using only the Opportunistic class for the slack resources to the profit when using Economy class, calculating the profit ratio θ for the two scenarios:

$$\theta = \frac{P_e}{P_{op}}$$

$$\theta = \frac{[f_e \cdot p_e + (f_{slack} - f_e) \cdot p_{op}] \cdot (1 - X_e)}{f_{slack} \cdot p_{op}}$$

Notice that the profit ratio θ is independent of the capacity C and the units of time T, as they cancel out when dividing one profit by the other.

The profitability analysis of Economy class is based on the prediction results presented earlier. We use the estimated fractions of slack resources to be offered in Economy class and the resulting service availability for many prediction confidence levels for the ETS method, as presented in Figure 8. We also explore different price options for resources offered in Economy class, identifying the cases where adopting this new service class could be more profitable than offering all slack resources in the Opportunistic class.

In order to define the resource prices for the Opportunistic class, we used historic price values for different Amazon EC2 Spot Instances, measured from December 2013 to March 2014, to calculate a *relative resource price*: the average spot price divided by the (constant) on-demand price. We gathered the relative spot price for 4 general purpose Amazon EC2 instance types (m1.small, m1.medium, m1.large, m1.xlarge) in the us-west-1 region, and found that the average relative prices for each instance type ranged from 15.4% to 59.9% of the on-demand prices. The observed average relative spot price for the four instance types was 31.8%, which we used as the relative price for the resources offered in the Opportunistic class in our evaluation (i.e., $p_{op} = 31.8\%$).

As we want to identify reasonable prices for resources with SLOs to offer in Economy class, we explore different relative resource prices for this class, varying p_e from 30% to 100% of the On-demand class price. Lower and higher values were not explored as we believe that the resources offered in Economy class should be at least as valuable as the average Opportunistic price ($p_e \geq p_{op}$), as they have associated SLOs; but not more expensive than the resources offered in the On-demand class ($p_e \leq 1$), as reclaimable resources will probably be offered with lower SLOs.

Figure 9 shows the profit ratio and average availability when using different combinations of confidence levels and prices for Economy class resources.

As we have already seen, the higher the confidence level (i.e., more conservative), the higher the availability tends to be, so the penalties paid by the provider tend to be lower. However, the increase in the confidence level results in less resources offered in Economy class, which decreases the gains for this class. Therefore, the provider needs to choose a confidence level that will result in a good balance for the availability and the amount of resources offered in Economy class, which could bring high revenues, good SLOs and low penalties for the reclaimable resources.

We also see that the profit ratio increases when the price for resources offered in Economy class increases. Not surprisingly, the profit from Economy class is lower than the profit for the Opportunistic class when its relative resource price is $p_e = 30\%$, which is slightly lower than the opportunistic price $p_{op} = 31.8\%$. But for almost all other relative resource prices, offering slack resources in Economy class is more profitable, even after paying penalties for SLO violations.

For the more conservative predictions (i.e., 90% confidence level), the average availability ranges from 98.9% to 100%. Let's assume that 70% of the On-demand class price is a reasonable relative price for the resources offered in Economy class with this availability level. For this scenario, the profit when using Economy class would be 22–60% higher than the profit when all resources are offered in the Opportunistic class. For the less conservative predictions (i.e., 50% confidence level), the average availability ranges from 88.7% to 100%. If we consider a relative resource price for Economy of 50% for this scenario, the profit increase for offering resources in Economy class would be 41–50%.

These results are based on the assumption that all reclaimable resources offered are actually consumed by users, regardless of their price and availability. In practice, high prices and/or poor SLOs would reduce demand, but we do not know by how much - we don't have any data on how much users are willing to pay for different SLOs. But it seems reasonable to assume that there will be at least some demand for Economy class if it is priced lower than the On-demand price. Any remaining unsold capacity can always be offered in the Opportunistic class, so there are few downsides for the service provider, as long as the availability is good enough to avoid too many penalties. As we noted above, our penalty calculations are extremely conservative, so this seems eminently achievable.

