5-15-2012

# AUTONOMIC MANAGEMENT OF SOFTWARE AS A SERVICE SYSTEMS WITH MULTIPLE QUALITY OF SERVICE CLASSES

Tobias Brandt
*University of Freiburg*

Ye Tian
*University of Freiburg*

Markus Hedwig
*University of Freiburg*

Dirk Neumann
*University of Freiburg*

# AUTONOMIC MANAGEMENT OF SOFTWARE AS A SERVICE SYSTEMS WITH MULTIPLE QUALITY OF SERVICE CLASSES

Brandt, Tobias, University of Freiburg, Chair of Information Systems Research, Platz der Alten Synagoge, 79085 Freiburg, Germany, tobias.brandt@is.uni-freiburg.de

Tian, Ye, University of Freiburg, Chair of Information Systems Research, Platz der Alten Synagoge, 79085 Freiburg, Germany, ye.tian@is.uni-freiburg.de

Hedwig, Markus, University of Freiburg, Chair of Information Systems Research, Platz der Alten Synagoge, 79085 Freiburg, Germany, markus.hedwig@is.uni-freiburg.de

Neumann, Dirk, University of Freiburg, Chair of Information Systems Research, Platz der Alten Synagoge, 79085 Freiburg, Germany, dirk.neumann@is.uni-freiburg.de

## Abstract

*In recent years the emergence of Software as a Service (SaaS) provision and cloud computing in general had a tremendous impact on corporate information technology. While the implementation and successful operation of powerful information systems continues to be a cornerstone of success in modern enterprises, the ability to acquire IT infrastructure, software, or platforms on a pay-as-you-go basis has opened a new avenue for optimizing operational costs and processes. In this context we target elastic SaaS systems with on-demand cloud resource provisioning and implement an autonomic management artifact. Our framework forecasts future user behavior based on historic data, analyzes the impact of different workload levels on system performance based on a non-linear performance model, analyzes the economic impact of different provisioning strategies, derives an optimal operation strategy, and automatically assigns requests from users belonging to different Quality of Service (QoS) classes to the appropriate server instances. More generally, our artifact optimizes IT system operation based on a holistic evaluation of key aspects of service operation (e.g., system usage patterns, system performance, Service Level Agreements). The evaluation of our prototype, based on a real production system workload trace, indicates a cost-of-operation reduction by up to 60 percent without compromising QoS requirements.*

*Keywords: Service Level Agreements, QoS Classes, Workload Forecast, Autonomic Management.*

# 1 Introduction

The emergence of Software as a Service (SaaS) provision and cloud computing in general since the beginning of this century has had a tremendous impact on corporate information technology. While the successful operation of powerful information systems continues to be a cornerstone of success in modern enterprises, their sustainable and efficient operation has become an exceptionally hard problem. With the emergence of the Service World in generally and the ongoing establishment and acceptance of cloud computing and SaaS as the new standards in corporate computing, today enterprises have gained the ability to acquire IT infrastructure, software, or platforms on a pay-as-you-go basis. This new way of corporate computing has turned the former asset IT into an expense (Carr, 2004) and hence has opened a new avenue for optimizing operational costs and processes by eliminating the need for corporate-owned server infrastructure, the maintenance of that infrastructure, as well as software licenses.

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 propose our new SaaS management artifact which enables highly efficient and QoS aware operation and 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 Service Level Agreements (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 new management approach derives profit optimal operation strategies. Furthermore, our new model enables the automatic assignment of user to QoS classes and hence enables to ensure a high QoS for a preferred user group while at the same time offering decent QoS to the remaining users. 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. In contrast to most other work in this domain, we additionally allow multiple user classes with different QoS requirements. Usually, 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 SaaS Systems such as CRMs or ecommerce portals. 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). Additionally, our model incorporates a novel concept to manage multiple user classes with different SLAs. It enables the SaaS provider to allocate more resources to a certain user group without compromising the QoS of the remaining users.

