Cost Accounting for Shared IT Infrastructures

Estimating Resource Utilization in Distributed IT Architectures

The Authors

Reinhard Brandl
Martin Bichler
Michael Ströbel

Dipl.-Wi.-Ing. Reinhard Brandl
Prof. Dr. Martin Bichler
Technische Universität München
Fakultät für Informatik
Lehrstuhl für Internetbasierte Geschäftssysteme
Boltzmannstr. 3
85748 Garching
{brandlr | bichler}@in.tum.de

Dr. Michael Ströbel
BMW Group
80788 München
michael.stroebel@bmw.de

Eingereicht am 2006-10-12,
nach zwei Überarbeitungen
angenommen am 2007-01-19
durch Prof. Dr. W. König

Executive Summary

Cost allocation of shared IT infrastructure such as server capacity or network equipment is technically difficult. We propose a method to derive adequate estimators for the resource consumption of a service invocation, which can then provide a basis for cost allocation keys.

- The expected resource consumption of services can be estimated with high accuracy in load tests. Measurements during regular operations can be omitted.
- We use Queuing Networks to validate the estimated resource profiles under different workloads.
- Experiments with multi-tier database applications in a heterogeneous environment yield surprisingly high accuracy of the estimated resource profiles.

The estimates also provide the necessary input for analytical capacity planning, which often suffers from a lack of data to parameterize respective Queuing Network Models.

Keywords: IT Infrastructure Cost Accounting, Usage-based Cost Allocation, Capacity Planning, Queuing Network Theory

1 Introduction

Allocation of IT infrastructure costs is an increasingly important topic in corporate IT departments. According to a study by Forrester[1] [Forr06] computer hardware (21 %) together with Networking and Communications hardware (14 %) account for more than one third of current IT budgets. Typically, these infrastructure costs (e.g., servers and networking infrastructure) are allocated to business units based on simplified cost allocation keys [Sysk02; GaMa05, p. 136]. Comprehensive IT Asset Management and chargeback solutions have found limited use as of yet [GaJK05]. Usage-based accounting and pricing should lead to higher transparency and has recently become a major topic for CIOs. Nowadays enterprise software architectures are mostly designed as multi-tier client-server applications. In such environments the determination of usage-based allocation keys is very challenging [Bert01]. If the hardware is dedicated to specific customers, for instance business units, those costs can be treated as direct costs. The actual resource consumption (e.g., CPU time or number of I/O operations) is not a relevant cost driver and can be ignored. However, more and more IT infrastructure is nowadays shared among multiple applications and business units. Typical examples include database servers, application servers or virtualized servers (e.g., using VMware [Vmwa] or Xen [UCCL]). In these cases the resource consumption of the installed applications and the workload generated by the customers is a significant cost driver. An average application server in an industrial data center (e.g., 4 CPU / 16 GB memory) can easily host dozens of light-weight applications, whereas in other workload scenarios, the same server may be fully utilized by one or two applications. If the costs for such a server are treated as indirect costs and are apportioned via flat rates or fixed percentages, the IT Management, as well as the concerned business units, has only few possibilities for cost controlling and planning. Furthermore, the server is usually not offered as standalone product, but as part of a larger IT system. A request in a 3-tier database application, for example, comprises a web server, an application server, and a database server, which are all typically used by different applications.

A cost accounting approach, which ignores the resource consumption, may lead to multiple free-rider problems. For example, application owners do not consider the resource requirements when selecting off-the-shelf software. Also, the owners of "light-weight" applications might have to bare a very high share of the cost for a particular application or database server, which in turn makes it more difficult to finance these applications. Obviously, a usage-based model, where IT infrastructure costs can easily be allocated to application owners or even to users directly increases cost transparency and would have a number of advantages. A technical possibility would be to determine consumption-based cost allocation keys by detailed monitoring and metering of each service request. This would require assigning a unique ID for each user to each database request and each thread running on an application server in order to determine exactly how much of the resources a service customer has consumed. It would force the adaptation of the entire IT infrastructure, cause a huge monitoring and metering overhead, and is typically not viable (irrespective of the fact that a business user probably would not accept technical accounting metrics as the CPU times of different servers).

In this paper, we propose a method to determine usage-based cost allocation keys for customer-oriented services based on their estimated resource consumption. Deriving such an estimator, however, is a nontrivial task.

- First of all, the estimator should be unbiased, in the sense that on average it should not over- or underestimate the true resource consumption.
- Second, the estimation should be applicable to various IT infrastructures (i.e., different hardware, operating systems and applications), without a need to change the respective systems.
- Third, the estimation should cause little extra work and integrate well with existing IT service management processes.

While the first requirement is essential for the cost allocation key to achieve incentive compatibility, the second and third requirements target the viability of the approach. Clearly, direct apportioning of IT infrastructure costs could easily outweigh the benefits of a usage-based cost allocation key.

Based on a series of load tests, we derive for every service a so-called resource profile as estimator for its true resource consumption. We consider CPU time, storage I/O, and network traffic, which typically are the scarce resources, and consequently the cost drivers and parameters for new investment decisions. This profile multiplied by the number of service invocations in an accounting period can then be the basis for a usage-based cost-allocation. The actual charges for business units might of course also be influenced by other non-cost-based factors, such as management incentives to use certain applications.

A main question is how well such an estimator meets the real resource consumption, given different workloads and different usage behavior in today's complex IT infrastructures. Only, if the estimates are of high accuracy, business units will accept a respective cost allocation key. We use Queuing Networks to validate the estimated CPU times and predict server utilization for different workloads. These predictions are compared and tested against the parameters measured in load tests with respective workloads.

We conducted several experiments with J2EE applications in a distributed client/server infrastructure consisting of Unix, Linux, and Windows servers in a data center of the BMW Group and achieved very promising results even in a very heterogeneous environment with multiple software modules, operating systems, and hardware infrastructures. The estimation procedure could be integrated with the IT service management processes at the central IT division of the BMW Group with little extra effort and is applicable to different types of IT infrastructure including server capacity, storage I/O, and network bandwidth.

