EFFICIENT AND FLEXIBLE MANAGEMENT OF ENTERPRISE INFORMATION SYSTEMS

Completed Research Paper

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

Simon Malkowski
Georgia Institute of Technology
266 Ferst Drive
30332-0765 Atlanta, USA
simon.malkowski@cc.gatech.edu

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

Abstract

The growing awareness of the substantial environmental footprint of Information System has increasingly focused corporate transformation efforts on the efficient usage of Information Technology. In this context, we provide a new concept to enterprise IS operation and introduce a novel adaptation framework that harmonizes operational requirements with efficiency goals. We concretely target elastic n-tier applications with dynamic on-demand resource provisioning for component servers and implement an adaptation engine prototype. Our framework forecasts future user behavior, analyzes the impact of workload on system performance, evaluates the economic impact of different provisioning strategies, and derives an optimal operation strategy. More generally, our adaptation engine optimizes IT system operation based on a holistic evaluation of the key factors of influence. In the evaluation, we systematically investigate practicability, optimization potential, as well as effectiveness. Additionally, we show that our framework allows flexible IS operation with up to a 40 percent lower cost of operation.

Keywords: Green IT/IS, Performance management, Service management, Cloud Computing, Enterprise software/systems

Introduction

"Being Efficient"—How well this maxim is implemented by a company is often the deciding factor that separates market leaders from market followers. In today's highly dynamic business environments, agile and effective adaptation in response to new market conditions, changed customer requirements, and disruptive technologies, is the key to sustainable success. While there is no "silver bullet" that guarantees successful companies to permanently solidify their competitive advantage, there are, however, various prominent best practices, which show that continuous transformation efforts result in long-lasting market leadership. For instance, Customer Relationship Management (CRM) systems allow synchronizing customer requirements with business processes in the manufacturing industry. Interestingly, these modern management philosophies have initially not considered the operation and management of Information Technology (IT) infrastructure as a first-class citizen in their optimization efforts (Carr, 2004). The main reasons for neglecting IT infrastructures were twofold; first, the concept of IT as an asset is relatively young; second, the complexity of IT systems and IT infrastructure often results in non-transparent costs, which is perceived as high risk. As a consequence, missing awareness and a lack of best practices further contributed to the perceived risk and have caused IT strategies to neglect efficiency for years (Carr, 2004).

Roughly a decade ago, the role of enterprise IT has changed dramatically, fueled by the observation that enterprise IT infrastructures expenditures are responsible for a significant share of total corporate expenditures. Moreover, empirical analysis showed that datacenters only achieve an average utilization of around twenty percent (ITP 2007), which implies that the majority of the existing computing potential remains unused (i.e., irrecoverable potential loss) (T. Velte et al. 2009). Consequently, many efficiency-centered IT trends emerged, which culminated in the dawn of Green IT as a result of the conjunction of IT efficiency and ecological responsibility (Schulz 2009).

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

It is a well-established fact that Green IT has already significantly reduced the environmental footprint of corporate IT (e.g., energy optimized datacenters); nevertheless, Green IT initiatives will continue to fall short of their full potential as long as they do not unify their efforts to optimize the operation of IT and the application of IT. Modern IT systems are typically offered as a service under certain Quality of Service (QoS) assertions. Because there is a design phase in which the offered service is specified, the design is affected by the service system that contains the IT infrastructure. As a result of this design, Service Level Agreements (SLAs) specifying the QoS assertions, which reflect the capabilities of the service system including the IT infrastructure, are created. In the operation phase, the service is offered to clients according to these SLAs.

Although SLAs have become the de-facto industry standard for defining the terms of operation in IT infrastructures, including revenue for maintaining the services, penalties for QoS violation, monitoring metrics, and corresponding thresholds, SLAs are predominantly treated statically. More concretely, SLAs are translated into a static set of requirements at the design time of the IT services. During runtime, IT infrastructures are then provisioned according to the maximum predicted peak load, which results in very low resource utilization on average. For small-sized systems, virtualization and consolidation are sufficient for adjusting the IT infrastructure size to the fluctuating demand; however, for mid-sized and large systems, consolidation becomes non-trivial due to the complexity of the underlying infrastructure, and dependencies may become prohibitively difficult to manage.