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 QoS classes within a single service, the problem becomes increasingly complex. For instance, preferred users should be managed more conservatively. 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 management artifact 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 validate 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 consists of a cloud infrastructure, a benchmark application, extensive monitoring and control software, and the new management artifact 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 60 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 management artifact that enables the efficient and QoS aware operation of SaaS. Our novel model allows highly efficient operation modes and leverages the full potential of elastic applications in cloud computing environments.
- Our management artifact allows the operation of multiple QoS classes in a single system. More concretely, our model automatically assigns users to a certain class and allocates the workload on the system accordingly. By this means, the system is able to offer preferred users a better user experience while at the same time offering a decent service quality to the remaining user.
- The evaluation of our model is based on production driven workload traces and real system performance data recorded in our test environment. Based on this data, we are able to analyze the impact of our model and can estimate its potential impact on the cost of operation in production.

The remainder of this paper is structured as follows. The next section discusses related work in the field of elastic application management and system performance modeling. Section three introduced the formal background of our management artifact and introduces the all required input models. Section four presents a case study of our management artifact. Afterwards the paper concludes with a summary and an outlook on our future work.

## 2 Related Work

The presented autonomic management artifact combines various different research threads into one interdisciplinary model. In the related work section, we will discuss the different aspects of performance modeling, workload forecasting, and online 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. The modular structure of our artifact allows the application of various performance analysis tools. However, the main benefit of our empirical model is its applicability in environments with limited monitoring functionality such as clouds. 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 by 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 SLAs 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 SLAs. The authors in Lim et al. (2010) developed an autonomic control model to scale elastic storage systems based on the utilization of the system.

Service Level Agreements and their different aspects have been addressed computer science researchers. Buyya et al. (2009) provide a good overview of SLAs in the field of clouds. Yeo and 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. Aib and Boutaba (2007) present 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. Hasselmeyer et al. (2006) 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, Koller and Schubert (2007) present architecture for autonomous QoS management based on SLA specifications. The paper by 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 SLA management and application have been solved individually. However, we haven't found any work combining all aspects into a single integrated model for information systems. In this sense, our work extends the current research in the field of SLA management, towards integrated and dynamic runtime management concepts for online transaction processing systems.

In this paper we merge the findings on performance modeling and on SLA management towards an integrated scaling mechanism for cloud resources. This mechanism derives a cost-optimal system configuration that satisfies QoS requirements under a given performance model.

# 3 Model Design

The premise of this model is the fact that SaaS providers may face heterogeneous customers with respect to their SLA requirements. Take for example two companies that contract a third party service provider to operate the same software service. However, both may have a different valuation of reliability or speed of this service. Consequently, they would negotiate different SLO parameters in their respective SLAs.

The SaaS provider now faces several options. First, s/he can simply ignore the difference between the SLAs and just treat all the customers equally according to the strictest QoS requirement in any SLA. This does, however, imply a decline of the revenue margin or even a net loss, as customers with the less demanding SLA ("low priority customers") are not willing to pay for the higher operation cost. The second option is only feasible if the server technology allows prioritizing certain requests, enabling the service provider to privilege high priority customer. It is unrealistic to assume that such a solution can always be implemented, since the system performance model would need to be able to account for any possible combination of high and low priority requests.

Finally, we propose a third option that requires the SaaS provider to maintain a separate set of servers for each user class. However, any spare capacities for high priority requests can be used to respond to low priority requests (according to the high priority SLO), such that the workload on the servers for the low priority user class decreases. The SaaS provider anticipates this and orders the server configuration for that time slot accordingly.

In the following subsections we describe the elements of this model and our autonomic management artifact, displayed in Figure 1.

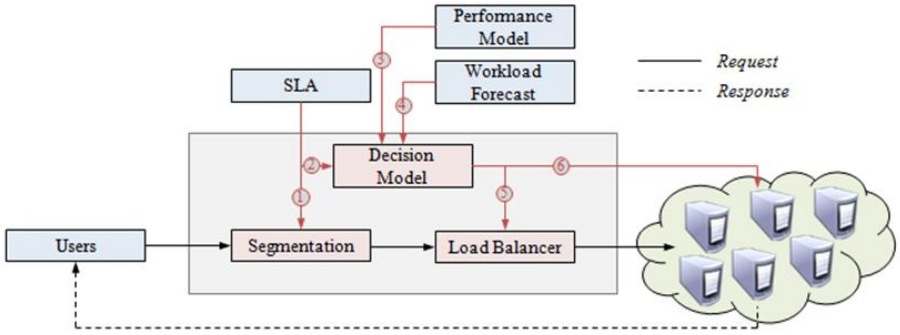


*Figure 1. Autonomic Management Artifact for SaaS System Operation*

## 3.1 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 this paper we investigate a scenario with two different user classes and, consequently, two different SLAs. However, the approach we choose theoretically allows any number of different SLAs to be included. The SLAs provide on the one hand the objective values according to which the decision model finds the optimal server configurations over the SLA lifetime (Fig. 1: 1), and on other hand define the user classes for the segmentation process (Fig. 1: 2).