The remainder of the paper is structured as follows. Section 2 briefly discusses current approaches. Section 3 describes the concept of allocating infrastructure costs via resource profiles. Sections 4 and 5 present methods for estimating resource profiles for IT services and describe how we use Queuing Network Models for validation and capacity planning. The concept is illustrated by experiments with the example application Java Pet Store from Sun Microsystems [SuMi]. Section 6 compares the approach with related work. The paper concludes with a short summary in section 7.

2 Survey of Literature and Current Practice

Cost accounting for IT infrastructures falls under the realm of *IT (Infrastructure) Controlling* (see for instance [Karg99; KrBR00; GaMa05]) or, from the ITIL perspective [OoOC01], *IT Financial Management*. Several concepts exist (e.g., cost-center accounting, process costing). The choice of an appropriate approach is mainly dependent on management requirements and the organizational structure of the IT unit (e.g., service center, cost center, profit center). Overall, cost accounting can be divided into three major steps (figure 1) [Gart03].

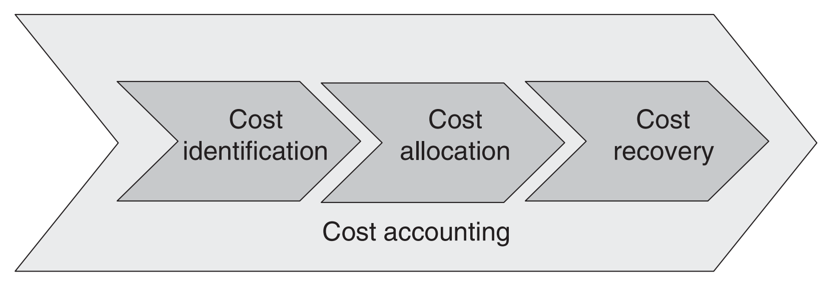


Figure 1 The cost accounting process (adapted from [Gart03])

Cost identification is an organizational/accounting issue. It focuses on making IT costs visible and assigning them to accounting objects, such as the provided IT services. *Cost allocation* distributes the costs to the business units, enables the assessment of their financial performance and improves forecasting and decision making. It deals mostly with technical aspects, such as measuring usage and identifying cost allocation keys for the different IT services. Finally, *cost recovery* describes the process of setting prices and charging the business units for their usage of IT services. One objective is to set incentives, e.g., for cost-conscious behavior, and to regulate supply and demand of IT services. Hence, it has a strategic orientation.

The latter two steps are optional. However, if costs are allocated to business units the determination of an appropriate accounting base is crucial. From the IT perspective, it should integrate all relevant cost drivers. On the business side, it should enable an effective usage and cost control.

In the following, we focus on cost allocation and in particular on the determination of usage-based cost allocation keys in shared client/server environments where the resource consumption of the provided IT services is a significant cost driver. Despite the vast amount of literature in the domain, technical aspects of cost allocation and usage metering have mostly received little attention. So we draw, for an analysis of existing approaches, on a variety of different streams in the literature:

- Controlling and Cost Accounting [Rieb94; Horv06]
- Information Management [Krcm04] and IT Controlling [Karg99; KrBR00; GaMa05]
- Chargeback systems [Drur97] and IT Controlling in distributed environments [Aure97; HWWB99; Sche05]
- Organizational design of chargeback systems [McKa87; VeTB96; RoVB99]
- Costing approaches: Traditional costing [MaiJ96; Spit00], Process costing [Fuer94; Funk99], Activity-based costing [GNMA02]
- Best practice guidelines [OoOC02] and advices from analysts [Gart03]
- Vendors of chargeback software (see below)

We classified the identified concepts in the following according to the mapping of hardware (*HW*), applications (*App*) and business units (*BU*). We excluded approaches that are based on non-IT allocation keys (e.g., cost allocation per employee or per revenue share, etc.).

Direct cost allocation (HW → BU) Where a hardware resource is dedicated to a specific application or a business function, the costs of the assets can be directly allocated. This procedure is transparent and easy to implement. It is not IT-specific. However, the underlying assumption – that there is a single business owner of a resource – is mainly limited to computing and storage infrastructure. Other resources, like network or EAI components, are usually shared.

Measured usage (HW → BU) A widespread approach to allocating costs of a shared infrastructure is a proportional breakdown to the business units according to their resource consumption. Measured usage is commonplace for disk storage and telephony services. However, when it comes to client/server computing, a single request usually involves multiple heterogeneous components (e.g., web server, application server and database server). Measuring and allocating resource consumption directly to business units is associated with two fundamental problems.

First, from a user perspective, diverse technical metrics are difficult to comprehend and to control. Let's consider a business manager, receiving an account statement, based on the CPU times of multiple servers. As he cannot directly correlate the charges with his (business) activities, it is difficult for him to plan or manage resource consumption. In mainframe environments, with one single resource, this might have been possible, but the approach cannot be transferred to client/server infrastructures.

Second, due to performance and security reasons the original business context of a transaction (e.g., user, service) is mostly not available in the backend. User names are not further transmitted after a successful authorization and connection pools are used for database access. A SQL statement, for instance, does not contain any information about the actual user of the application who submitted the query. Modern database management systems support the allocation of resource consumption (CPU time, query runtimes, etc.) to the connection pools of the application servers, but they cannot reconstruct the original user.