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

Our adaptation engine prototype is designed based on the goal of economical operation of large enterprise IT systems, which are typically distributed. The distributed architecture has three major advantages. First, commodity hardware can be utilized; second, standard modular software components can be deployed; third, scalability of different application parts becomes decoupled. In fact, the functionality of different application modules is separated into separate layers, whereby each layer is responsible for conducting a designated task (e.g. persistent data storage or application logic). Typically, each tier is deployed on separate hardware nodes or separate virtual hardware nodes. Depending on the complexity of the application and the hardware characteristics, the system is able to handle a specific workload level. By replicating servers in the different tiers and adding additional computing resources to them, the maximum sustainable workload level can be increased. The actual load management within a specific tier is typically handled by load balancers (e.g., round-robin request routing).

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

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

1. We provide a green perspective on enterprise IT systems and introduce a novel framework that enables highly efficient and context-aware operation modes.

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

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

The remainder of the paper is structured as follows. The next section presents the related work. The subsequent section discusses the design problem statement followed by the presentation of the Adaptation Engine Framework. Afterwards, the Adaptation Engine is evaluated in multiple scenarios. The paper concludes with a summary and an outlook.

Related Work

The presented adaptation engine framework combines various different research threads into one interdisciplinary IS management artifact. In the related work section, we 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 (I 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. 2008) 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. 2009) 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 for the near future. 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 by computer science researchers. (Buyya et al. 2009) provide a good overview of SLAs in the field of clouds. (Yeo & Buyya 2007) developed an integrated risk analysis scheme to analyze the effectiveness of resource management policies. Based on SLAs, they determine whether a system is capable of meeting the required objectives and whether the acceptance of a single job is economically feasible. (Aib & 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 (Koller & Schubert 2007) present architecture for autonomous QoS management based on SLA specifications. The paper by (Sahai et al. 2004) 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 of performance modeling and on SLA management towards a holistic adaptation engine. This mechanism enables highly efficient operation modes that satisfies QoS requirements under a given performance model.

Design Problem Statement

Technical advances, demanding customer expectations, as well as increasing system complexity, make sustainable operation of IT systems a challenging task. In our recent research we designed and developed various methods and concepts to address this challenge. In this section we present our own research work in this field. Please refer to the related work section to learn about alternative or complementary approaches. Classic system design paradigms prescribe to scale systems to the maximum expected workload. This inevitably leads to the situation where system operators tend to overprovision their IT (i.e., usage of more hardware resources than necessary) in order to guarantee continuous high quality of service (Hedwig, Malkowski, Bodenstein, et al. 2010). Nonetheless, many end-user information systems face highly volatile workloads, which, as a consequence, results in a low average utilization (Hedwig et al. 2009). An alternative to static operation modes is the use of Resource Management Systems. These management tools usually supervise the performance of a systems and allocate resources according to the user demand (Malkowski, Hedwig, et al. 2010). While these approaches work well for small-sized systems (i.e., applications which at most require one server), their application in large environments is fraught with problems (Malkowski et al. 2009). Similar to the concept in small systems, the idea in this domain is to allocate resources according to the demand. By dynamically adding and removing resources (e.g. servers) from the system, the average utilization of the system can be significantly increased while providing at the same time a high Quality of Service and hence contributing to highly efficient operation modes. However, the optimal adaptation of the system is non-trivial. More specifically, one key challenge is to overcome the *reconfiguration lead time*. Usually, the hardware configuration cannot be adapted instantaneously, rather there occurs a delay between the initiation of reconfiguration and its availability due to configuration of the resources, synchronization and reconfiguration of the load balancers (Hedwig, Malkowski & Neumann 2010). Nowadays, Resource Management Systems are rather reactive controllers (i.e., they only react to changes in their environment) and thus exhibit an inherent risk of performance violations due to the lead-time (Hedwig et al. 2011). *In this paper we attempt to design a practicable green operation model – the adaptation engine framework – that optimizes IT system operation based on a holistic evaluation of all key factors of influence.*

