

# CloudXplor: A Tool for Configuration Planning in Clouds Based on Empirical Data

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

Markus Hedwig  
Chair of Information Systems  
Albert-Ludwigs-University  
Platz der Alten Synagoge  
79805 Freiburg, Germany  
markus.hedwig@is.uni-  
freiburg.de

Deepal Jayasinghe  
CERCS  
Georgia Institute of Technology  
266 Ferst Drive  
30332-0765 Atlanta, USA  
deepal@cc.gatech.edu

Calton Pu  
CERCS  
Georgia Institute of Technology  
266 Ferst Drive  
30332-0765 Atlanta, USA  
calton@cc.gatech.edu

Dirk Neumann  
Chair of Information Systems  
Albert-Ludwigs-University  
Platz der Alten Synagoge  
79805 Freiburg, Germany  
dirk.neumann@is.uni-  
freiburg.de

## ABSTRACT

Configuration planning for modern information systems is a highly challenging task due to the implications of various factors such as the cloud paradigm, multi-bottleneck workloads, and Green IT efforts. Nonetheless, there is currently little or no support to help decision makers find sustainable configurations that are systematically designed according to economic principles (e.g., profit maximization). This paper explicitly addresses this shortcoming and presents a novel approach to configuration planning in clouds based on empirical data. The main contribution of this paper is our unique approach to configuration planning based on an iterative and interactive data refinement process. More concretely, our methodology correlates economic goals with sound technical data to derive intuitive domain insights. We have implemented our methodology as the *CloudXplor Tool* to provide a proof of concept and exemplify a concrete use case. CloudXplor, which can be modularly embedded in generic resource management frameworks, illustrates the benefits of empirical configuration planning. In general, this paper is a working example on how to navigate large quantities of technical data to provide a solid foundation for economical decisions.

## Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems—*distributed applications*

## General Terms

Measurement, Performance, Experimentation.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SAC'10 March 22-26, 2010, Sierre, Switzerland.

Copyright 2010 ACM 978-1-60558-638-0/10/03 ...\$10.00.

## Keywords

Cloud, Configuration Planning, N-tier applications, RUBBoS.

## 1. INTRODUCTION

Although modern businesses are under constant market pressure to reduce the cost of their computing infrastructures, making sustainable configuration planning decisions is becoming an increasingly challenging task. Under the premise of lean and cost efficient information systems, new Information Technology (IT) trends and paradigms have led to a rapid growth of management complexity. In fact, recent research shows that there are several developments that are of particular relevance to the area of configuration planning.

Despite massive amounts of empirical data, generated through systematic experimentation, have become readily available, traditional approaches to configuration planning still largely rely on analysis through analytical modeling. There is a gap between the technical potential for large-scale data generation and methodologies that are suitable for efficient data evaluation and interpretation. In parallel, enterprise-class N-tier systems with web servers, application servers, and database servers are ever-growing in economic importance, infrastructure footprint, and application complexity. Classic performance analysis methods are challenged by this growth due to bottlenecks that so far have been considered rare and unusual. Moreover, non-stationary workloads and dependencies among tasks, which are commonly encountered in modern enterprise systems, may violate popular modeling assumptions such as single bottleneck queuing networks [16]. Unlike analytical approaches, methods based on actual empirical data do not rely on such rigid assumptions and are not susceptible to the aforementioned oversimplifications.

While classic design approaches to datacenters dictate investment in hardware capable of sustaining peak workloads, service oriented approaches suggest purchasing a base infrastructure and renting the rest. This trend not only calls for decision systems with flexible cost model support but also places particular emphasis on the optimization of operational expenditures [9]. Similarly, a key concern in green datacenter management are the rising costs of operation. New approaches are required to tame the energy costs

of enterprise information systems through adaptive provisioning. However, this is particularly difficult for large distributed systems with highly volatile workload processes [8]. New tools are necessary to provide decision makers with comprehensive understanding of the relationship between their computing performance landscape and their financial infrastructure constrains.

This paper addresses these developments and presents a novel approach to reliable configuration planning for clouds based on empirical data. Our approach is founded on an interactive and iterative data refinement process that enables configuration planners to follow intuitive data aggregation steps, leading from raw data to high-level configuration planning decisions. We further introduce the *CloudXplor Tool*, which prototypically implements our configuration planning approach within a web framework accessible from any generic web browser. Through this implementation, we were able to evaluate our configuration planning functionality on a large experimental dataset that has been previously collected.

The main contribution of this paper is our unique approach to configuration planning based on data refinement. More concretely, our methodology correlates economic goals with sound technical data to interactively derive intuitive domain insights at different aggregation levels. We have implemented our methodology as the *CloudXplor Tool* to provide a proof of concept and exemplify a concrete use case. *CloudXplor*, which can be modularly embedded in generic resource management frameworks, illustrates the benefits of empirical configuration planning. In general, this paper is a working example on how to navigate large quantities of technical data to form a solid foundation of economical decisions.

The data in this paper are part of an extensive experimental dataset, which was collected using software tools for automated system management. These data may be used to predict and manage N-tier system performance and utilization, and the results in this paper suggest the need for more studies on how to effectively take advantage of such large datasets.