## 3.2 System Performance Model

A thorough understanding of the system characteristics is necessary to provide the optimal hardware configuration for any given workload. Usually elastic applications do not scale linearly with the amount of hardware resources. In our recent work (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 and workload level. This influences the optimal configuration strategy in the decision model (Fig. 1: 3). Figure 2 presents a dataset of our empirical performance model based on data from our test system. The figure plots the degree of SLO compliance of different configurations for different workload levels. Evidently, the system scales with the number of resources in the system. Depending on the QoS requirements, different systems qualify for different workload levels. For instance, if the system faces 400 concurrent users and the operator targets a degree of SLO compliance of 99% the system must be operated with six nodes. In this work we advanced our performance model to determine the optimal infrastructure size for one QoS class and derive the residual capacity of the system. By allocating workload from a less strict QoS class, we can achieve a higher system utilization while at the same time reducing the overall cost of operation.

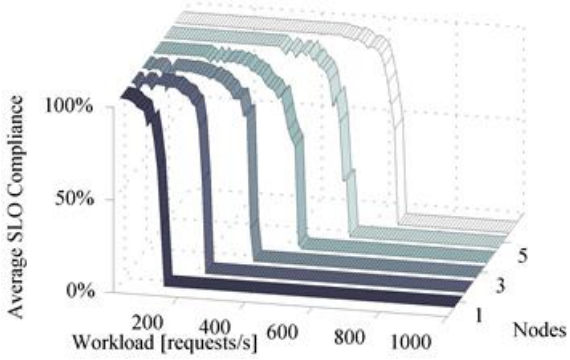


*Figure 2. System Performance Model*

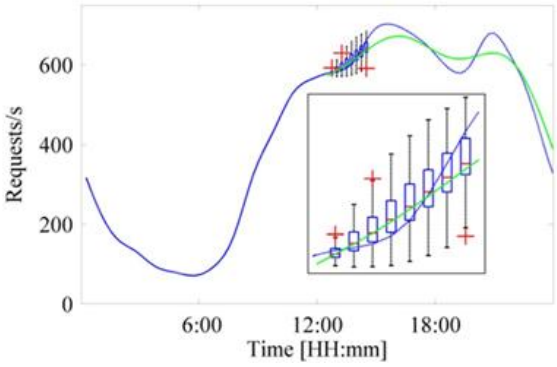


*Figure 3. Workload prediction versus actual workload*

## 3.3 Workload Forecast Model

While modern elastic applications allow resource adaptive operation without compromising system stability, significant reconfiguration lead times demand to reconfigure the system in advance. In our recent research (Hedwig et al., 2010), we developed a Fourier Transformation based workload forecast model, which decomposes the workload process with the help of the Discrete Fourier Transformation into its single spectra components. In an end-user generated workload scenario this includes in particular predominantly 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 refer to our previous publications. The forecast model supplies the decision model with the necessary information on future workload (Fig. 1: 4).

Figure 3 depicts a sample prediction result of our forecast model, which comprises two forecast methodologies. The first is a near-future workload predictor, which predicts the near future workload with very high accuracy. This forecast mechanism is used to initiate configuration adaptation decisions as it does not only predict the expected workload level but also estimates the prediction accuracy. The second, long-term forecast mechanism approximates the behavior of the workload process for up to several days. Though this mechanism provides a good estimate of the behavior of the process it entails a large prediction error and is thus not applicable for the operation of the system. However the combination of both allows on the one hand the real-time management of the system and on the other hand to evaluate the long-term impact of different operation strategies.