The holistic management of an information system requires a thorough understanding about all factors of influence. Even though the idea of resource-adaptive operation of large enterprise applications seems to be very tempting, its application is absolutely non-trivial. This complexity stems from the fact that the overall performance of systems depends on various, interdependent factors; changes in each factor can lead to critical performance limitations or – in the worst case – cause total system failure. In our recent work, we discussed these factors. In order to manage a system efficiently, the factors of influence need to be well understood. Based on our recent research work (Malkowski et al. 2011; Hedwig et al. 2012), we categorize the following factors of influence.

- **User behavior:** This factor represents the usage patterns and is often highly volatile over time (e.g. day time usage intensity is significantly higher than during night). Due to its stochastic nature, it can be deduced that the operation of the infrastructure needs to be dynamic, adapting to the actual user requests (Hedwig et al. 2009; Hedwig, Malkowski & Neumann 2010).
- **Hardware Characteristics:** The hardware and infrastructure characteristics (.e.g. amount of RAM, disk space, CPU speed, hardware design) naturally determine the physical computation power of the system (Malkowski et al. 2007).
- **The Software Characteristics** determines the resource requirement of a given information system to provide its functionality. In this sense, smart software contributes to a sensible resource usage and is hence a key factor of service for a given system (Malkowski, Jayasinghe, et al. 2010).
- **System State:** Next to hard- and software characteristics, the system state also significantly affects the overall system performance. For instance, transactions on a highly populated database probably demand more computational power than the same transaction on a small dataset (Malkowski et al. 2011).
- **Service Level Agreements:** As mentioned, Service Level Agreements specify all aspects of the business relation between the contract partners, such as the rights and duties of each party, contract duration as well as guarantees and warranties (Malkowski, Hedwig, et al. 2010).

Based on our recent work, these factors have the highest impact on the overall system performance. However, most of the factors cannot be adjusted at operation time in the short run. Hard- and software endowment can be assumed to be constant in the short-run, as hardware upgrades or software updates have a long lead time. Essentially, the system state is also constant in the short run, as it evolves gradually over time with sudden changes very unlikely to occur. As hard- and software characteristics and the system state are all fixed in the short run, we aggregate them into the factor 'system characteristics'.

Dependent on the operation environment – being stable in the short run – the system provider tailors services by the instantiation of SLAs at design time and offers it to customers.

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

Adaptation Engine Architecture

Having explained the factors of influence and their impact on system performance, we will develop a conceptual framework that integrates all those factors into a comprehensive model. The framework attempts to formalize all factors in a mathematical way, which is henceforth denoted as Adaptation Engine.