Measured usage (HW → App → BU) For accounting and billing of shared IT services, a broad range of commercial tools is available (see for instance IBM [IBCo], USU [USU] Nicetec [Nice] and Econet [Econ]). Their strengths are the collection of accounting data by custom agents or by log files analysis, the consolidation of this information and the generation of reports for chargeback and management information. However, in client/server environments they are also facing the problems described in the previous section. Thus, they mostly provide a per-application view on resource consumption and use external data from the application or the organization to derive cost portions for the business units:

"Once you know the cost of an application as a whole, you can determine the cost of functional metrics produced by the application. For example, if you knew that "Payroll" cost $ 10,000 and that it produced 1,000 paychecks then the average cost of a paycheck is $ 10.00. Then, you could allocate the cost of payroll to the departments and business units based on the number of paychecks they received". (Excerpt from the CIMS System Description Manual [CIMS])

(Tiered) flat rate per application (App → BU) The application-oriented approach can be further simplified if one considers only the provision of an application in a data center as "IT product" and allocate costs by a flat rate to a single business owner. The actual resource consumption is not considered. The BMW Group uses such flat rates for applications on its J2EE infrastructure. Furthermore, Gartner recommends tiered flat rates [Gart03], based for instance on the required service levels. This approach is particularly easy to implement, as no explicit differentiation of resource costs and no metering is required. The accompanying lack of transparency may be accepted if the applications are of a similar nature (complexity, workload, etc.) and are predominantly used by a single business unit.

In our opinion, none of the approaches fulfills the above described requirements, such as customer-orientation (e.g., through non-technical accounting metrics) and the integration of relevant cost drivers – the resource consumption in particular.

3 Resource Profiles for Cost Accounting and Capacity Planning

3.1 Concept

In the following, we consider an IT system from the customers' (e.g., end-users, business units) perspective. For them the complexity of the underlying software and hardware is mostly not transparent. Instead, they perceive the system largely as black-box, accessible through (graphical) user interfaces and providing a number of business-related services. Categories of such services could be 'Execution of a business transaction' (e.g., 'process order', 'update stock', and 'add customer') or 'Access to an Information System' (e.g., 'retrieve order details', 'browse catalog', and 'check plant status'). An application may implement one or more of those services, but a service can also comprise multiple applica-

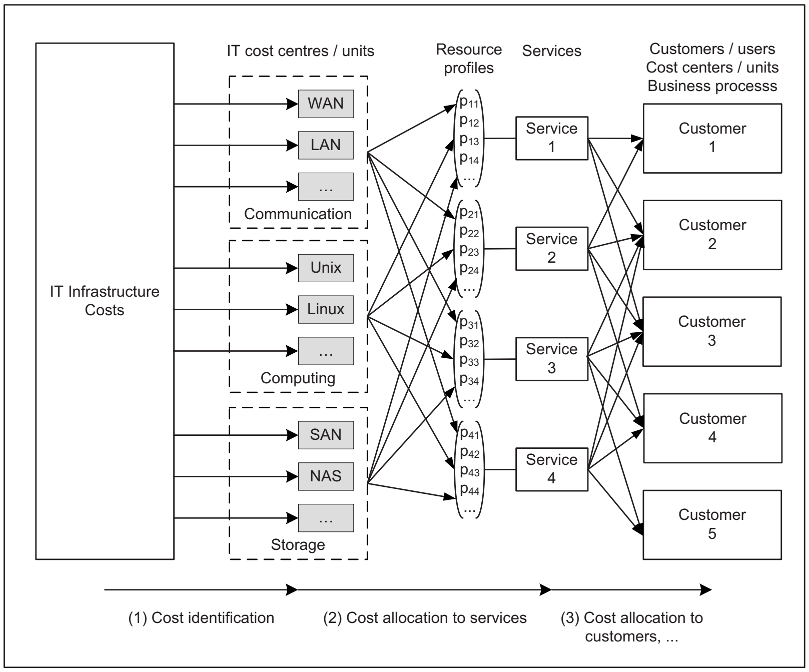


Figure 2 Cost allocation using resources profiles

tions. We assume that the invocation of such a service always results in similar resource consumption in the infrastructure. If estimates for the expected consumption at the different resources and the number of service invocations were known, this could constitute the basis for cost allocation keys. The elaborate process of measuring and consolidating log data from different components could be bypassed. Furthermore, these estimates would be valuable inputs for the alignment of business forecasting and IT capacity planning. The concept is not dependent on a specific costing methodology. It is applicable for traditional cost accounting (full and variable costing) as well as for process- or activity-based costing approaches. Generally, we propose a three-stage process (see figure 2). Budgeted costs for the provision of shared IT resources are apportioned among the services according to services' expected resource consumption and forecasts of the resources' total usage. Thus, for every service a cost portion is determined, which is allocated to the customers' accounts according to their number of service invocations.

In the following, we describe a methodology to estimate for a service i a so-called *resource profile*, $\mathbf{p}$, which is defined as a vector of n resource consumption values p_{ij}, with $j = 1 \ldots n$ (CPU time Web Server, CPU time Application Server, CPU time Database Server, transferred bytes, etc.). The expected values p_{ij} are estimated by the arithmetic mean of measurements during a load test. The resource profile multiplied by the number of service requests of a certain service customer (which can easily be traced) should result in an estimator for the resource consumption of this customer. The question is, whether the sample average of values in $\mathbf{p}$ is a good estimator for the true resource consumption of a user given varying workloads.

3.2 Requirements

It is unclear whether such estimates can be made with high enough accuracy in heterogeneous IT infrastructures:

- (1) The resource consumption of a service request may depend to a certain degree on the workload. A linear extrapolation of a single service request might therefore be biased.
- (2) Inaccuracies of the measurement and analysis instruments may introduce a systematic error, or also lead to high sample variance. In order to cause as little extra overhead as possible (e.g., no reimplementation of applications and no additional monitoring software installed), we intend to work with standard OS performance monitors. The question is, can measurements based on standard log file data be sufficient to derive accurate resource consumption estimates?
- (3) Assumptions about the expected user behavior (e.g., parameters passed, interactions with the system) during the setup of the load tests may not reflect reality.

Overall, a validation of $\mathbf{p}$ can be performed ex-post by comparing the real consumption with the estimated resource utilization. CPU is typically the scarce resource for database or application servers [KeEi06, pp. 607 et seqq.]. Thus, for (1) and (2) we use Queuing Network Models (see section 5) to predict resource utilization at different workloads using the estimated CPU times in $\mathbf{p}$ and compare those to the results of separate experiments with the same workload. If there is no significant difference between the prediction and the values observed in real load tests for different workload scenarios, these values should satisfy (1) and (2) and also be sufficiently accurate for cost allocation keys. Overall, these parameter estimates provide valuable inputs for analytical capacity planning and infrastructure sizing, which are not readily available in most organizations.