With the goal of identifying non-rare phenomena with potentially wide applicability, our data analysis focused on representative benchmarking configurations very similar to their default settings. Unlike traditional system tuning work, we did not attempt to tune specific software products to find “the best a product can do” settings. Nevertheless, such tuning work is an interesting area for future work, and of particular importance for applied research in industry.

The remainder of this paper is structured as follows. In Section 2 we provide a brief overview of background on Service Level Agreements, experimental infrastructure, and multi-bottleneck phenomena. In Section 3 we outline our approach to configuration planning through empirical data refinement. Section 4 introduces the actual implementation of the *CloudXplor Tool*. In Section 5 we present a configuration planning case study based on actual empirical data. Related work is summarized in Section 6 before Section 7 concludes the paper.

## 2. BACKGROUND

This section provides background that is of particular importance for our approach. We briefly present our view on Service Level Agreements (SLAs), experimental infrastructure for data generation, and multi-bottleneck phenomena. Readers familiar with these aspects of our work may also directly skip to the configuration planning approach in Section 3.

### 2.1 Service Level Agreements

*CloudXplor* is designed to support the process of configuration planning for IT infrastructures with an explicit focus on economic

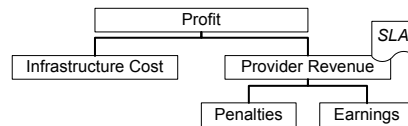


Figure 1: Profit model schema.

aspects. This perspective on the process demands the explicit definition of all relevant economic aspects. In order to provide an intuitive understanding, we define a *infrastructure cost* model and a *provider revenue* model that together reflect the cost and the value of the provided service. Modeling these two layers separately enables a cost-benefit analysis. Figure 1 shows the structure of the model. The profit is defined as the provider revenue of the system minus the infrastructure cost for providing the service. While it is usually easy to define a cost model for the operation of the infrastructure, the definition of a realistic provider revenue model is more complex. In the context of cloud computing, SLAs have become state-of-the-art. They define the level of service a provider guarantees to his customers. The SLA document usually contains the provider’s revenue model, determining the earning of the provider for SLA compliance as well as the penalties in case of failure. In the presented scenario the provider’s revenue is the sum of all earnings minus the sum of all penalties. The definition of reasonable SLAs is a non-trivial task because these agreements need to reflect economic value as well as customer service requirements. For instance, a mission-critical service should have higher earnings and penalties to set the right incentives for the provider to comply with the agreement. Furthermore, SLAs have to describe the common terms of business such as performance measures, metrics for the evaluation of the latter, legal and accounting issues, as well as exact contract periods. In the technical context of *CloudXplor*, the metrics for evaluating the performance characteristics of the system are of particular importance.

$$rev(rt_i) = \begin{cases} v & \text{if } 0 \leq rt_i \leq t_1 \\ v - c_1 & \text{if } t_1 < rt_i \leq t_2 \\ \vdots & \\ v - c_n & \text{if } t_n < rt_i \leq t_p \\ p & \text{otherwise} \end{cases} \quad (1)$$

$$\text{provider revenue} = \sum_i rev(rt_i) \quad (2)$$

Several methods exist for the performance evaluation of system monitoring data. A common method is the definition of Service Level Objectives (SLOs) that set a maximum response time for each request. The SLO applied in the our case study (Section 5) is more complex and is formally defined in Equation (1). The response time  $rt_i$  of each request  $i$  is evaluated according to the following formulation. For each successfully processed request  $rt_i < t_n$ , the provider receives a certain earning  $v$ . With increasing response time a late charge is applied in discrete steps. If the response time exceeds  $t_j$ , then a late charge of  $c_j$  is deducted from the earning. Solely if the response time drops below a predefined threshold  $t_p$ , the SLO is violated, leading to a penalty of  $p$ . The corresponding SLA in Equation (2), abstractly defines the provider revenue as the sum of all earnings and penalties for all request. The resulting profit is defined as the revenue minus the infrastructure cost. This type of SLA supports a reliable operation of systems because the provider is motivated to provide the service at a high

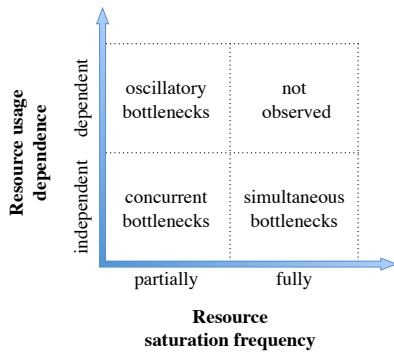


Figure 2: Simple multi-bottleneck classification [16].

quality. Whenever the provider is not able to meet highest performance standards, he is incentivized to continue to provide the service as his cash flow continues to be positive. Agreements with hard thresholds might lead to a situation in which the provider reduces the priority of services that he cannot satisfy because he has to pay a penalty anyways.