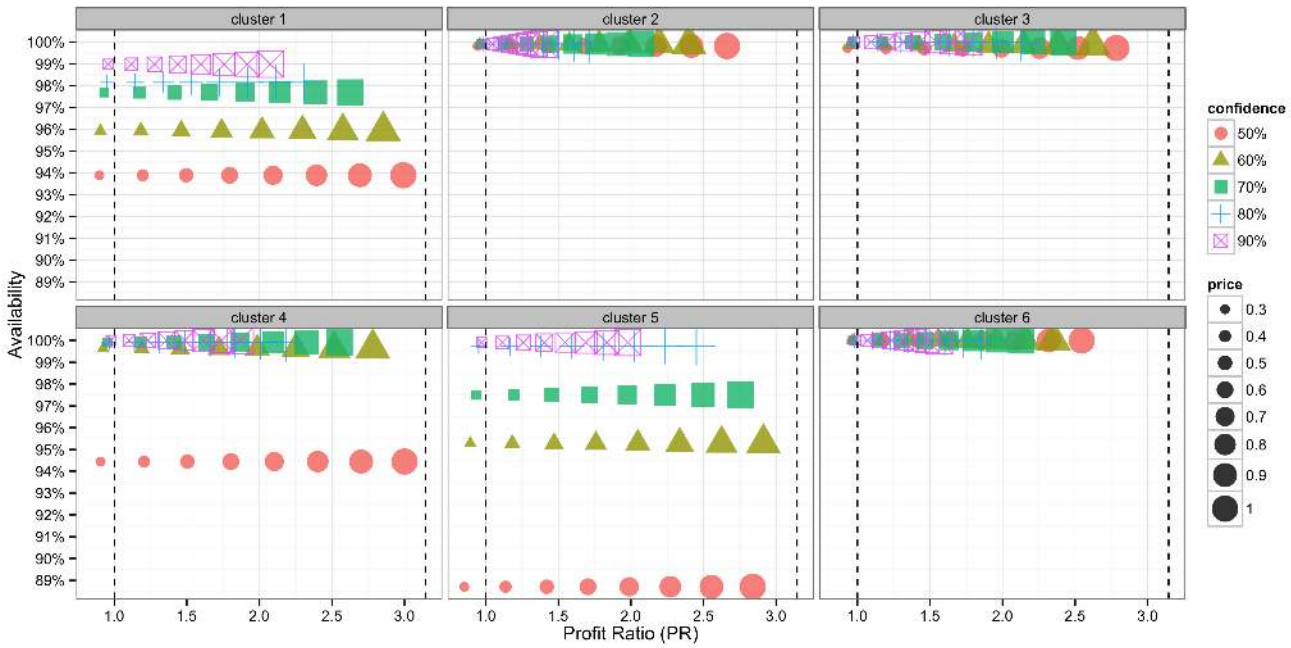

Figure 9. Profit ratio and availability for different scenarios of prediction confidence levels and resource prices for Economy class. Different shapes represent different confidence intervals; different sizes represent different prices. The leftmost dashed line presents the break-even point (profit ratio $\theta = 1$); points on the right side present scenarios on which it is profitable to offering resources in Economy class; points further to the right are more profitable. The rightmost vertical line shows the maximum possible profit ratio, but this requires the (unimplementable) oracle predictor and setting the price for Economy resources to be the same as for On-demand resources.

6. Related work

Much previous work has covered resource management in cloud computing, such as alternatives to increase cloud computing utilization [3, 20], analysis of cloud and Grid workloads [10, 12, 22, 24, 27] (they're different), prediction [11, 14, 16, 21, 23, 25] of cloud workloads and energy-efficient resource management [6, 8, 15].

Marshall et al. [20] proposed reusing unused cloud resources by offering opportunistic, preemptible leases for them with no SLOs. Their results show that doing so can increase cloud utilization while meeting availability requirements for applications, which corroborates the motivation of our work. Amazon EC2 Spot Instances [3] is a product that essentially does the same thing. Although these solutions increase cloud utilization, the offered resources have no SLO. In our work we add more value for reclaimable resources by offering long-term SLOs for Economy class, allowing the cloud provider to increase utilization with resources that are more profitable and providing a new QoS class that is easier for users to take advantage of because of its predictability.

Previous work has proposed prediction models for resource load over time on cloud environments using several techniques, in order to help dynamic provisioning decisions and avoid SLO violations [11, 16, 23, 25]. The main focus was on forecasting point values in relatively small future windows (a few hours or days), by detecting workload patterns such as diurnal cycles. Our work also uses time series forecasting techniques to predict the minimum amount of resources that would be unused, but does so over a much longer period (6 months); and combines this with confidence intervals. We go on to explore trade-offs between the conservativeness of the predictions against the estimates, allowing the service provider to choose a long-term availability SLO target and determine the confidence level accordingly.