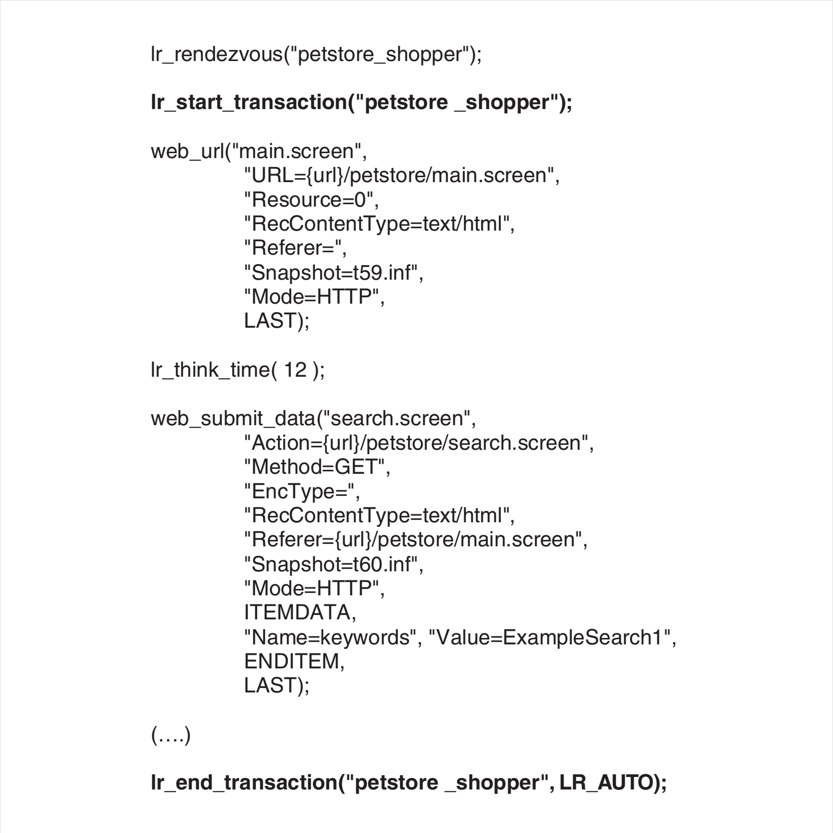
Highly interactive applications are more difficult and require adequate user behavior models. A variety of different tools [IdOb] and modeling approaches [MeAl00, pp. 41–64], particularly for web-based applications, exist. If historical usage data is available, this can be used to derive respective models. Even for very interactive applications, it is typically possible to get at least a good estimate of how often particular service requests are made. If services are modeled and metered at a very fine-granular level (e.g., single requests) it is not even necessary to make assumptions about user behavior.

4 Estimation of Resource Profiles

The overall requirements on the process of estimating resource profiles in heterogeneous environments are detailed in section 1. Against this background, existing profiling tools have two major drawbacks. First, their main focus is diagnosing performance problems, like memory leaks or CPU consumption of specific methods or transactions. They are not designed for measuring the consumption of coherent user activities. Second, the tools are technology-

or even vendor-specific, (e.g., Java/J2EE [JaPe], .NET [Schw], Intel [InCo], different ERP/CRM systems [SyCo]) and typically do not cover all necessary resources.

Load generators (e.g., Mercury LoadRunner [MeIC], SilkPerformer [BoSC], The Grinder [SoFo]), on the other hand, provide elaborate means of simulating different user behaviors in various environments. However, their main focus lies on response times experienced by their virtual users. Although most tools provide consoles for a remote monitoring of hardware and server performance, none of the products we evaluated calculates resource consumption values for coherent user activities. In the following, we describe a number of tools that we have developed to derive resource profiles for user-oriented services.



```
lr_rendezvous("petstore_shopper");

lr_start_transaction("petstore _shopper");

web_url("main.screen",
          "URL={url}/petstore/main.screen",
          "Resource=0",
          "RecContentType=text/html",
          "Referer=",
          "Snapshot=t59.inf",
          "Mode=HTTP",
          LAST);

lr_think_time( 12 );

web_submit_data("search.screen",
          "Action={url}/petstore/search.screen",
          "Method=GET",
          "EncType=",
          "RecContentType=text/html",
          "Referer={url}/petstore/main.screen",
          "Snapshot=t60.inf",
          "Mode=HTTP",
          ITEMDATA,
          "Name=keywords", "Value=ExampleSearch1",
          ENDITEM,
          LAST);

(….)

lr_end_transaction("petstore _shopper", LR_AUTO);
```

Figure 3 Excerpt from a LoadRunner script

4.1 The Service Profiler

The *Service Profiler* is based on a commercial load generator, the Mercury LoadRunner, and combines it with the standard performance monitoring tools of Unix, Linux (*sar*) and Microsoft Windows (*perfmon*). These operating system tools allow for a detailed monitoring. We chose the LoadRunner for the simulation of service invocations, as it supports around 30 protocols for different front- and backend interfaces, including HTTP, J2EE, .NET as well as protocols for major ERP/CRM systems. The user behavior is first recorded in a script (see figure 3 for an HTTP script example), which can then be replayed by the desired number of virtual users. The tool provides the possibility to parameterize the script after the recording, for instance to generate arbitrary input data for forms and to integrate user-defined transactions. In our experiments we used this feature to define the start- and endpoint services. In the example of figure 3 the script replays the behavior of an average shopper in the Java Pet Store (see section 4.2).