## 2.2 Experimental Infrastructure

The empirical dataset that is used in this work is part of an ongoing effort for generation and analysis of performance data from cloud information systems. We have already run a very high number of experiments over a wide range of configurations and workloads in various environments, ranging from large public clouds to small private systems. A typical experimentation cycle (i.e., code generation, system deployment, benchmarking, data analysis, and reconfiguration) requires thousands of lines of code that need to be managed for each experiment. The experimental data output are system metric data points (i.e., network, disk, and CPU utilization) in addition to higher-level metrics (e.g., response times and throughput). Although the management and execution scripts contain a high degree of similarity, the differences among them are subtle and important due to the dependencies among the varying parameters. Maintaining these scripts by hand is a notoriously expensive and error-prone process.

## 2.3 Single-bottlenecks vs. Multi-bottlenecks

This subsection provides a brief overview of bottleneck phenomena and their particular implications for configuration planning. Interested readers should refer to dedicated sources [15, 16] for a more comprehensive overview.

The abstract definition of a *system bottleneck* (or bottleneck for short) corresponds to its literal meaning as the key limiting factor for achieving higher system throughput. Consequently, this intuitive understanding has usually been consulted for analysis of bottleneck behavior in computer system performance analysis. Despite the convenience of this approach, these formulations are based on assumptions that do not necessarily hold in practice. For instance, the term bottleneck is often used synonymously with the term single-bottleneck. In a single-bottleneck case, the saturated resource typically exhibits linearly load-dependent average resource utilization that reaches 100 percent for large system loads. However, if there is more than one bottleneck resource in the system, bottleneck behavior typically changes significantly. This is the case for many real N-tier applications with heterogeneous workloads. Therefore, we explicitly distinguish between single-bottlenecks and the umbrella-term *multi-bottlenecks*.

Because system resources may be causally dependent in their usage patterns, multi-bottlenecks necessitate classification according to *resource usage dependence*. Additionally, multi-bottlenecks necessitate classification according to their *resource saturation frequency*. Resources may saturate for the entire observation period (i.e., fully saturated) or less frequently (i.e., partially saturated). Note that previous efforts in this area have typically omitted the notions of dependence and saturation frequency in their analysis (e.g., [14]). Figure 2 summarizes the classification that forms the basis of the multi-bottleneck definition as described above. It distinguishes between *simultaneous*, *concurrent*, and *oscillatory* bottlenecks. In comparison to other bottlenecks, resolving oscillatory bottlenecks is a very challenging task. Multiple resources form a combined bottleneck, which may only be addressed by considering the saturated resources in union. As a result, the addition of resources in saturated complex N-tier systems does not necessarily improve performance. In fact, determining regions of multi-bottlenecks through modeling may be an intractable problem. Consequently, multi-bottlenecks require measurement-based experimental approaches that do not oversimplify system performance in their assumptions.

## 3. EMPIRICAL CONFIGURATION PLANNING

Data refinement of experimentally derived data is a non-trivial and error prone task when done manually and without proper tooling support. In this section we introduce the interactive and iterative data refinement process that forms the basis of the configuration planning capability of the CloudXplor Tool.

In general, the data refinement process for configuration planning, as depicted in Figure 3, can be divided into four distinct data analysis modules<sup>1</sup>. Each of these modules can be characterized by a different degree of information density. The data transformation process itself is a direct result of the sequential application of various aggregation and filtering functions that are necessary to navigate, understand, and interpret the complex data space. In order to deal with the complexity and richness of the underlying data space, the transition process is multi-directional by design. Choices are made iteratively and interactively.

The **response time analysis** module (top left in Figure 3) allows analyzing the response time in the system. However, since the analysis of averaged values may easily lead to oversimplification, the data is aggregated in histograms that offer more detailed insights in the response time distributions. By fixing specific configurations and workloads, the user may zoom into the data and inspect response time distributions according to an a priori specified interval function. The latter is typically chosen according to an SLA function of interest. In the example graph (see Figure 8 for details), the interval function is specified as a mapping of six response time intervals, which are later mapped to specific revenue and penalty amounts as described in Section 2.1.

The **throughput analysis** module (top right in Figure 3) can be used to perform a throughput analysis of the system. In this stage, the data may be analyzed separately from other performance metrics, such as response time. In order to maximize the amount of displayed data, a three-dimensional graphical representation is used to allow the choice of two variable dimensions (e.g., number of SQL nodes, number of application servers, or workload). In cases where solely configuration parameters are chosen as axes, the throughput has to be aggregated in the z-dimension (e.g., maximal throughput).

<sup>1</sup>All included graphs reappear in the latter part of this paper.