Reiss et al. [24] highlighted the heterogeneity and dynamicity of cloud resources and workload, showing a high variation on machine configurations and task request shapes (e.g., CPU:memory ratios), as well as specific task requirements. This variation can have an impact when managing the resource classes presented in this paper, as we discuss only coarse grain unused capacity available for the new QoS class proposed, aggregated from all machines in each cluster. In future work we plan to do a finer grain per-machine analysis, to measure the amount of stranded resources that are useless because no task request shape fits. This could reduce the practical amount of reclaimable resources available, requiring strategies such as VM migration to improve utilization.

Other work has explored classification of tasks according to their duration and amount of resources requested [22];

modelling of task resource consumption over time [27]; and comparison of cloud and grid workloads regarding tasks and machine load [10]. The durations of the cloud traces analyzed were up to 1 month, from a single cluster. In our work, we analyzed long-term aspects of cloud workloads, such as the fraction of users' ceilings that are actually requested over time, and the amount of resources left unused in many dimensions. We analyzed a year of measurements from 6 Google production clusters, and evaluated our prediction process using traces of almost 4 years of data.

Energy-efficient resource management approaches were also proposed, showing that predictive and reactive strategies to shut servers down or to put them in sleep state can lead to reduced power consumption (and costs) in data centers, while meeting SLAs [6, 8, 15]. Although our prediction process could also be used to estimate the amount of servers that could be inactive, we believe that this approach is not always possible, as idle machines may also run storage servers and, in this case, cannot be turned off. Thus, efficiently reclaiming unused resources for such servers as we show in this paper would be more adequate. A hybrid approach that turns off part of the machines and resells slack resources from others could be considered in future work.

7. Conclusion

Cloud computing elasticity comes at a price. In order to offer strong obtainability and availability SLOs, cloud providers must over-provision resources, resulting in low utilization and driving up costs. A common way to fix this and increase utilization is to offer the unused resources opportunistically (i.e., with no SLOs). Although such resources can be useful for short-lived jobs that can tolerate unknown unavailability periods for lower prices, they are unattractive for many other use cases, especially services that run for long periods of time and need some indication on the expected uptime for them over future months.

We believe that offering multi-month availability SLOs for at least some of these slack resources is important, and showed how to do so, by defining a new service class called Economy class. We used predictions based on time series forecasts to populate Economy class with resources that have good long-term obtainability and availability SLOs. We used confidence intervals to quantify the risks of SLO violations, and provided a trade-off between risk and the quantities of resources that could be made available in this class.

We evaluated our approach using almost 4 years of data from 6 production clusters at Google, spanning tens of thousands of machines. We believe this is representative of a wide range of cloud workloads, given the long time period (unseen in previous cloud studies); the large scale of the clusters; the use of several different clusters; and their workload diversity – each cluster ran many thousands of applications, and had significantly different workload patterns.

Our results show that even with a very conservative approach, 6.7–17.3% of resources could be re-offered in Economy class with an availability of 98.9% or better. We also showed that a service provider could increase its profit from selling reclaimed resources in Economy class – in our examples, by up to 60%, even after paying penalties for SLO violations.

Finally, we believe that having this new option of service class will prove beneficial to cloud users, too: they can benefit from strong, long-term SLOs for availability to better manage their applications, while paying less than they would for resources in the Reserved or On-demand classes.

Acknowledgments

The authors would like to thank Abhishek Verma and the anonymous reviewers for providing valuable feedback on this paper. Francisco Brasileiro thanks the support from CNPq/Brazil.

References

[1] Amazon EC2 FAQs - how many instances can i run in Amazon EC2? "http://aws.amazon.com/ec2/faqs/#How_many_instances_can_I_run_in_Amazon_EC2", Apr. 2014.

[2] Amazon EC2 service level agreement. "http://aws.amazon.com/ec2/sla/", Apr. 2014.

[3] Amazon EC2 instance purchasing options. "https://aws.amazon.com/ec2/purchasing-options", Apr. 2014.

[4] G. E. P. Box and G. Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990. ISBN 0816211043.

[5] K. P. Burnham and D. R. Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer, 2 edition, 2002. ISBN 0387953647.

[6] J. S. Chase, D. C. Anderson, P. N. Thakar, A. M. Vahdat, and R. P. Doyle. Managing energy and server resources in hosting centers. In *ACM Symposium on Operating Systems Principles (SOSP)*, pages 103–116, 2001.