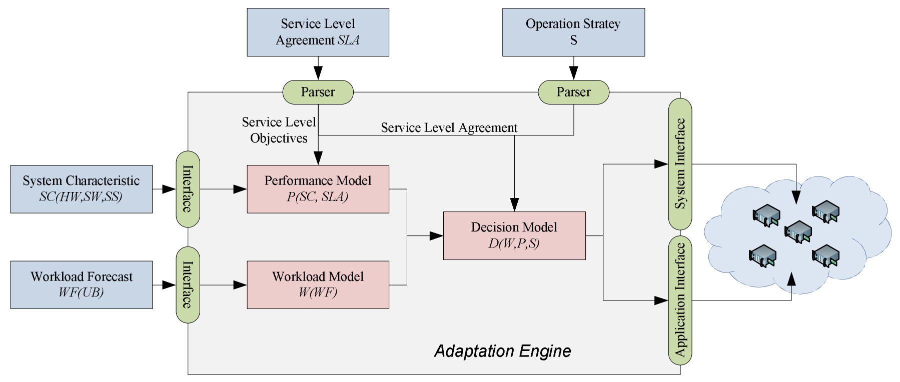

Figure 1: Adaptation Engine Architecture

Figure 1 shows our conceptual framework on a high level. The factors Service Level Agreements SLA, the system characteristics SC consisting of the hardware HW, software SW and the system state SS determine the performance of the system. The performance model specifies how many users can simultaneously access the service without violating the SLA. The factor user behavior UB affects the workload of the system. The Operation Strategy S defines the operation mode based on the aforementioned input factors. The workload model uses past workloads to forecast future workloads of the system. The Decision model confronts the system model with the forecast model and derives in conjunction with the operation strategy the right hardware configuration.

The Adaptation Engine is designed as a software component with clear defined interfaces to interact with its environment. In the figure, the Adaptation Engine is depicted as the shaded box containing the internal models consisting of the performance, workload and decision model. The interfaces of the Adaptation Engine include the inputs workload, system characteristics and SLAs. From a technical point of view, the adaptation engine encompasses the actuator interfaces, which initiate the reconfiguration of the infrastructure (i.e. add and remove servers) as well as the reconfiguration of the application (i.e. configure load balancers). The Adaptation Engine itself consists of three different internal models that convert the input information into configuration decisions. Finally, the framework imposes requirements on the input data, but not on the methodology these data are generated which facilitates the modular design.

Internal Adaptation Engine Model Design

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

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

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

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

The idea of the Adaptation Engine is that any conceivable forecasting component, which complies with the data requirements, can be plugged into the software component. For instance (Gmach et al. 2007) used Fourier Transformation for data smoothing and applied time series analysis for workload prediction. In (Powers et al. 2005) the authors used time series as well as regression analysis for workload prediction. In our recent research, we developed a workload forecast model, which predicts the near future behavior of a workload process based on Fourier Transformation and Time Series Analysis. This design assures that the Adaptation Engine does not rely on our models that will be presented in the evaluation section, but allows incremental improvement as long as the data requirements of inputs and outputs are satisfied.

$$P = \begin{array}{c} \\ c_1 \\ \vdots \\ c_k \end{array} \begin{array}{ccc} w_1 & \dots & w_n \end{array} \begin{pmatrix} x_{p_1w_1} & \cdots & x_{p_mw_m} \\ \vdots & \ddots & \vdots \\ x_{p_kw_1} & \cdots & x_{p_kw_m} \end{pmatrix} \quad (3)$$

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

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

The resource efficient and SLO aware operation of an elastic application requires a thorough understand of the system, including the infrastructure and software characteristics as well as the user behavior. In our recent research we developed an empirical performance model. Based on the monitored performance during operation, our empirical model is able to determine which configurations of the system are able to sustain a certain workload level according to the SLO definition. Similar to the workload model, the Adaptation Engine supports different types of input models. Alternative concepts include for instance, queuing models (Urgaonkar et al. 2008) or machine learning (I Cohen et al. 2004).

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

accordingly, configuration I has a zero probability of satisfying the SLA in period τ_1, configuration II has an 11% probability to meet the SLA in τ_1, whereas configuration III has a probability of 90%.