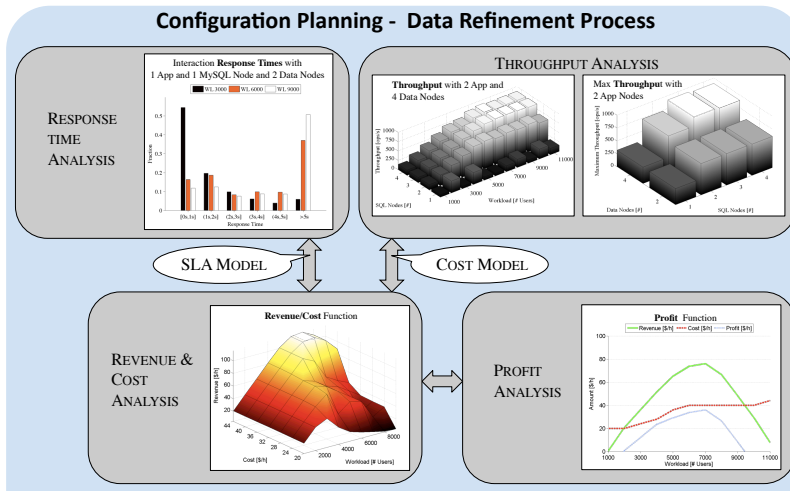


Figure 3: Schematic data refinement process in the CloudXplor Tool.

The **revenue and cost analysis** module (bottom left in Figure 3) combines the datasets from the response time and throughput analysis models and aggregates the data into a three-dimensional revenue function. This process requires the inclusion of two additional models. The response time data has to be correlated with the SLA function. This yields the revenue as illustrated on the z-axis (compare Figure 9). Additionally, the sizing information from the throughput analysis is correlated with a cost model, which yields the configuration cost as illustrated on the y-axis (compare Figure 9). In the simplest case, the cost model is a linear mapping between the hardware cost and the number of nodes in the system. In general, the revenue and cost are subject to the changing workload conditions (x-axis).

The **profit analysis** module (bottom right in Figure 3) can be used to assess the optimal workload size for the system in terms of profit. In the transition between the revenue and cost module and the profit module, the three-dimensional relationship between system load, system cost, and revenue are aggregated to investigate the dataset from an economical perspective. Economic reasoning dictates that the actual size of the chosen infrastructure is directly (and solely) implied by a profit maximization scheme. More concretely, as long as the profit is being maximized, the decision maker does not care whether he is running a large or small infrastructure. Given a certain workload, the maximal profit can be found by calculating the cost of each infrastructure and subtracting this value from the corresponding revenues. Once the infrastructure cost dimension is collapsed, the two-dimensional output can be directly used to determine the workload that yields the optimal profit for the application and system under test. The final output are revenue, cost, and profit functions. Each point on the workload span (x-axis) corresponds to an optimal configuration that is unambiguously mapped through the data aggregation in the last transition (i.e., profit maximization).

#### 4. TOOL IMPLEMENTATION

Our configuration planning methodology has been prototypically implemented in the CloudXplor Tool. This application has been developed in Microsoft Visual Studio 2008 as a light weighted web application. To enable fast and complex data processing, a MySQL server is integrated into the tool for data storage as well as an interface to remotely call Matlab (i.e., an language for technical com-

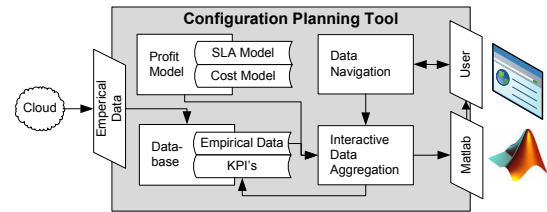


Figure 4: Schema of CloudXplor Tool.

puting) programs. Figure 4 depicts the main program structure and the tool environment. CloudXplor consists of four main units and three interfaces.

The first interface imports the empirical data, generated on an arbitrary cloud with an arbitrary benchmark. The second interface allows our tool to utilize the functionality of Matlab programs, which is our method of choice for complex calculation tasks and graphical rendering. The third interface exposes the functionality of CloudXplor to the end user. Furthermore, the tool functionality and analysis results can also be directly exported through this interface as a service. This allows the integration of CloudXplor into generic resource management frameworks, which enriches the usability of the tool in more involved settings.

The program logic of CloudXplor is divided into four units. The foundation of the data refinement process is provided by the Profit Model Unit and the Database Unit. The latter contains the empirical data to enable fast access during the analysis. The former (i.e., the Profit Model) derives economic key figures based on the performance behavior of the system by utilizing the SLA model and the cost model. The Data Navigation Unit provides the logic for the data refinement process verifying the validity of user commands. These input parameters are sent to the Interactive Data Aggregation Unit that aggregates and correlates empirical and economic data. The results of this process are sent to the Matlab interface for further processing as well as for graphical rendering with external Matlab modules. The final analysis results are transferred to the user interface. All useful Key Performance Indicators (KPIs), generated during the refinement process, are stored in the CloudXplor

