In: Proceedings of the 13th Annual Workshop of HP OpenView University Association, Hosted by the University of Nice at Cote d'Azur May 21 - 24, 2006

Combining Application Instrumentation and Simulation to Forecast Costs and Revenue in Application Service Provisioning Scenarios

M. Schmid, J. Schaefer, R. Kroeger
University of Applied Sciences Wiesbaden
Distributed Systems Lab
Wiesbaden,
Germany
{schmid|jan.schaefer|kroeger}@
informatik.fh-wiesbaden.de

K. Sotnikov, Y. Karpov
St.Petersburg State Polytechnical University
Distributed Computing
and Networking Department
St. Petersburg,
Russia
sotnikov@dcn.infos.ru, karpov@xjtek.com

Abstract

The paper presents an approach for complementing an existing application service provisioning architecture with a simulation component that is calibrated using measurements from the production system. The overall aim of the approach is to support service providers in planning and refining tariffs, adjusting system capacity, and estimating costs and profit.

Keywords

QoS, Simulation, Instrumentation, Accounting, Charging

1. Motivation

The economic pressure on enterprises in recent years has significantly increased the need for outsourcing IT services to specialised providers. Not so long ago, service provisioning contracts mainly existed in the domain of network provisioning. In the meantime, service providers have enhanced their portfolio to also offer application level services.

Outsourcing of services or applications to external companies requires the definition of a legal framework. Service Level Agreements (SLAs) [3, 7] are a formal representation of agreed-upon contracts between service providers and customers. SLAs are an important element in the outsourcing business as they define the Quality of Service (QoS) of a provided service. Typically, an SLA defines SLA parameters that are related to performance and availability, as well as Service Level Objectives (SLOs) that provider and customer agreed upon. In order to assure SLA compliance, SLA parameters have to be monitored at runtime, and the measured QoS has to be compared with the SLOs. In addition, SLAs define prices for service provisioning and penalties for the provider in case of SLO violation.

To succeed in a competitive environment, flexible pricing models are vital for service providers. Customers increasingly demand pay-per-use pricing schemes that reflect the customers actual service usage. Looking at network provisioning, a pay-per-use scheme is set up using standard network traffic measurement tools. In the field of application service provisioning, a pay-per-use scheme ideally may be realised by charging customers

In: Proceedings of the 13th Annual Workshop of HP OpenView University Association, Hosted by the University of Nice at Cote d'Azur May 21 - 24, 2006

per business transaction. However, currently often CPU and memory usage are utilised as resource level indicators for service usage, which results in coarse-grained monitoring of customer behaviour.

Currently, the importance of on-demand usage of services and resources raises massively. Providers start to build up whole data centres on a pay-per-use basis, where customers simply rent computing power or application usage based on agreed tariffs instead of buying physical machines. This trend is complemented by a virtualisation of data centre resources.

Independent from the kind of services they offer, providers are interested in predicting costs (caused by service usage) and revenue for optimising capacity planning and for estimating the impact of pricing changes on the overall revenue.

[1] define an architecture for Management by Business Objectives (MBO) that predicts the impact of management decisions on the compliance with previously defined SLOs. As a result, the architecture proposes the most promising course of action.

In contrast to that, we present an approach that combines application instrumentation with simulation in order to help service providers in forecasting future developments. The approach is currently tested in a lab environment, using a fully performance-instrumented, multi-tier architecture for running e-business applications.

Section 2 describes a distributed e-business application setting that is attached to a charging and accounting architecture for distributed services. In section 3, we describe our current simulation approach. Section 4 presents a prototypical implementation of the architecture and summarises next steps.

2. Accounting Architecture and Test-bed

The University of Applied Sciences in Wiesbaden developed a flexible architecture for business transaction-based charging and accounting of distributed applications [6]. The architecture allows to define customer - provider relationships using tariffs and SLAs. Tariffs may be defined based on business transactions, which are measured within the application environment. A tariff may define a complex pricing scheme. Associations between tariff and prices contain pricing conditions that control the selection of relevant prices. SLA violations on the provider side might lead to price reductions or even penalty payments.

A second part of the architecture deals with detailed accounting of service provisioning costs on the provider side. Provisioning costs may be defined as a combination of usage costs, licensing costs, costs of purchase, and so on. Usage costs are based on the same measurements that are used in tariffs for charging per business transaction.

Main benefits of the architecture are the possibility to combine business-oriented and technical view of a service-provisioning scenario, as well as the possibility to define tariffs that are based on business transaction level usage information.