## 3.4 The Integrated Scaling Mechanism

This mechanism is an artifact, which determines operation strategies according to SLA specifications that support multiple user classes, system performance, and the prediction of the workload during the SLA lifetime. It also allocates user requests according to the respective user classes and the operational strategy. The paramount goal of the decision model is to maximize profits of the SaaS provider by minimizing the cost of operation. Recall that the SaaS provider acquires server instances from an Infrastructure-as-a-Service (IaaS) provider on a pay-as-you-go basis, such that fewer required instances directly translate into a reduction of operational costs. Therefore, the decision mechanism essentially minimizes the number of hardware instances over the SLA lifetime conditional on all QoS requirements being fulfilled. The resulting operational strategy determines the configuration within each time period beforehand (Fig. 1: 6), such that the hardware necessary to comply with SLA goals is provided when the workload arrives. The decision model also supplies the integrated load balancer with information on which requests to forward to which server (Fig. 1: 5).

## 3.5 Derivation of the Operational Strategy

The optimization problem reduces to minimizing the total number of instances over all user classes and the entire SLA lifetime, since server capacity is purchased on a pay-as-you-go basis.

$$\min_{c_{i+\Delta} \dots c_T} \sum_{t=\, i+\Delta}^{T} \sum_{\psi \in \Psi} c_t^{\psi} \qquad \text{with } i \in \{1-\Delta \dots T-\Delta\} \qquad (1)$$

We define $c_t^{\psi}$ as the hardware configuration operated in period $t$ with $t \in \{1 \dots T\}$, i.e. the SLA lifetime, for type $\psi$. We construct a set of possible types $\Psi = \{L, H\}$ – a low priority type with a less demanding SLA and a high priority type with stricter SLOs. Furthermore, we assume that reconfiguring the hardware requires a predefined lead time $\Delta$, such that decisions need to be made in advance. Hence, the configuration decision for $t = 1$ needs to be made in$i = 1 - \Delta$. In this work we assume that the lead is one period. This problem is subject to several constraints.

$$\sum_{t=i+\Delta}^{T} \beta_t^{\psi}\left(c_t^{\psi}\right) w_t^{\psi} \geq \alpha^{\psi} W^{\psi} - \sum_{t=1}^{i+\Delta-1} \beta_t^{\psi}\left(c_t^{\psi}\right) w_t^{\psi} \qquad \forall\, \psi \in \Psi \qquad (c1)$$
$$= \gamma_{i+\Delta}^{\psi}$$

$$w_t^{\psi} = \begin{cases} \widehat{w_t^{\psi}}, & t \leq i \\ E\left(w_t^{\psi}\right), & t > i \end{cases} \qquad (c2)$$

$$\beta_t^{\psi}\left(c_t^{\psi}\right) = \Lambda^{\Psi}\left(\mathrm{c}_{\mathrm{t}}^{\psi}, \omega_{\mathrm{x}}\right) = P_{c_t, x}^{\psi} \qquad \text{with } w_t^{\psi} \in \omega_x \qquad (c3)$$

$$P^{\psi} = \begin{pmatrix} \Lambda^{\Psi}(c=1,\omega_1) & \cdots & \Lambda^{\Psi}(c=1,\omega_m) \\ \vdots & \ddots & \vdots \\ \Lambda^{\Psi}(c=c_{max},\omega_1) & \cdots & \Lambda^{\Psi}(c=c_{max},\omega_m) \end{pmatrix} \qquad (c4)$$

The constraints represent the SLOs that need to be met during the SLA lifetime. For each type we set a maximum response time for incoming requests $\theta^{\psi}$ with $\theta^H \leq \theta^L$ and a compliance goal $\alpha^{\psi}$ with $\alpha^H \geq \alpha^L$. Thus, if $\theta^H = 300\,ms$ and $\alpha^H = 95\%$, then 95% of all incoming high priority requests over the SLA lifetime need to have a response time of 300ms or less for the SLA to be fulfilled (For reasons of readability we will drop the type index where possible). We further define $\beta_t$ as the compliance level in period $t$. The SLA compliance goal must then be less than or equal to the sum of the weighted compliances in each period of the SLA lifetime. Contrary to recent work where each period was weighted equally, they are now weighted by the workload of that period $w_t$ as a fraction of the total workload of the entire SLA lifespan, $W$. The compliance for all future periods starting with $i + \Delta$ necessary to fulfill $\alpha$ is then given by $\gamma_{i+\Delta}$ (c1).

The workload for $t$ is given either by the actual workload $\widehat{w_t}$, if known, or else by the workload predicted by the forecasting model, $E(w_t)$ (c2). The compliance level in period $t$, $\beta_t$, is derived from the system performance matrix $P$ (c3). $P$ yields for every combination of server configuration $c$ and workload interval $\omega$ the associated compliance $\Lambda$ according to the exogenously provided system performance model (c4, and Fig. 1: 3).