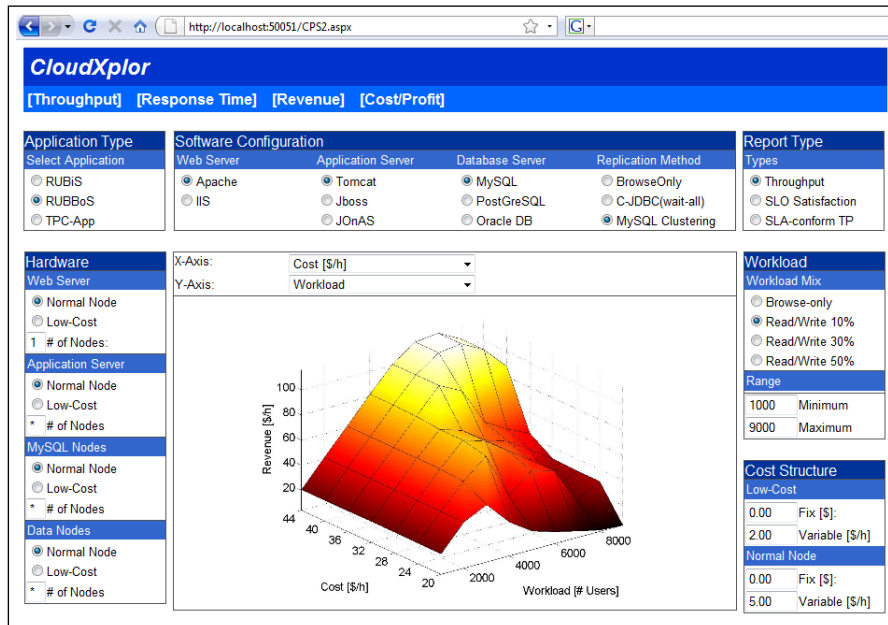


Figure 5: Screenshot of current implantation of the CloudXplor Tool.

database. These KPIs can be used for the comparison of different setups or accessed by other tools for further high level processing.

Figure 5 shows a CloudXplor Tool screenshot that was taken during the execution of the data refinement process that is subject of Section 3. The key element of the data refinement process is the central graph. It provides an intuitive view on system performance metrics as well as on economic data. The user can interactively change parameters to evaluate the impact of configuration changes in real-time. A variety of options is provided in the controls aligned around the graph, such as hardware and software configuration as well as economic and workload parameters. While the option fields require distinct input, the numeric input boxes can either be set to a certain value or a wild card. In the latter case, CloudXplor analyzes the scenario for each valid value of this field. The results are presented in the graph, whereby the user can assign each metric to each axis. In case multiple input parameters remain flexible, the user can specify a concrete report type. CloudXplor then uses the best setting for each flexible parameter for the graph. This allows an easy comparison of different configurations, which implies more intuitive assessment of economic impact of this parameter range.

## 5. CASE STUDY

In this section we present a case study that exemplifies the use of CloudXplor on an actual empirical data set. We first introduce the dataset in terms of benchmark application, software, and hardware setup. In the second part of the section, we detail the output of the case study performed with our tool.

### 5.1 Setup

Among N-tier application benchmarks, the Rice University Bidding System (RUBBoS) has been used in numerous research efforts due to its real production system significance. In our experiments, each experiment run consist of an 8-minute ramp-up, a 12-minute run period, and a 30-second ramp-down. Performance measurements (e.g., response time or CPU utilization) are taken during the run using the benchmark’s client generator module or Linux

Function	Software
Web server	Apache 2.0.54
Application server	Apache Tomcat 5.5.17
Database server	MySQL-cluster 6.2.15
Operating system	GNU/Linux Redhat FC4 Kernel 2.6.12
System monitor	Systat 7.0.2

Table 1: Software setup.

account logging utilities (i.e., Sysstat) with one-second intervals. Readers familiar with this benchmark can directly refer to Table 1, which outlines the concrete choices of software components used in our experiments.


RUBBoS [2] is an N-tier e-commerce system modeled on bulletin board news sites similar to Slashdot. The benchmark can be implemented as 3-tier (web server, application server, and database server) or 4-tier (with the addition of cluster middleware such as C-JDBC) systems. The benchmark places high load on the database tier. The workload consists of 24 different interactions (involving all tiers) such as register user, view story, and post comments. The benchmark includes two kinds of workloads: browse-only and read/write interaction mixes. Typically, the performance of benchmark application systems depends on a number of configurable settings (including software and hardware). To facilitate the interpretation of experimental results, we chose configurations close to default values. Deviations from standard hardware or software settings are spelled out when used.

The data in this section are generated from a set of experiment that were run in the Emulab testbed [1], which provides various types of servers. Table 2 contains a summary of the hardware used in this paper. Normal nodes were connected by a 1Gbps network. The experiments were carried out by allocating a dedicated physical node to each server.

### 5.2 Results

Due to the space constraints of this article, we can solely include a



Type	Components		
Normal 	Processor	Xeon 3GHz	64-bit
	Memory	2GB	
	Network	6 x 1Gbps	
	Disk	2 x 146GB	10,000rpm

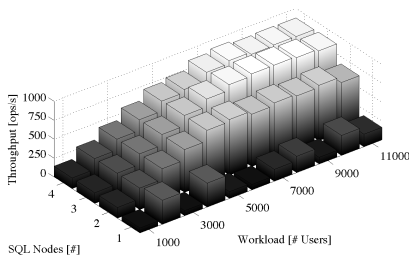
**Table 2: Hardware setup.**

few sample graphs, which comprise a tiny subset of the actual navigable space of our data. More concretely, the data that is shown in this section is limited to read/write workload with ten percent write interaction frequency. Furthermore, the workload spans 1,000 to 11,000 users in steps of 1,000. The user numbers reflect generated client threads, that interact with the system based on a Markov transition probability matrix, where each state has a exponential think time with seven second mean.