In our test bed, a fully performance-instrumented, distributed, multi-tier e-business application system [2] provides usage measurements for the charging and accounting architecture. The application system consists of several middleware components (squid proxy server, Apache web server, Apache Tomcat web container, JBoss EJB server, MySQL

DBMS), which have been performance-instrumented using the Open Group Application Response Measurement (ARM) API [4]. This instrumentation allows tracking of user requests through the multi-tier architecture by executing fine-grained performance measurements.

For this work, the concrete accounting architecture and the instrumented e-business application are used as a case study. In principal, the approach works with any application system that provides the necessary instrumentation.

3. The Simulation Approach

Figure 1 depicts the general architecture of our simulation approach. Service providers may analyse current costs and revenue based on the measurements performed in the production system while operating the service. This is displayed in the upper part of figure 1. These production system measurements allow a very limited prediction of future scenarios.

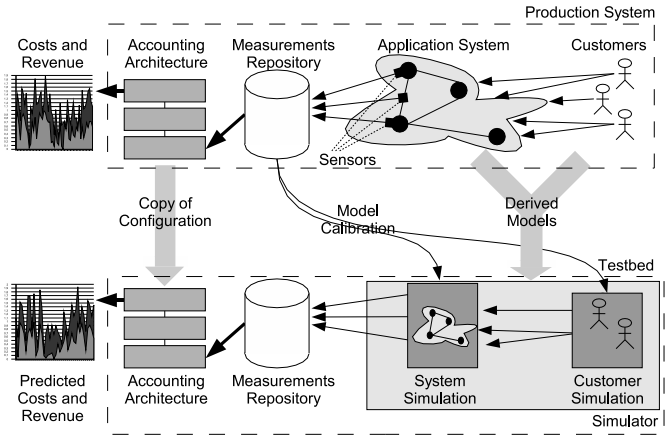

Figure 1: Overview of the system simulation approach

To allow service providers to perform a detailed analysis of cost and revenue progression and prediction, we developed simulation models for customers and services based on the real production system. The simulation models are used to generate usage information for the measurement repository (depicted in the lower part of figure 1). The accounting system should be a copy of the one used in the production system. To provide realistic results, simulated application system entities are calibrated using historical measurements from the real system. This link between a real production system and the simulation is the main advantage of our approach, compared to other simulation approaches (e.g., [5]).

Some individual flows in an IT infrastructure or behaviours of separate entities in a simulation model can be described with a high degree of accuracy using a mathematical model for the whole system. For example, a server response time can be described using a probability distribution gained from statistical measurements. However, when talking

about architectures with complex structures, interrelations and inner processes, it is impossible to create a single tractable mathematical model. We solved this problem using simulation as this provides the ability to create a virtual computer-based representation of the entire architecture.

When the service provider is contacted by customers in order to establish a relationship, the customer is registered with the accounting architecture. Periodically the service provider triggers the generation of customer bills, and delivers the bills to the customers. In case a customer terminates the relationship, the customer is unregistered again.

3.1 Customer simulation

The simulation represents customer behaviour in state machines (see figure 2 for an example). With a certain probability, customers do establish a customer-provider relationship with the service provider. A customer possesses a certain budget (B), which is refilled once per month. Depending on the size of his budget and his satisfaction concerning response times, a customer performs a number of operations on the system, where the request rate (R) represents a customer's initial intensity of system usage. Once a month, the customer receives a bill from the service provider, which reduces the budget depending on the number of billable business transactions. Initially, a customer is in the `normal` state. If the number of SLA breaches (F) exceeds a certain percentage (k) of the last N requests (with $N = 100$ in the example), the customer changes to the `unsatisfied` state and uses the new request rate $\frac{R}{2}$.

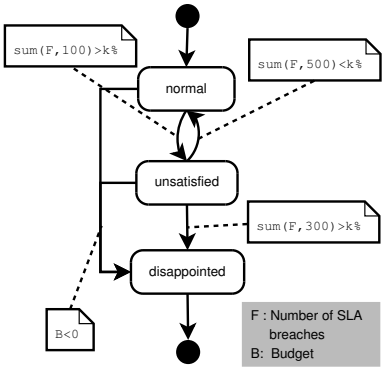

Figure 2: Example for a state machine representing a Customer

If the number of SLA breaches exceeds $k\%$ for the next M requests (with $M = 300$ in the example), the customer changes to the `disappointed` state. Otherwise he changes back to normal. If the customer is disappointed, he terminates the relationship with the service provider. To date, the model does not cover relationship termination due to better offers from competitors.

Variables, which currently influence the customer behaviour, are the probability of establishing a relationship, the customer's monthly budget, the request rate R, and the con-

stant k that represents the sensitivity to SLA breaches. Customer models may be refined based on customer behaviour statistics.