The whole shopping process is considered as a single service. In the script the service starts with the command *lr_start_transaction("petstore_shopper")* and ends with *lr_end_transaction("petstore_shopper", LR_AUTO)*. In the meantime the user browses on a predefined path through the store. Start- and end-time of the services are recorded in the result file of the load test. This allows for great flexibility in defining IT services. Depending on average user behavior and accounting requirements, one can define a single dialogue step as well as larger number of user interactions as a service. Overall, however, service definition requires careful planning (see previous section). In the experiments we installed the applications under consideration in an isolated test environment, as is usual for operational approval tests [OoOC02] prior to the roll-out of a new software release. By means of the load generator, we then simulated consecutive service invocations, while the performance monitors recorded the system's utilization in log files. After the load test, the Service Profiler parses the different log files and consolidates them, together with the results from the load generator, in a database. It then correlates start and end times of service invocations with the performance data and calculates the resource profiles. Finally, a custom reporting package visualizes the results. In order to determine the resource consumption of a single service request, we successively raised the number of parallel service invocations and determined the increase in resource consumption by means of a linear regression (i.e. the slope of the regression line).

4.2 The Java Pet Store Example

In our experiments we used CPU time, network and storage I/O as variables p_{ij} of our resource profile. We excluded disk space, as it is usually a priori allocated to a specific application or a database and elaborate accounting tools are available. Memory is often also considered as a scarce resource. However, the maximum amount of physical memory a server can allocate on a machine is typically determined at startup (e.g., by setting a range for virtual memory) and it is possible to take this value as the basis for cost allocation.

The experimental infrastructure was set up in a data center of the BMW Group (see table 1). We combined different operating systems (Linux, Windows and Unix) and servers (Apache HTTP, Bea Weblogic and Oracle Database). The tablespaces of the database are stored in a Storage Area Network (SAN), which is connected via Fibre Channel to the database server.

Within the context of the BMW Group we evaluated the approach with custom

Table 1 Infrastructure used in the experiments

	Load Generator	Web Server	Application Server	Database Server
Pet Store Application	petstore.usr Vuser script & load scenario	/petstore (static content)	petstore.ear	Oracle petstore tablespace
Software / Server Infrastructure	LoadRunner 8.0 JRE 1.5.0	Apache http 2.0.54	Bea Weblogic 8.1 JRE 1.3.1	Oracle 9.1
Operating Systems	Windows 2000 Advanced Server	Red Hat Linux Advanced Server 2.1 (Pensacola)	Windows 2000 Advanced Server	HP-UX 11.11
Number of CPUs	2	2	2	2
CPU Performance	1000 MHz	1400 MHz	1000 MHz	440 MHz
CPU Type	Intel x86 Pentium III Coppermine	Intel x86 Xeon MP	Intel x86 Pentium III Coppermine	PA 8500 CPU Module 2.3
CPU Architecture	CISC (32 bit)	CISC (32 bit)	CISC (32 bit)	RISC (64 bit)
Disks	3 * 73 GB (RAID 5)	3 * 72 GB (RAID 5)	3 * 18 GB (RAID 5)	2 * 36,4 GB (RAID 1) 13 GB LUN on SAN (HP XP128) via FC
Network	100 Mbps	100 Mbps / 1Gbit	100 Mbps	100 Mbps Fibre Channel
Memory	2 GB	2 GB	2 GB	2 GB
Server Type	HP DL360	IBM X360-03	HP DL360	HP N4000
Network Name	xxxxxx20	xxxxxx03d	xxxxxx15	xxxxxx01a

business applications. As example scenario in this paper we use the J2EE reference implementation Java Pet Store [SuMi]. We chose Pet Store over commercial benchmarks such as TPC-App [TPPCa] for three reasons. Firstly, it is readily available and the experiments are easy to repeat; secondly, it covers most J2EE technologies (EJBs, Servlets, JSPs, Web Services and XML, JMS, CMP, etc.), and, thirdly, the software architecture, with several interacting applications in the front- and back-end, is an appropriate representation of the structure of modern enterprise systems.

We have chosen a very interactive scenario, because it is typical for web applications. One possibility to cope with this interactivity would be the definition of very fine granular services (e.g., "view catalogue page", "add item to cart"). However, for this paper, we decided to consider the whole Pet Store as single service ('Access to an Information system', see section 3). Our intention was, first, to compare the resource consumption of different user profiles and, second, to verify that Queuing Network Theory is also applicable to services consisting of multiple user interactions with the system (see section 5.3). Thus, we defined the following user profiles: The 'curious_visitor' enters the store and visits two product descriptions. The 'first_time_shopper' visits five different product sites, submits his personal information and buys one product. The 'determined_shopper' buys four products all selected by the search engine. Finally, the 'power_shopper' extensively uses the online-catalogue and the search engine and buys ten different products. 'Power_shopper' and 'determined shopper' are already subscribed to the Pet Store. The input data for forms on the web sites (e.g., personal information or search strings) was generated arbitrarily. Before the load test and after each measurement cycle, the involved servers were rebooted and their system clocks synchronized with a time server in the local network. We started the profiling process for each user profile with 10 concurrent users and increased the number in steps of 10 until we reached 100 users. This process was repeated for each user profile.

Figure 4 was generated by our reporting tool. Each arrow indicates the start (black arrows) or stop (grey arrows) of a virtual user. At label (1), 20 concurrent 'power_shopper' start browsing through the Pet Store. To avoid peak loads, we built in arbitrary think times between two user activities. About two minutes later all users have finished their shopping tour (2). After an idle phase of 45 seconds (3), the procedure is restarted with 30 concurrent users. During the whole test, the performance monitors at the Web, Application and Database Servers record the system behavior in log files on their local disks. The upper diagram (4) shows the overall CPU utilization at the Application Server. Data about network and storage resources is analyzed in the same manner. After the experiment, the Service Profiler computed the consumption per resource and user profile. Therefore, it first summed up the measurements for each interval with constant number of users (e.g., the CPU times for 10 users, 20 users, and so forth) and then applied a linear regression. We consider the slope of the regression line as average consumption per additional user and as estimate for the expected resource consumption. The Pearson correlation coefficient ($r \in [-1; 1]$) and the resource profile $\mathbf{p}$ for the Pet Store example is shown in table 2. For comparison we included the resource profile of an 'edit_user' of a simple Servlet-