These components yield $\gamma_{i+\Delta}$, the minimum compliance for the remaining periods starting with $i+\Delta$. The operation strategy is derived by an iterative algorithm that first sets all strategic variables (i.e. $c_{i+\Delta} \dots c_T$) to their maximum value and then successively scales the configurations down by 1 such that the reduction has a minimal impact on the expected future compliance $\sum_{t=i+\Delta}^{T} \beta_t^{\psi}\left(c_t^{\psi}\right) w_t^{\psi}$. This process is repeated until no configuration can be reduced without violating (c1).

### 3.6 Modifications

Assuming this predictive strategy, the simplest implementation would maintain separate sets of servers for each SLA class. In this case, the optimization problem in equation (1) would be equal to the sum of the minimization problems of each user class. However, this approach does not exploit possible capacity reserves on H-type servers that can be used for L-type requests. Consider the case where the decision mechanism derives some configuration for the H-type $c_{i+\Delta}^{H*}, \dots, c_T^{H*}$ (2).

$$\sum_{t=i+\Delta}^{T} \beta_t^H(c_t^{H*}) w_t^H \geq \gamma_{i+\Delta}^H = \alpha^H W^H - \sum_{t=1}^{i+\Delta-1} \beta_t^H(c_t^H) w_t^H \tag{2}$$

This may result in a compliance level higher than necessary, but scaling down any of the configurations would cause an expected SLA violation by dropping below $\gamma_{i+\Delta}^H$. However, it may be possible to increase the workload in $i+\Delta$ by a certain amount $\delta$ and still comply with the compliance requirement (3). $\gamma_{i+\Delta}^{H\prime}$ reflects the change that this additional workload has on this compliance requirement.

$$\Lambda^{\mathrm{H}}\left(c_{\mathrm{i}+\Delta}^{H*}, w_{i+\Delta}^H + \delta\right)\left(w_{i+\Delta}^H + \delta\right) + \sum_{t=i+\Delta+1}^{T} \beta_t^H(c_t^{H*}) w_t^H \geq \gamma_{i+\Delta}^{H\prime} \tag{3}$$

$$\gamma_{i+\Delta}^{H\prime} = \alpha^H(W^H + \delta) - \sum_{t=1}^{i+\Delta-1} \beta_t^H(c_t^H) w_t^H \tag{4}$$

$$\Lambda^{\mathrm{L}}\left(c_{\mathrm{i}+\Delta}^{L}, w_{i+\Delta}^L - \delta\right)\left(w_{i+\Delta}^L - \delta\right) + \sum_{t=i+\Delta+1}^{T} \beta_t^L(c_t^L) w_t^L \geq \gamma_{i+\Delta}^{L\prime} \tag{5}$$

$$\gamma_{i+\Delta}^{L\prime} = \alpha^L(W^L - \delta) - \sum_{t=1}^{i+\Delta-1} \beta_t^L(c_t^L) w_t^L \tag{6}$$

$\delta$ could then be moved from low to high priority, thereby possibly reducing the necessary hardware requirements for the low type (5), the change in the compliance goal again reflected in $\gamma_{i+\Delta}^{L\prime}$ (6). If perfect information about the workload in all periods would be available beforehand, this operation strategy would guarantee a total hardware demand equal to or less than the result provided by the strategy without this modification. However, due to the uncertainty of future workloads this does not necessary need to be the case.

## 4 Case Study

To further investigate the effects of the autonomic scaling mechanism illustrated in the previous section and evaluate the potential cost and resource savings for SaaS providers we implemented the management artifact in a test environment based on production driven workload traces and real system performance data. In the evaluation we systematically analyzed the effect of introducing the predictive

operation strategy compared to a benchmark static operation strategy. Additionally, we empirically examined the effect of shifting workload from low to high priority by utilizing free capacities.

The scenario is based on the workload process of Wikipedia Germany over 40 days. The workload trace was scaled appropriately to be applicable to our system performance matrix. This matrix models the expected compliance level for two different values of the response time $\theta$ (representing the different SLAs) for any combination of 1 to 6 servers and 100 different workload levels. The target compliance for both user classes was set to 95%.