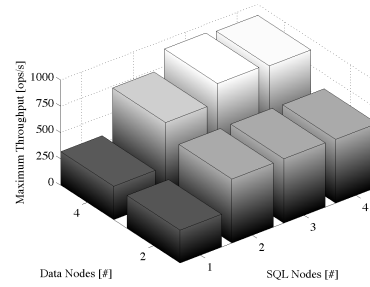
Figure 6 shows the throughput of the system with one web server, two application servers, and four data node servers under read/write workload. As the workload (x-axis) increases, the system bottlenecks at a workload of 2,000 users for the configuration with a single SQL node (y-axis). This bottleneck can be resolved by adding more SQL nodes. However, once there are three SQL nodes in the system and a workload of 7,000 users is reached, the system throughput can no longer be increased through the addition of SQL nodes. This analysis yields the assumption that the system bottleneck has shifted elsewhere or has potentially become a multi-bottleneck between multiple server types.

A different type of analysis seems to offer a potential explanation for the observed system behavior. Figure 7 shows the maximal system throughput that is achievable with one web server and two application servers. In contrast to the previous graph, there are two different kinds of bottlenecks that are successfully resolved in this dataset. First, there is a SQL node bottleneck between one and two SQL nodes. Second, there is a data node bottleneck that is resolved between the configuration with two SQL nodes and two data nodes and the configuration with two SQL nodes and four data nodes. After that, the addition of another SQL nodes again increases performance to a maximal system throughput around 900 interactions per second. This analysis suggests, that the addition of further data nodes might increase the overall system throughput even further. Note that it is due to the implementation specifics of the MySQL clustering mechanism, that the number of data nodes may only be increased in powers of two.

In parallel to the analysis of the throughput, an analysis of the response time may be performed. Figure 8 shows three sample response time distributions. The underlying configurations for these



**Figure 6: Throughput of a RUBBoS system with 1 web, 2 application, and 4 data nodes servers under read/write workload.**



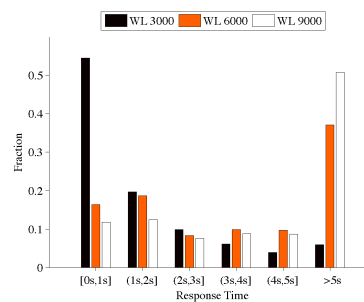
**Figure 7: Maximal throughput of a RUBBoS system with 1 web and 2 application servers under read/write workload.**

Response Time Interval	Revenue/Penalty
[0s, 1s]	0.0033 cent
(1s, 2s]	0.0027 cent
(2s, 3s]	0.0020 cent
(3s, 4s]	0.0013 cent
(4s, 5s]	0.0007 cent
> 5s	-0.0033 cent

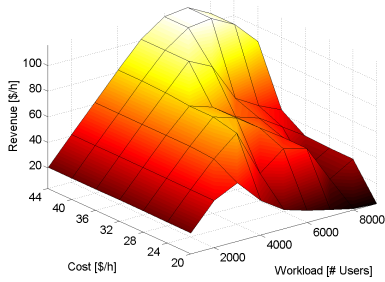
**Table 3: SLA Model**

graphs are one web server, one application server, one MySQL server, and two data node servers. The depicted workloads are between 3,000 and 9,000 users in steps of 3,000. The histogram intervals have been chosen according to the SLA model, which is assumed in this case study. This model is designed according to the formulation in Section 2.1 and summarized in Table 3. In this way the graph can be easily interpreted in its economical context. At a workload of 3,000 users, the system is largely able to meet SLA demands. Consequently, over 50 percent of all user interactions result in a full revenue payoff. However, as the workload increases, the (relatively small) system gets overloaded very quickly. The response time distributions become highly right-skewed with high percentages of penalized interactions. In the case of a workload of 9,000 users, over half of all interactions are penalized as unsuccessful (i.e., response time is greater than five seconds).

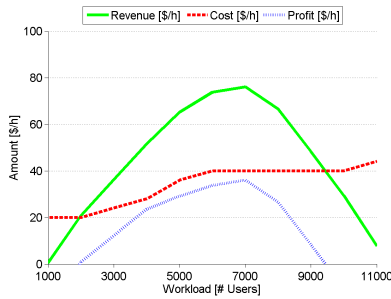
After exploring throughput and response time separately, the data can be combined in a unified analysis under economical aspects. Following the data refinement process depicted in Figure 4, the response time data need to be transformed with the SLA model



**Figure 8: Response time distributions of a RUBBoS system with 1 web, 1 application, 1 MySQL, and 2 data node servers under read/write workload.**



**Figure 9: Revenue and cost analysis of a RUBBoS system under read/write workload.**



**Figure 10: Profit and cost analysis of a RUBBoS system under read/write workload.**

(Table 3), and the throughput data need to be transformed with a cost model. For simplicity, we assume that the usage cost for each server node are uniform at a price of four dollars per computing hour. Note that CloudXplor is also able to implement any arbitrary cost model through a direct mapping of each configuration to a fixed and variable cost component.