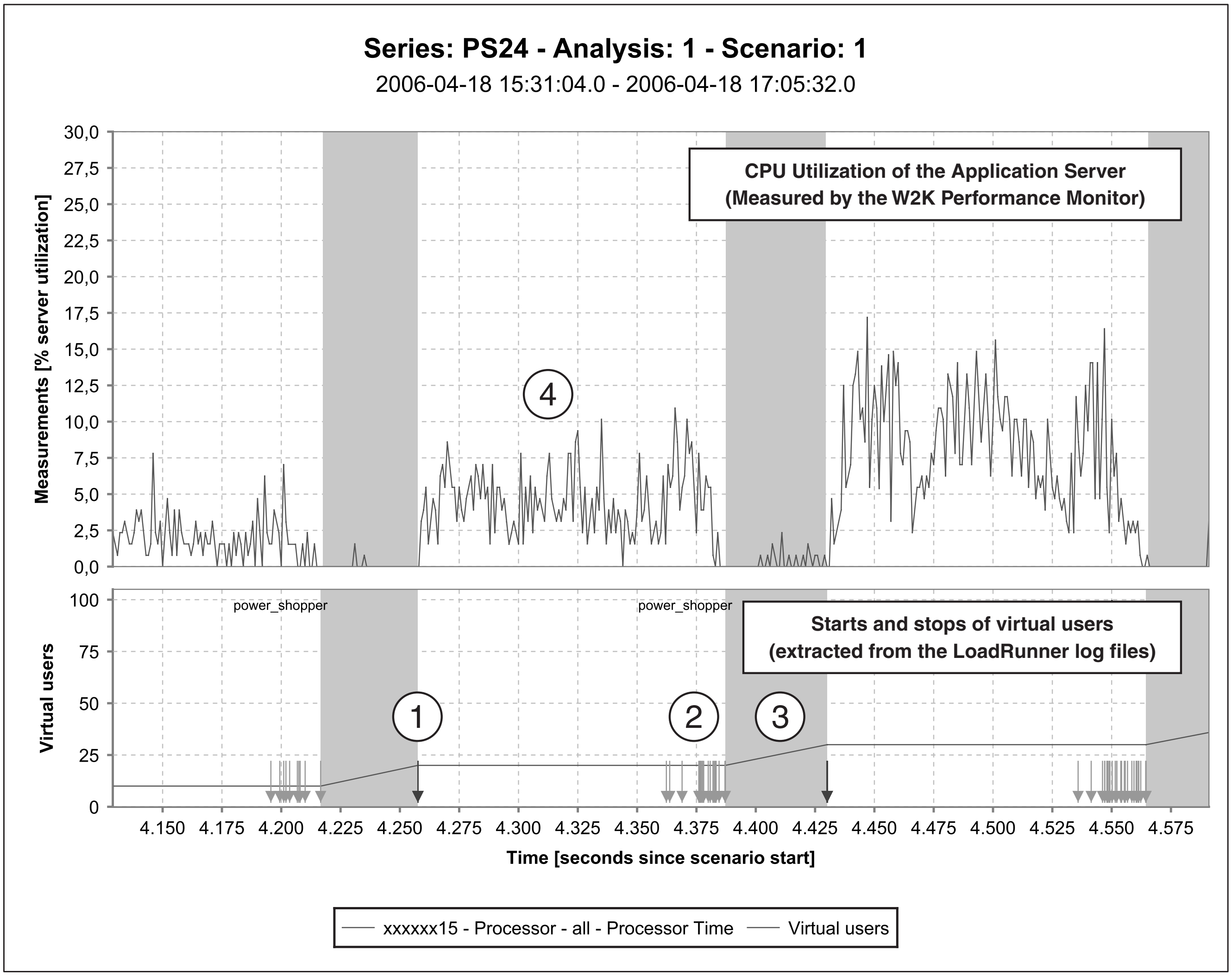


Figure 4 Determination of the resource profile for a Pet Store shopper

Table 2 Resource profiles for the Pet Store frontend application and an example application of the BMW Group

	curious_visitor		first_time_shopper		determined_shopper		power_ shopper		edit_user (BMW App)	
	p	r	p	r	p	r	p	r	p	r
Web Server CPU time [sec]	0,0064	0,9249	0,0130	0,9836	0,0076	0,9812	0,0122	0,9677	0,040	0,999
App. Server CPU time [sec]	0,0389	0,5530	0,1293	0,8183	0,0838	0,7577	0,2071	0,8251	0,830	1,000
DB Server CPU time [sec]	0,0008	0,1474	0,0200	0,9550	0,0089	0,9449	0,0251	0,8749	3,715	1,000
SAN I/O [blocks]	0	n/a	7,1230	0,7518	0	n/a	0	n/a	159,14	0,970
Network I/O [bytes]	174.521	1,0000	434.651	1,0000	250.183	1,0000	562.443	1,0000	398.283	1,000

based application used for project status tracking at the BMW Group.

The linear increase of the resource consumption is most evident for the network resource. In contrast, the transferred bytes of the Pet Store to the SAN seem not to be related with the number of concurrent users. The application is programmed to avoid "expensive" disk accesses as far as possible. Instead most data is kept in the memory of the servers. Accordingly, most SAN I/Os happen at the very beginning of the load test. Finally, concerning the CPU times, we can observe a mixed situation for the Pet Store. While the resource consumption for 'power_shopper' is approximately linear ($r > 0{,}8$) on every server, the other user profiles have varying correlation coefficients for the Application and the Database server. This can be explained by their different resource requirements, e.g., a 'curious_visitor' does not require database access. Overall, the 'first_time_shopper' has relatively high consumption values. This is caused by the functions for registering a new user. We conducted the Pet Store experiments with the standard database content as provided by SUN. However, in real-world scenarios it is important to use realistic database sizes. To compare the CPU times with measurements on other infrastructures, they must be normalized, e.g., by using standard performance benchmarks [SPECa].