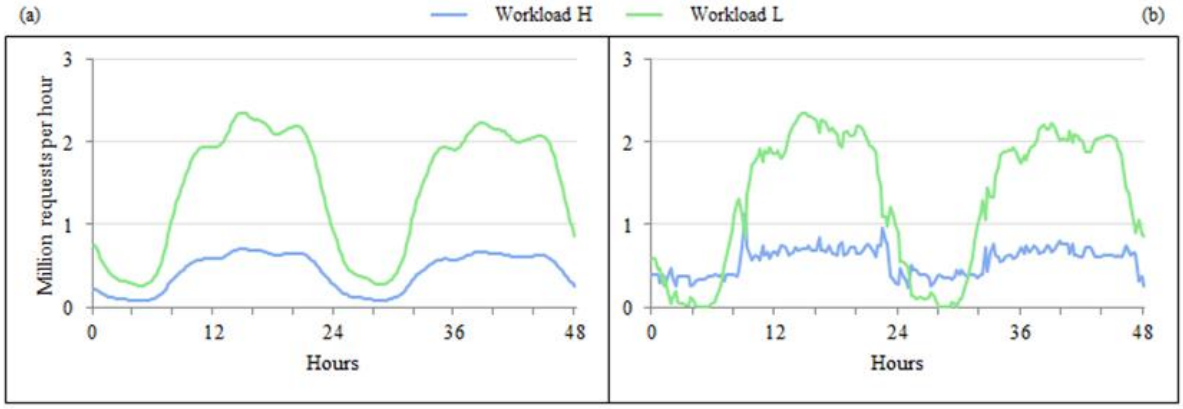


*Figure 4. Sample Workload Traces*

The workload process of the high priority type was derived as 30% of the low type process plus a normal distributed noise component. Fig. 4(a) illustrates the traces for two days taken from the sample.

The effect of shifting some workload intentionally from low to high priority to fully utilize high priority servers can be observed in Fig. 4(b) by comparing these traces to those in the previous illustration. It becomes obvious that the algorithm frequently shifts workload from low to high, resulting in much more irregular curves. It should be mentioned, however, that these irregularities do not affect the forecasting algorithm, since it considers the true type of every request.

## 4.1 Strategies

The benchmark case (BC) for the evaluation of our case study is a static operation strategy. This strategy attempts to satisfy the static compliance goal $\alpha$ in every period of the SLA lifetime. The peak values of the workload traces in our 40 day require 6 server instances for the low priority user class, and 5 for the high type. This results in 264 instance hours during the entire SLA lifetime of 24 hours.

The predictive strategy (PS) uses 14 days of historic workload data to predict the workload until the end of the SLA lifetime based on the previously described workload forecast model. Server capacities are scaled accordingly to achieve the compliance goal $\alpha$ over the entire SLA lifetime, thereby allowing for violations of this goal in single periods. However, operation strategies for each type are independent and separately determined.

The modified predictive strategy (MPS) extends PS by introducing the load shifting mechanism described in section 3.6. This allows for low priority requests to be processed on high priority servers if this shift is not predicted to violate the compliance goal. Especially during off-peak times it may occur that both, low and high priority requests, can be processed on a single high priority server, saving the additional instance that would be necessary if both user classes would be run separately. Operation strategies are interdependent and sequentially determined – first, the high priority strategy to estimate free capacities, after that the low priority strategy that incorporates these free capacities.

## 4.2 Evaluation

The results of our case study are summarized in Table 1. The benchmark has been chosen in such a way that both SLAs are always fulfilled; therefore, we have not calculated exact values for rows (1) to (4). The first important result is that SLA compliance is not violated by using the predictive strategies as the compliance level over the entire SLA lifetime never drops below 95%. This, however, is opposed by substantial reductions in the demand for instance hours, which is below 40% of the benchmark case for both predictive strategies. The effect of the modifications in the MPS becomes apparent, as well, as the required instance hours for the high priority type increase slightly. This effect is then more than offset by the demand reduction for the low type.

| | BC | PS | MPS |
|---|---|---|---|
| **(1)** | ≥ 95% | 95.71% | 95.04% |
| **(2)** | ≥ 95% | 95.45% | 95.77% |
| **(3)** | ≥ 95% | 95.10% | 95.00% |
| **(4)** | ≥ 95% | 95.03% | 95.01% |
| **(5)** | 100% | 29.48% | 32.28% |
| **(6)** | 100% | 46.02% | 41.84% |
| **(7)** | **100%** | **38.50%** | **37.50%** |
| | **PS** | | **MPS** |
| **(8)** | **100%** | | **97.40%** |
| **(9)** | 100% | | 101.38% |
| **(10)** | 100% | | 92.04% |