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

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

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

Operation Strategy Functions

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

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

Static system operation mode

The most basic operation strategy is a static system operation mode that keeps the hardware configuration unchanged over the whole SLA lifetime (7).

$$c \leftarrow S^{static}() \quad (7)$$

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

Basic reactive operation mode

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

$$S^{reactive}(D_0, c^*) = \begin{cases} D_{0c^*} < \alpha & \text{scale-up} \\ D_{0(c^*-1)} \geq \alpha & \text{scale-down} \\ else & \end{cases} \quad (8)$$

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

Reactive operation mode with performance reserve

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

$$S^{rr}(D_0, c^*) = S^{reactive} + 1 \quad (9)$$

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

Predictive operation mode

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

$$S^P(D, \alpha, c^*) = \begin{cases} D_{c^*\Delta} \leq \alpha & \text{scale-up} \\ D_{(c^*-1)\mathrm{d}} \geq \alpha, \forall_{d=0,\ldots,\Delta} & \text{scale-down} \\ else & \end{cases} \quad (10)$$

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

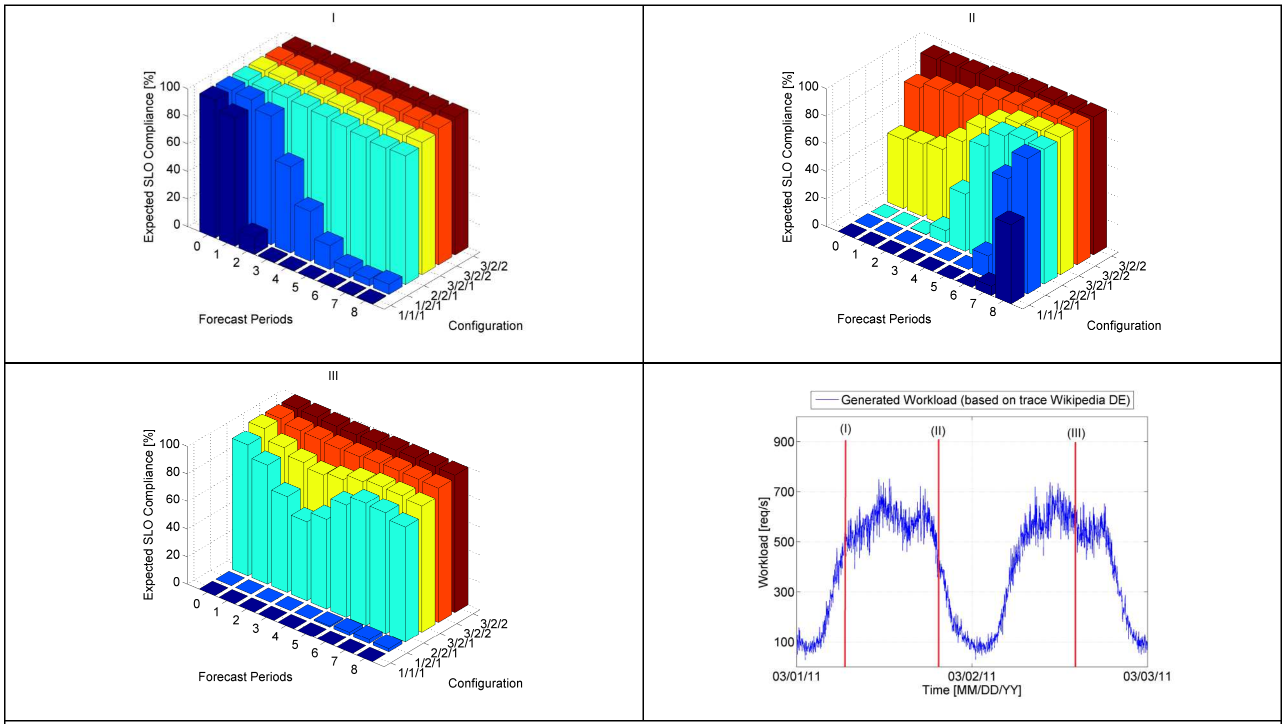

Figure 2: Decision matrix at different points in time

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

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

Evaluation

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

- **Practicability:** Can the adaptation engine framework be implemented for current IT systems?
- **Optimization of IT system operation:** Under what conditions can the adaptation engine optimize the IT system operation?
- **Effectiveness:** Can the Adaptation Engine significantly increase the operational efficiency?