These resource profiles provide an overview of the expected resource consumption for different types of user behavior. If historical usage data is available, an average resource profile can be derived. Otherwise, the determined bandwidth of the consumption values already allow for an approximate classification, which might be sufficient for accounting purposes.

5 Evaluation of CPU Service Times Using Queuing Network Theory

In order to use resource profiles for cost allocation keys, they need to be unbiased and independent of the workload of the system in question. Queuing Networks are an excellent tool to validate this assumption. If we are able to accurately predict resource utilization under different workloads, this means that our estimates for CPU time are not only unbiased in a statistical sense, but also independent of the system workload.

5.1 The QN Solver

Queuing Network Theory is a well-studied methodology for the mathematical analysis of systems with waiting lines and service stations. Today, Queuing Network Models are used in various domains, ranging from manufacturing system planning [TeKu98] to computer performance modeling [BoRi97; MeAD04; BoGM06].

A queue consists of one or more service stations with a joint waiting room. Jobs arrive at the queue with an arrival rate λ and are served in an average time S. If the service stations are all occupied, jobs have to line up. The so-called Kendall notation [Kend53] is typically used to classify different types of queues: A / B / C (where A stands for the distribution of interarrival times of customers, B for distribution of service times, C for number of service stations). A and B usually take the following distributions types: M (Exponential / Markovian Distribution) or G (General / Arbitrary Distribution). A Queuing Network Model consists of a number of interconnected queues. Depending on their characteristics and of the workload (number/type of jobs), several exact and approximate solution techniques exist. A solution consists of response times for jobs, the lengths of waiting lines and the utilization of queues. Parameters such as the service time of jobs in a computer system are often not readily available, which is one reason why the technique is rarely used for the capacity planning of IT systems.

For the computation of Queuing Network Models, we implemented a software component, the *QN Solver*. It relies on the same data model as the Service Profiler presented above and the QN Verifier (see next section). A number of alternative tools are available as open source software [Hlyn].

The QN Solver implements algorithms for Queuing Networks with unbounded number of customers ("open QNs") and M/M/n queues, as well as algorithms for Queuing Networks with limited number of customers ("closed QNs") and queues belonging to the BCMP family [BCMP75]:

- M/M/m – First come first serve (FCFS) (different types of jobs must have equal service times.)
- M/G/1 – Processor sharing (PS)
- M/G/∞ – Infinite server (IS)
- M/G/1 – Last come first serve with Preemptive Resume

If these properties were fulfilled by all queues in the network, efficient solutions algorithms exist. For the computation of those closed QNs we apply either the exact Mean Value Analysis (MVA) [ReLa80] or, for networks with a large numbers of users and multiple job classes, the Self Correcting Approximation Technique (SCAT) [NeCh81]. We refer the interested reader to [MeAD04] and [BoRi97; BoGM06] for a more detailed description of these algorithms.

5.2 The QN Verifier

For the validation of the Queuing Network computations, we first developed an appropriate load test setup. Again (see section 4.1), we use Mercury LoadRunner for the simulation of users and the performance monitors of the different operating systems to record the system behavior. Instead of simultaneous starts and stops, we now put the users in endless loops. After a certain period of time (e.g., 5 minutes), we add additional users, until the first component reaches its bottleneck.

After the load test the *QN Verifier* stores the results in a database. The utilization of servers in our load tests are then compared with the predictions of the QN Solver. We calculate the absolute difference between the observed and the predicted server utilization to measure predictive accuracy of our Queuing Network Models. The interplay between the different software components is depicted in figure 5.

5.3 The Java Pet Store Example (Continued)

In the load test setup, presented in the previous section, the number of users remains constant during certain time intervals. Therefore, we model the infrastructure as closed QN and compute for each interval separate results. Furthermore, for the Pet Store example, we made the following modeling assumptions:

- We modeled solely the processors, since the disk times were negligible during the profiling. Hard disks would be typically modeled as M/M/n – FCFS queues [BoRi97, p. 73].
- A processor can be modeled as M/G/1 – PS queue [BoRi97, p. 73] In our test infrastructure we use dual-processor machines, which would be modeled accordingly as M/G/2 – PS queue. Unfortunately, an efficient solution algorithm for a network containing queues with multiple service stations exists only if the queues are of M/M/m – FCFS type (see previous section). Closed Queuing Net-

works are quite robust towards service time distributions [BoRi97, p. 172]. Thus, we approximated the non-exponential service time distribution by an exponential distribution. In this approach only one job class is permitted.

- The think time of the users is represented by an M/G/∞ queue. The infinite number of service stations indicates that independent from the actual load no queing effects occur. This reflects reality as the think time of a virtual user is not dependent on the number of concurrently active users.

The structure of the resulting Queuing Network Model is depicted in figure 6. We selected the 'power_shopper' (see section 4.2.) for the validation of the resource profile. The mean service times were taken from the resource profile in table 2. As these are non-normalized measurements on a dual-processor machine, we had to double them for the Queuing Network model. The load intensity is specified by the total think time of a user. In the example we assume an arbitrary chosen think time of 38s.

For the solution of the Queuing Network Model we applied the exact Mean Value Algorithm. We calculated the average server utilizations in steps of 10 from 10 to 220 users. At this level the predicted utilization of the Application Server converged towards 100 %. In a second step, we set up the load test according to the description in section 5.2. Like in the QN calculations, the load test was started with ten users in an endless loop. Every five minutes we added ten more users until we reached the predicted bottleneck of 220 concurrent users. The QN Verifier then analyzed the data and computed for the intervals with constant numbers of users the average utilization of the involved servers. As the sudden and simultaneous start of 10 new users may lead to a non-steady transient behavior in the system, we excluded the first minute of each interval from the analysis. After one minute with constant load, we assume the system has arrived in a steady state. In table 3 the arithmetic means of the measured utilization during time intervals with constant number of users are compared to the values predicted by the Queuing Network Model.