(1) Average compliance H
(2) Average compliance L
(3) Minimum compliance H
(4) Minimum compliance L
(5) Average instance hours over 24h (in percent of BC) H
(6) Average instance hours over 24h (in percent of BC) L
(7) Average instance hours over 24h (in percent of BC) Total
(8) Average total instance hours (in percent of PS)
(9) Maximum total instance hours (in percent of PS)
(10) Minimum total instance hours (in percent of PS)

*Table 1. Summary of case study results*

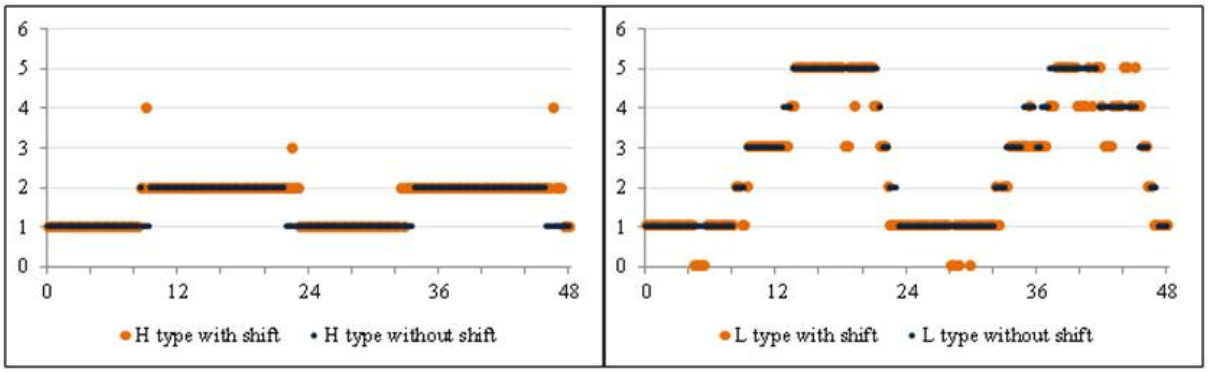


*Figure 5. Server configurations with and without workload shift*

That this does not always have to result in a net decrease of resource demand can be seen in rows (8) to (10). In some cases MPS produces results that are slightly above those provided by PS. This effect is caused by forecast errors when workload is shifted from low to high priority. If the workload prediction for the high type underestimates the actual workload, compliance levels for the high type may drop tremendously and need to be balanced during later periods by providing a higher compliance level. This is associated with higher resource requirements, resulting in a net increase of instance hours demanded. This is further illustrated in figure 5, which depicts the server configurations corresponding to the 48 hours of workload traces in figure 4. The left diagram shows that the configurations for the high priority user class are sometimes larger with shift than without. Contrary to that the low priority user class often requires smaller configurations if MPS is applied.

Table 1 shows that over all 40 SLAs MPS dominates PS by 2.60% on average and in some cases by up to 8%. Remember, that these reductions directly translate to an overall cost reduction for the SaaS provider. If we take the results yielded by our case study as representative and assume a cost of $2 per instance hour, the static operation strategy would incur annual costs of $\$2 \times 264 \times 365 = \$192{,}720$. The assumption of $2 per instance hour is estimated on the base of the Amazon EC2 resource prices (aws.amazon.com) for an extra large instance plus additional charges for traffic and storage. These annual costs would be reduced to $74,197 if PS was implemented and by an additional $1927 if the modified predictive strategy was implemented.

# 5 Conclusion

In this paper we presented a novel SaaS management artifact for the sustainable and efficient operation of elastic information systems in cloud environments. Based on a system performance model and a 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. The dynamic character of our 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. Consequently, the overall performance and reliability depends in particular on the integrated components. 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. 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 SLA specifications. Additionally, it is designed to simultaneously deal with different QoS classes. Thus, depending on the economic situation, it mitigates the risk of performance violations compared to rigorous cost-driven adaptive operation modes, while at the same time leveraging the potential of workload collocation. We showed that our model allows flexible system operation with a more than 60 percent lower cost of operation compared to static operation modes.