The transformation results are shown in Figure 9. Another transformation that was applied to the data is a direct result of economical reasoning. Although technically possible due to the real-system character of this investigation, decreasing revenue under constant workload and increasing resource costs has been removed through maximization. In other words, the graph has been transformed to be monotonically increasing along the y-axis (i.e., configuration cost). The analysis of the three-dimensional graph reveals two highly significant insights. First, the revenue grows evenly across all configurations for low workloads (i.e., constant slope plane for workload between 1,000 and 5,000 users). Second, the ridge of the graph runs diagonally between 2,000 users in the cheapest configuration and 7,000 users in the high-end version. Past a workload of 7,000 users, all configuration variations result in decreasing revenue due to frequent SLA violations and corresponding penalties. This means that the system under test is not able to further increase profitability by sustaining more than 7,000 concurrent users.

From an economic perspective, it is not significant how a particular revenue is generated as long as the profit of the enterprise is maximized. Therefore, it is beneficial to reduce the complexity of the three-dimensional data into a single two-dimensional representation. This can be done by optimizing the profit along the cost axis. The result is shown in Figure 10. This figure immediately reveals the economic impact of any arbitrary workload situation

Workload [# Users]	Opt. Configuration
1,000	1/1/1/2
2,000	1/1/1/2
3,000	1/1/2/2
4,000	1/2/2/2
5,000	1/2/2/4
6,000	1/2/3/4
7,000	1/2/3/4
8,000	1/2/3/4
9,000	1/2/3/4
10,000	1/2/3/4
11,000	1/2/4/4

**Table 4: Profit optimal configurations.**

within the examined workload span. Concretely, decision makers are able to assess the profitability of their system directly from usage statistics. On the other hand, this aggregation can also be automatically resolved to correlate each workload situation with its profit maximizing cloud configuration.

The profit optimal configurations of the examined system are shown in Table 4. We use a four-digit notation #W/#A/#C/#D to denote the number of web servers, application servers, SQL nodes, and data nodes. The table shows that the data analysis process revealed a unique configuration plan that can be used to provision the system under the premise of optimal profitability.

## 6. RELATED WORK

Traditionally, performance analysis in IT systems postulates models based on expert knowledge and uses standard statistics to parameterize them based on some experimental dataset [11, 13]. Queuing models have been widely applied in many successful performance prediction methodologies [20, 21]. Nonetheless, these approaches often suffer from generality limitations due to their rigid assumptions when handling all evolution of large applications. For example, assuming the availability of extensive instrumentation data profiles [20] or constant mean inter-arrival times of request [21] do not hold in general because many real parameters may be subjected to high variability. Moreover, non-stationarity of workload is very common in N-tier systems. This characteristic has been exploited for performance prediction by Stewart et al. [19]. The authors also explain in their work how their anomaly detection can be used to invoke a bottleneck detection process. More recently, statistically induced models have been suggested to automate the model generation process and remove the dependence on expert knowledge [3, 6, 10]. In fact, extensive experiments have been conducted to compare different bottleneck detection methodologies [15].

In parallel to the technical complexity, economic considerations have recently become a key issue in IT system operation. This new awareness on sustainable operation modes can be seen in the emerging trend of Green IT [18]. Although this term is primarily ecologically motivated, it is also closely related to efficiency in the domain of datacenter operation. For IT operation, environmental goals are largely congruent with economic objectives because the major Green IT scope is to increase efficiency in this case [17]. One option is the enhancement of isolated units such as power supplies [23]. Another more holistic approach is utilization optimization. For instance, virtualization and consolidation efforts are well-established concepts from this area [17]. However, these concepts may not always be applicable. Large information systems with volatile workload processes, for example, might have too complex performance patterns, which requires a thorough an in-depth understanding of the system behavior in order to satisfy QoS [8].

The development of advanced Service Level Management (SLM) concepts, defining the terms of businesses between providers, has

recently become a very active research topic [22]. Defining feasible SLAs is non-trivial for the providers because they need to supply the guaranteed QoS while operating efficiently. With the emerging trend of cloud computing, the importance of SLM has even grown further. The benefits of cloud computing in enterprises have been previously ascertained in theory. Dias de Assuncao et al. investigated the potential of combining cloud and owned infrastructure resources to achieve higher performance [7]. Hedwig et al. developed a model to determine the optimal size of an enterprise system that satisfies peak demands with remote resources [9]. Both works suggest the inclusion of cloud resource into efficient production systems. Ragusa et al. developed a prototype of such a system able to automatically scale by including remote resources. Despite these efforts, Buyya et al. examined the current state-of-the-art in cloud markets concluding that today's implementations do not fulfill the requirements of modern enterprise systems [5]. Moreover, Risch and Altmann empirically verified this argument empirically verified this argument by showing that cloud computing is seldom used by enterprises [4]. One significant reason is that cloud providers usually solely guarantee best effort. More concretely, agreeing to specific SLAs bears significant economic risks due to the complexity of the underlying infrastructures.