3.2 System simulation

The system is invoked by customers and records system usage as generated measurement records in a repository. With a defined probability, the system fails to comply with the SLAs defined beforehand. In that case the customer's satisfaction decreases. The system behaviour currently can be customised as follows: probability of SLA violation in relation to load (requests per time period), average value and standard deviation for the generation of measurement information, as well as average increase and standard deviation for situations where SLAs are violated.

3.3 Model Calibration

The simulator is calibrated using measurement data from the production system to be simulated. Customers are grouped into clusters, based on their system usage frequency. For each cluster, the request rate R and the budget B are set to the average values of that group. k is currently adjusted manually to reflect customers' behaviour in the real world system.

The response time behaviour of the simulation system is calibrated by measurements from the production system, from which the simulator interpolates response times for concurrent system access. As a default, the maximum number of concurrent requests is extracted from the production system measurements.

In order to check the appropriateness of the simulation settings, the simulator may be run in parallel to the production system, using the same settings and tariffs. However, periodical readjustments are necessary as the simulation model currently is still rather simple.

The main goal for this approach is to provide new possibilities for analysing *what-if?*-questions that cannot be answered based on historical usage-data alone. Examples for these questions are: *"How does the overall profit change when I increase the capacity of my services (faster response times vs. higher costs)?"* or *"How many customers can use a service without a significantly reduced QoS?"*. Users may modify simulator settings (concerning capacity, load, number and types of customers, sensitivity to SLA breaches, etc) or implement new tariffs in the accounting system in order to answer these questions.

4. Prototypical Implementation

For a prototypical implementation we used the AnyLogic* simulation engine to model the service provider, the application system, and customers, which interact to drive the accounting architecture from the simulator. AnyLogic was used, because it offers a wide range of modelling approaches including discrete, continuous and hybrid modelling, as well as stochastic and deterministic modelling and comes with a number of predefined libraries including a discrete event systems modelling library. The simulation engine sup-

*See http://www.xjtek.com

ports the integration of the resulting model with other software, which is the basis for the interaction with an accounting architecture.

The prototype implements the architecture that has been presented in section 3. The flexible realisation allows modifying structure, specification, parameters and other features of the simulated system easily.

The resulting system allows tracking of customer outflows and satisfaction dynamics as well as services' costs and other financial indicators, which complement the knowledge gained from an analysis of the accounting architecture results. The simulator is able to provide a number of forecasting information views, a testing facility, as well as strategy optimisation and gaming possibilities. As a result one may observe a variety of statistics as an answer to a *what-if?*-scenario and as a basis for strategy, and tariff optimisation.

Challenges are currently seen in a fully automated calibration of the system according to historical performance measurements and in the calibration of customer behaviour. This currently is a manual process.

By refining the model, it is possible to tune the simulator to get improved and more realistic results. Future activities in this field will be: refinement of customer simulator based on more sophisticated state machines. An event-driven model of a customer will allow us to calibrate satisfaction with tariffs and the provided service based on historical data. Agent technology allows to define the behaviour of individual entities as well as targeting groups while observing their integral characteristics.

We will test the prototype with different tariffs, SLAs and SLOs to observe and analyse the output dynamics of the simulator.

References

[1] C. Bartolonini, M. Salle, A. Boulmakoul, S. Sabiani, and J. M. Bronoel. IT Service Incidents Prioritization driven by Business Objectives. In Bruno F. Marques, Thomas Nebe, and Raul F. Oliveira, editors, *Proceedings of the 12th Annual Workshop of HP OpenView University Association*, pages 29–41, 2005. http://www.hpovua.org.

[2] Markus Debusmann, Marc Schmidt, Markus Schmid, and Reinhold Kroeger. Measuring Service Level Objectives in a Complex Web-Based e-Business Environment. In *10th HP OpenView University Association Workshop*, Jul 2003. http://www.hpovua.org/.

[3] Lundy Lewis. *Managing Business and Service Networks*. Kluwer Academic / Plenum Publishers, 2001.

[4] The OpenGroup. *Application Response Measurement (ARM) Issue 4.0, V2 - C Binding*, 2004. http://www.opengroup.org/management/arm/.

[5] Timofei Popkov, Yuri Karpov, and Maxim Garifullin. Using Simulation Modelling for IT Cost Analysis. In *10th HP OpenView University Association Workshop*, Jul 2003. http://www.hpovua.org.

[6] Markus Schmid, Markus Debusmann, Reinhold Kroeger, and Marc Halbig. Flexible Charging and Accounting of Distributed Services. In *12th HP OpenView University Association Workshop*, Jul 2005. http://www.hpovua.org.

[7] Rick Sturm, Wayne Morris, and Mary Jander. *Foundations of Service Level Management*. SAMS Publishing, April 2000.