In our future work, we plan to extend our work 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. Furthermore, the revenue and penalty parameters have only been arbitrarily mentioned. In our future work, we intend 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.

# References

Aib, I., and Boutaba, R. (2007). On Leveraging Policy-Based Management for Maximizing Business Profit. IEEE Transactions on Network and Service Management, 4(3), 25-39.

Ardagna, D., Trubian, M., and Zhang, L.( 2007). SLA based resource allocation policies in autonomic environments. Journal of Parallel and Distributed Computing, 67(3), 259-270.
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., Chang, R. N., Luan, L. Z., Ward, C., Wolf, J. L., and Yu, P. S.( 2004). Utility computing SLA management based upon business objectives. IBM Systems Journal, 43(1), 159-178.
Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems, 25(6), 599-616.
Carr, N. G. (2004). Does IT Matter? Information Technology and the Corrosion of Competitive Advantage. Harvard Business School Press, Boston, MA.
Chandra, A., Gong, W., and Shenoy, P. (2003). Dynamic Resource Allocation for Shared Data Centers. Quality of Service — IWQoS 2003 (Berkeley, CA, June 02-04, 2003), 270, 381-398.
Cohen, I., Goldszmidt, M., Kelly, T., Symons, J., and Chase, J. S. (2004). Correlating instrumentation data to system states: a building block for automated diagnosis and control. Proc. 6th USENIX OSDI (San Francisco, CA, December 2004), 6.
Gmach, D., Rolia, J., Cherkasova, L., and Kemper, A. (2009). Resource pool management: Reactive versus proactive or let's be friends. Computer Networks, 53(17), 2905-2922.
Hasselmeyer, P., Qu, C., Schubert, L., Koller, B., and Wieder, P. (2006). Towards Autonomous Brokered SLA
Hedwig, M., Malkowski, S., and Neumann, D. (2010). Towards Autonomic Cost-Aware Allocation of Cloud Resources. Proc. ICIS 2010 (Saint Louis, MO, Dec. 12-15, 2010).
IBM (2011). Smarter Planet. Available at: http://www.ibm.com/smarterplanet/
Koller, B., and Schubert, L. (2007). Towards autonomous SLA management using a proxy-like approach. International Journal of Multiagent and Grid Systems, 3(3), 313-325.
Lim, H. C., Babu, S., and Chase, J. S. 2010. Automated control for elastic storage. Proceeding of the 7th international conference on Autonomic computing (Washington, DC, June 07-11, 2010), 1-10.
Malkowski, S., Hedwig, M., Li, J., Pu, C., and Neumann, D. (2011). Automated control for elastic n-tier workloads based on empirical modeling. Proc. ICAC '11 (Karlsruhe, Germany, Jun. 14-18, 2011), 131.
Padala, P., Hou, K.-Y., Shin, K. G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., and Merchant, A. (2009). Automated control of multiple virtualized resources. Proceedings of the 4th ACM European conference on Computer systems - EuroSys '09 (Nuremberg, Germany, March 2009), 13.
Pettey, C. and Tudor, B. (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.
Raimondi, F., Skene, J., and Emmerich, W. (2008). Efficient online monitoring of web-service SLAs. Proc. SIGSOFT '08/FSE-16 (Atlanta, GA, Nov. 9-14, 2008), 170.
Sahai, A., Durante, A., and 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
Short, J., Bohn, R., and Baru, C. (2011). How Much Information - Report on Enterprise Server Information. Available at: http://hmi.ucsd.edu/pdf/HMI_2010_EnterpriseReport_Jan_2011.pdf.
Urgaonkar, B., Shenoy, P., Chandra, A., Goyal, P., and Wood, T. (2008). Agile dynamic provisioning of multi-tier Internet applications. ACM Transactions on Autonomous and Adaptive Systems, 3(1), 1-39.
U.S. Environmental Protection Agency (2007). Report to Congress on Server and Data Center Energy Efficiency – Response to Public Law 109—431. Available at: http://www.energystar.gov/ia/partners/prod_development/downloads/EPA_Datacenter_Report_Congress_Final1.pdf.
Yeo, C. S., and Buyya, R. (2007). Integrated Risk Analysis for a Commercial Computing Service. IEEE IPDPS 2007 (Long Beach, CA, Mar 26-30, 2007), 1-10.