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

The evaluation of our Adaptation Engine, based on real production system workload traces, was carried out in a self-developed test infrastructure (i.e., cloud testbed, elastic application, distributed monitors, and control framework).

Optimization of IT system operation

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

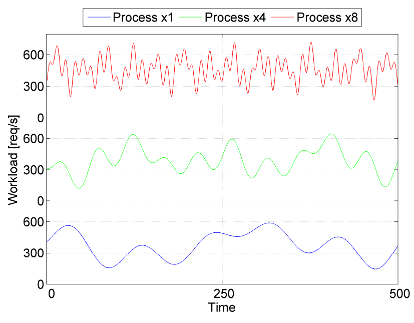

Figure 3: Synthetic workload process

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

In order to assess the performance of different operation modes exposed to different workload volatilities, we derived four additional processes on the base of the original process, whereas the periodicity of the process is increased by the factors 2, 4, 6 and 8. Figure 3 shows three sample input workload processes, where x1 is the original process, x4 represents the process with periodicity of 4, and x8 with a periodicity of 8 that we use in our evaluation. We assume that the SLO requires a system response time of below 200 ms for 95% of the requests within each second. In addition, we assume that the Service Level Agreement demands compliance with the SLO objective in 95% of the SLA lifetime, which is set to 500 periods $(t_1, \dots, t_{500})$.

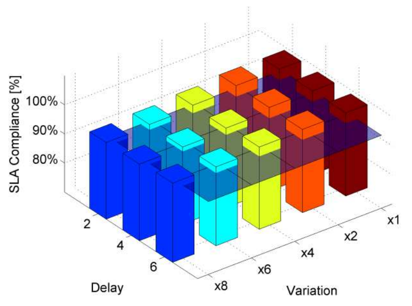

Figure 4: Predictive operation mode

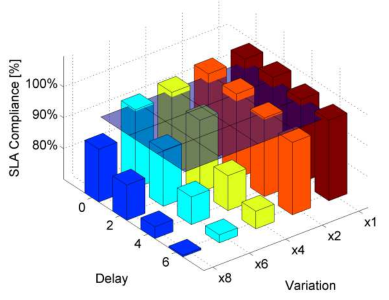

Figure 5: Reactive operation mode

Figure 4 and 5 present the resulting SLA compliances of 35 different system runs. Each system run uses a different lead time (delay) and process periodicity (variation). The left panel shows the performance of the predictive operation mode, whereas the right panel shows the performance of the reactive mode. X and Y-axes define the different settings for the reconfiguration delay and process periodicity. The shaded plane at the 95 % SLA compliance visualizes the target of the SLA – all scenarios below this plane violate the SLA; equal and above comply with the SLA. Evidently, the predictive operation mode succeeds in complying with the SLA in all scenarios, which we can summarize in result 1:

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

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

Table 1: Variation of processes with lag d

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

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

Table 2: System characteristics data

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

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

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

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

Effectiveness

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

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

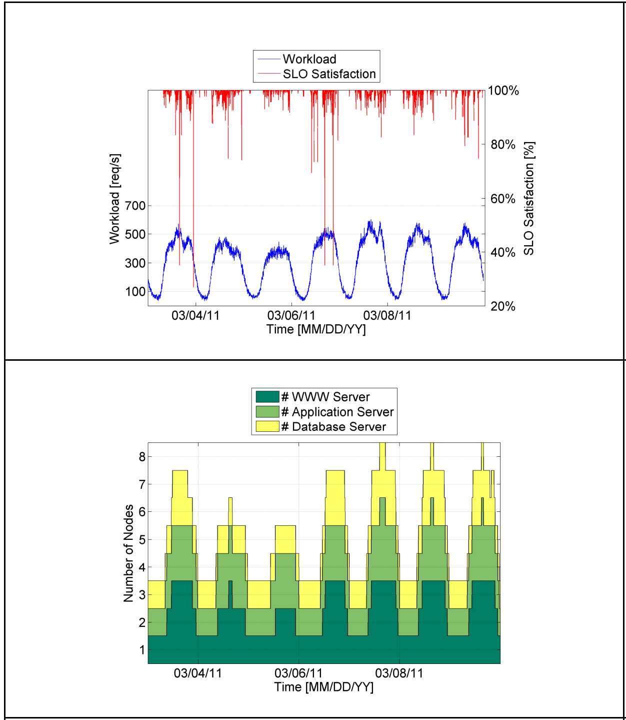

Figure 6: Predictive operation mode

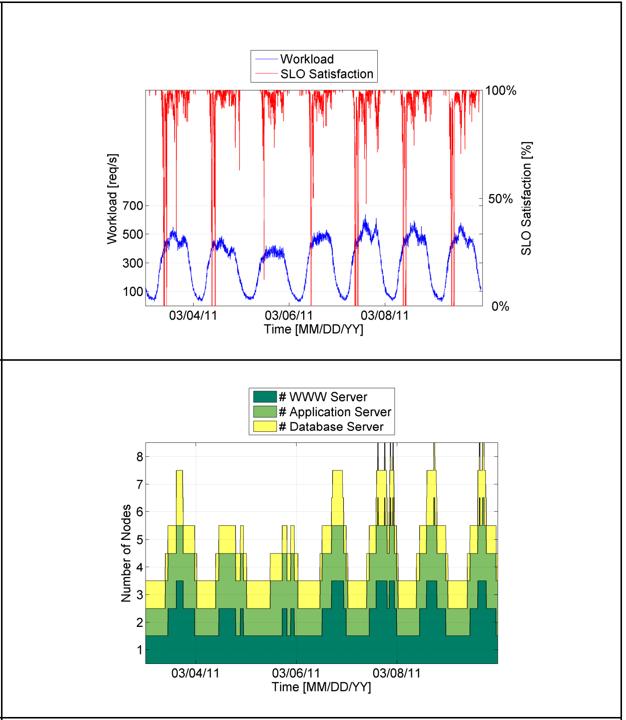

Figure 7: Reactive operation mode

Figures 6 and 7 exemplify the results of a single experiment series for the predictive (left) and basic reactive (right) operation mode. The upper graphs depict the workload process and the SLO compliance whereas the lower graphs show the hardware configuration. The reconfiguration lead time is set to one hour. While the predictive operation mode achieves an SLA compliance of 99.1%, the reactive strategy only achieves a compliance of 94.4%. Having defined the SLA of 95 % the basic reactive mode has violated the SLA. During off peak times, both operation modes achieve an SLO compliance of 100% (i.e. all incoming requests are processed within 200 ms), which results from the fact that the minimal configuration can process up to 290 concurrent requests. During peak times the predictive operation strategy continuously aims at selecting hardware configurations to achieve 95% compliance. As a consequence, during the first four days, the predictive strategy never switches to the strongest system configurations, as the smaller are already sufficient to comply with the SLO. Evidently, during the peak time several SLO compliance monitoring points are below the 95% threshold – refer to the red line in the upper graphs. Essentially, the SLO satisfaction drops considerably when the workload transforms from off peak to peak times. Nevertheless, the predictive operation mode selects configurations, which comply with the SLO on average during each period (15 minute intervals), and hence operates within the defined boundaries of the SLA. Compared to the basic reactive mode, the predictive mode employs less reconfiguration operation, which is a result of the look ahead and thus more sensitive scale-down operation of the predictive operation mode. We can summarize this in result 3:

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

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

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

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

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

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

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

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

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

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

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

Essentially, results 4-6 suggest that the reactive operation mode should be used whenever the lead time of reconfiguration actions is very short. If the application is write-intensive, the predictive operation mode is more appropriate. From the green perspective, the resource adaptive operation allows to shut-down the non-productive servers. Implementing such a policy would directly reduce the carbon footprint of the system by the same magnitude. From the economic perspective and assuming the total cost of operating a server is $1.00 per hour (cost of a small Amazon EC2 instance per hour), the Adaptation engine allows cost reductions of up to $4000 per year already for a small infrastructure with 8 servers.