[7] C. Chatfield. 5. forecasting. In *The analysis of time series: an introduction*. CRC Press, Boca Raton, FL, USA, 6th edition, 2004.

[8] Y. Chen, A. Das, W. Qin, A. Sivasubramaniam, Q. Wang, and N. Gautam. Managing server energy and operational costs in hosting centers. In *ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, pages 303–314, 2005.

[9] R. Costa, F. Brasileiro, G. Lemos, and D. Sousa. Analyzing the impact of elasticity on the profit of cloud computing providers. *Future Generation Computer Systems (FGCS)*, 29 (7):1777–1785, Sept. 2013.

[10] S. Di, D. Kondo, and W. Cirne. Characterization and comparison of cloud versus Grid workloads. In *International Conference on Cluster Computing (IEEE CLUSTER)*, pages 230–238, Beijing, China, Sept. 2012. .

[11] S. Di, D. Kondo, and W. Cirne. Host load prediction in a Google compute cloud with a Bayesian model. In *Interna-*

tional Conference on High Performance Computing, Networking, Storage and Analysis (SC), pages 21:1–21:11, Salt Lake City, UT, USA, Nov. 2012. ISBN 978-1-4673-0804-5.

[12] S. Di, D. Kondo, and C. Franck. Characterizing cloud applications on a Google data center. In *42nd International Conference on Parallel Processing (ICPP)*, Lyon, France, Oct. 2013.

[13] Y. Fu, J. S. Chase, B. N. Chun, S. Schwab, and A. Vahdat. SHARP: an architecture for secure resource peering. In *ACM Symposium on Operating Systems Principles (SOSP)*, pages 133–148, 2003.

[14] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Capacity management and demand prediction for next generation data centers. In *IEEE International Conference on Web Services (ICWS)*, pages 43–50, July 2007.

[15] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper. Resource pool management: Reactive versus proactive or lets be friends. *Computer Networks*, 53(17):2905–2922, 2009. ISSN 1389-1286.

[16] Z. Gong, X. Gu, and J. Wilkes. PRESS: PRedictive Elastic ReSource Scaling for cloud systems. In *International Conference on Network and Service Management (CNSM)*, pages 9–16, 2010.

[17] R. J. Hyndman and G. Athanasopoulos. 7. exponential smoothing. In *Forecast: principles and practice*. OTexts, 2013.

[18] R. J. Hyndman and Y. Khandakar. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 27(3):1–22, July 2008. ISSN 1548-7660.

[19] R. J. Hyndman, A. B. Koehler, J. K. Ord, and R. D. Snyder. Prediction intervals for exponential smoothing using two new classes of state space models. *Journal of Forecasting*, 24(1): 17–37, 2005.

[20] P. Marshall, K. Keahey, and T. Freeman. Improving utilization of infrastructure clouds. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 205–214, 2011.

[21] X. Meng, C. Isci, J. Kephart, L. Zhang, E. Bouillet, and D. Pendarakis. Efficient resource provisioning in compute clouds via VM multiplexing. In *IEEE/ACM International Conference on Autonomic Computing and Communications (ICAC)*, pages 11–20, Washington, DC, USA, 2010. ACM.

[22] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das. Towards characterizing cloud backend workloads: insights from Google compute clusters. *SIGMETRICS Performance Evaluation Review*, 37(4):34–41, Mar. 2010.

[23] F. J. A. Morais, F. V. Brasileiro, R. V. Lopes, R. A. Santos, W. Satterfield, and L. Rosa. Autoflex: Service agnostic auto-scaling framework for IaaS deployment models. In *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 42–49, 2013.

[24] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *ACM Symposium on Cloud Computing (SoCC)*, pages 7:1–7:13, San Jose, CA, USA, 2012.

[25] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes. CloudScale: elastic resource scaling for multi-tenant cloud systems. In J. S. Chase and A. E. Abbadi, editors, *ACM Symposium on Cloud Computing (SoCC)*, page 5, 2011.

[26] C. A. Waldspurger. Memory resource management in VMware ESX Server. In *OS Design and Implementation (OSDI02)*, pages 181–194, Dec. 2002.

[27] Q. Zhang, J. Hellerstein, and R. Boutaba. Characterizing task usage shapes in Google compute clusters. In *Workshop on Large Scale Distributed Systems and Middleware (LADIS)*, Seattle, WA, USA, Sept. 2011.