Resource Management Systems (RMSs) are a popular concept to control decentralized environments. Nonetheless, Krauter et al. developed a taxonomy for today's RMSs showing that economic aspects are rarely sufficiently integrated [12]. CloudXplor closes this gap by correlating SLAs to technical properties and deriving their economic implications.

## 7. CONCLUSION

Recent research has established configuration planning of modern IT systems as particularly difficult for a number of reasons. Among others, the popularity of cloud computing and trends such as green resource management challenge current practices. In this paper we have presented a support tool to help decision makers find sustainable configurations that are systematically designed according to economic principles. Our data based approach is novel because it is founded on a unique methodology to combine economic goals with technical data. We have provided a proof of concept by implementing our methodology as the web application. The latter is modular and can be regarded as a working example on derivation of economical insights from technical data through a systematic refinement process.

Our current and future work include augmenting the CloudXplor Tool (i.e., our configuration planning methodology) with a workload analysis module that is able to provide support for loading traces and time series analysis. Furthermore, we intend to extend the tool's comparison functionality to generate intuitive results with joint comparison of various hardware infrastructures.

## 8. ACKNOWLEDGMENTS

This research has been partially funded by National Science Foundation grants ENG/EEC-0335622, CISE/CNS-0646430, CISE/CNS-0716484, AFOSR grant FA9550-06-1-0201, NIH grant U54 RR 024380-01, IBM, Hewlett-Packard, Wipro Technologies, and Georgia Tech Foundation through the John P. Imlay, Jr. Chair endowment. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or other funding agencies and companies mentioned above.

## 9. REFERENCES

- [1] Emulab - Network Emulation Testbed. <http://www.emulab.net>.
- [2] RUBBoS: Bulletin board benchmark. <http://jmob.objectweb.org/rubbos.html>.
- [3] M. K. Aguilera, J. C. Mogul, J. L. Wiener, P. Reynolds, and A. Muthitacharoen. Performance debugging for distributed systems of black boxes. In *SOSP '03*.
- [4] J. Altmann, M. Ion, A. Adel, and B. Mohammed. A taxonomy of grid business models. In *Gecon '07*.
- [5] R. Buyya, C. S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *HPCC '08*.
- [6] I. Cohen, M. Goldszmidt, T. Kelly, J. Symons, and J. S. Chase. Correlating instrumentation data to system states: a building block for automated diagnosis and control. In *OSDI '04*.
- [7] M. D. de Assuncao, A. di Costanzo, and R. Buyya. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In *HPDC '09*.
- [8] M. Hedwig, S. Malkowski, and D. Neumann. Taming energy costs of large enterprise systems through adaptive provisioning. In *ICIS '09*.
- [9] M. Hedwig, S. Malkowski, C. Bodenstain, and D. Neumann. Datacenter investment support system (daisy). In *HICSS '10*.
- [10] C. Huang, I. Cohen, J. Symons, and T. Abdelzaher. Achieving scalable autoamated diagnosis of distributed systems performance problems. Technical report, HP Labs, 2007.
- [11] R. Jain. *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, Inc., New York, NY, USA, 1991.
- [12] K. Krauter, R. Buyya, and M. Maheswaran. A taxonomy and survey of grid resource management systems for distributed computing. *Softw. Pract. Exper.*, 32(2):135–164, 2002.
- [13] D. J. Lilja. *Measuring Computer Performance - A Practitioner's Guide*. Cambridge University Press, New York, NY, USA, 2000.
- [14] M. Litoiu. A performance analysis method for autonomic computing systems. *ACM Trans. Auton. Adapt. Syst.*, 2(1), March 2007.
- [15] S. Malkowski, M. Hedwig, J. Parekh, C. Pu, and A. Sahai. Bottleneck detection using statistical intervention analysis. In *DSOM '07*.
- [16] S. Malkowski, M. Hedwig, and C. Pu. Experimental evaluation of n-tier systems: Observation and analysis of multi-bottlenecks. In *IISWC '09*.
- [17] G. Schulz. *The Green and Virtual Data Center*. Auerbach Publications, Boston, MA, USA, 2009.
- [18] A. Singh, B. Hayward, and D. Anderson. Green it takes center stage. Springboard Research, 2007.
- [19] C. Stewart, T. Kelly, and A. Zhang. Exploiting nonstationarity for performance prediction. *SIGOPS Oper. Syst. Rev.*, 41(3):31–44, 2007.
- [20] C. Stewart and K. Shen. Performance modeling and system management for multi-component online services. In *NSDI '05*.
- [21] B. Urgaonkar, G. Pacifici, P. Shenoy, M. Spreitzer, and A. Tantawi. An analytical model for multi-tier internet services and its applications. *SIGMETRICS Perform. Eval. Rev.*, 33(1):291–302, 2005.
- [22] R. Vahidov and D. Neumann. Situated decision support for managing service level agreement negotiations. In *HICSS '08*.
- [23] T. Velte, A. Velte, and R. C. Elsenpeter. *Green IT: Reduce Your Information System's Environmental Impact While Adding to the Bottom Line*. McGraw-Hill, Inc., New York, NY, USA, 2009.