Conclusion

In this article, we presented our adaptation engine framework for the efficient and flexible management of large enterprise information systems. Based on a system performance model and workload forecast model, our new management concept enables highly efficient operation modes and thus contributes to a more sensible utilization of information system resources. Our model extends the current state of the art by directly translating SLA specifications into management requirements. It provides a methodology to automatically manage complex systems according to their requirements and is, therefore, particularly useful for services offered in different QoS classes with different price models.

Conceptually, our adaptation engine framework is a model that includes SLAs in the operation of IT systems in a dynamic and flexible fashion. To accommodate the heterogeneity of the domain, the engine was designed modularly, enabling different configurations according to actual system characteristics. During operation, the adaptation engine systematically analyzes all key factors of influence, such as hardware characteristics and software design, to evaluate all these factors in a compact decision model. Based on the decision model and a strategy function, the adaptation engine manages the enterprise IS. In our evaluation, we discussed several different types of such operation strategy functions and determined that particular predictive operation modes have a high potential in scenarios with volatile workload and complex performance characteristics.

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

The Adaptation Engine is designed to be integrated into productive information systems. The presented configuration with the Fourier Transformation based workload forecast model, the empirical system performance model and our SLA model is only one example. The modular design of the engine allows IT managers to configure the Adaptation Engine according to the individual characteristics of their system. The key challenge is to enable the software and the environment to support adaptive operation modes. This includes interfaces for the dynamic reconfiguration of the operation environment as well as interfaces to reconfigure the software at runtime. Though these modifications may be cost intensive during design time, they will pay off in the long-term and lead to a more sustainable practice in information system operation.

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

References

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

Ardagna, D. et al., 2009. Run-time resource management in SOA virtualized environments. *Proceedings of the 1st international workshop on Quality of service-oriented software systems - QUASOSS '09*, p.39. Available at: http://portal.acm.org/citation.cfm?doid=1596473.1596484.

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

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

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

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

Cohen, I et al., 2004. Correlating instrumentation data to system states: A building block for automated diagnosis and control. *In OSDI*.

Gmach, D. et al., 2008. Adaptive quality of service management for enterprise services. *ACM Trans. Web*, 2(1), pp.1-46.

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

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

Hedwig, M., Malkowski, S. & Neumann, D., 2012. Efficient and Flexible Management of Enterprise Information Systems. In *33nd International Conference on Information Systems (ICIS 2012), Orlando, Florida, 16-19 December 2012*.

Hedwig, M., Malkowski, S. & Neumann, D., 2011. Integrated SLA Management for Green Information Systems. In *32nd International Conference on Information Systems*.

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

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

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

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

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

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

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

Malkowski, S., Hedwig, M. & Pu, C., 2009. *Experimental evaluation of N-tier systems: Observation and analysis of multi-bottlenecks*, IEEE. Available at: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5306791.

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

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

Malkowski, S. et al., 2007. Bottleneck Detection Using Statistical Intervention Analysis A. Clemm, L. Z. Granville, & R. Stadler, eds. , 4785, pp.122-134. Available at: http://www.springerlink.com/content/ul256642j4u40182/ [Accessed October 7, 2011].

Malkowski, S., Jayasinghe, D., et al., 2010. Empirical Analysis of Database Server Scalability Using an N-tier Benchmark With Read-intensive Workload. In *SAC '10: Proceedings of the 2010 ACM Symposium on Applied Computing*. New York, NY, USA: ACM.

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

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

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

Sahai, A. et al., 2004. Automated Generation of Resource Configurations through Policies. In *in Proceedings of the IEEE 5 th International Workshop on Policies for Distributed Systems and Networks*. pp. 7-9.

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

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

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

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

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