Overall, the predictive accuracy, as measured by the mean absolute deviation for the web server, the application server, and the database server is 0,004 (0,4 %), 0,03 (3 %) and 0,01 (1 %) respectively. Despite the heterogeneity of the infrastructure and the different workloads during the load test, the accuracy of the estimates is surprisingly exact. The quality of the results is comparable to directly related approaches. Kounev and Buchman [KoBu03; BoGM06 pp. 703 et seqq.] describe a J2EE capacity planning experiment, conducted with Bea Weblogic Application Server, Oracle Database and the SPEC JAppServer 2002 benchmark [SPECb] as example application. They analyzed the system under three different workloads. Overall, their predictive accuracy is between 2 % and 4 % for both kinds of servers.

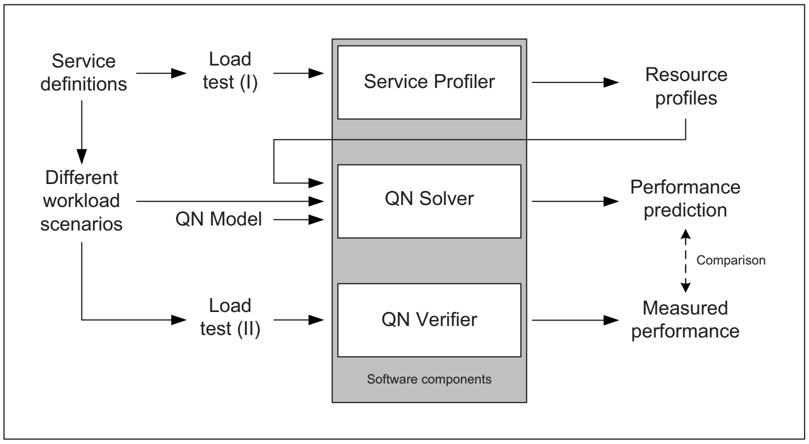


Figure 5 The interplay between the software components

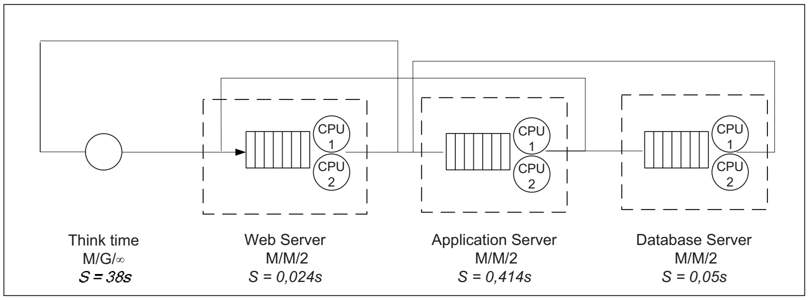


Figure 6 QN-Model of the infrastructure used in the experiments

6 Related Work

The need for service-level resource profiles is well motivated in proposals by Funke [Funk99] and Scheeg [Sche05] in the areas of Cost Accounting and IT Controlling. Concerning the determination of resource profiles our concept is related to Nagaprabhanjan and Apte, who recently presented a tool [NaAp05] for automated profiling of distributed transactions. They also combine load generation with performance monitors for the determination of resource consumption. Their focus lies on the determination of input parameters for performance analysis and capacity planning. They use a custom load generator, which requires measurement agents installed on the different servers. The use of custom agents limits the flexibility of the implementation (currently to Linux servers and Web applications [NaAp05]). In contrast, we use a commercial off-the-shelf load generator and require no additional software installations on the servers.

Table 3 Comparison of load test measurements with QN predictions

	Web Server			Application Server			Database Server		
# users	u [%]	u_{qn} [%]	$u - u_{qn}$	u [%]	u_{qn} [%]	$u - u_{qn}$	u [%]	u_{qn} [%]	$u - u_{qn}$
10	0,65%	0,32%	0,003	7,55%	5,38%	0,022	2,59%	0,65%	0,019
20	1,01%	0,63%	0,004	10,36%	10,76%	–0,004	1,72%	1,30%	0,004
30	1,31%	0,95%	0,004	20,14%	16,14%	0,040	2,17%	1,95%	0,002
40	1,71%	1,27%	0,004	18,37%	21,51%	–0,031	2,45%	2,60%	–0,002
50	2,06%	1,59%	0,005	33,40%	26,88%	0,065	3,45%	3,25%	0,002
60	2,41%	1,90%	0,005	29,78%	32,24%	–0,025	3,75%	3,90%	–0,002
70	2,70%	2,22%	0,005	39,68%	37,60%	0,021	4,42%	4,55%	–0,001
80	3,06%	2,53%	0,005	46,22%	42,94%	0,033	4,76%	5,20%	–0,004
90	3,40%	2,85%	0,005	47,39%	48,27%	–0,009	5,07%	5,84%	–0,008
100	3,76%	3,16%	0,006	64,22%	53,58%	0,106	5,58%	6,48%	–0,009
110	4,10%	3,47%	0,006	58,96%	58,86%	0,001	6,52%	7,12%	–0,006
120	4,38%	3,78%	0,006	75,58%	64,11%	0,115	6,74%	7,76%	–0,010
130	4,73%	4,09%	0,006	74,01%	69,31%	0,047	6,99%	8,39%	–0,014
140	5,05%	4,39%	0,007	83,23%	74,43%	0,088	7,78%	9,01%	–0,012
150	5,29%	4,69%	0,006	88,62%	79,43%	0,092	8,10%	9,61%	–0,015
160	5,56%	4,97%	0,006	92,66%	84,24%	0,084	8,30%	10,20%	–0,019
170	5,71%	5,24%	0,005	96,35%	88,76%	0,076	8,85%	10,74%	–0,019
180	5,79%	5,47%	0,003	97,12%	92,80%	0,043	8,79%	11,23%	–0,024
190	5,83%	5,67%	0,002	97,39%	96,07%	0,013	8,93%	11,63%	–0,027
200	5,63%	5,80%	–0,002	94,95%	98,32%	–0,034	8,51%	11,90%	–0,034
210	5,94%	5,87%	0,001	97,44%	99,48%	–0,020	9,00%	12,04%	–0,030
220	5,55%	5,89%	–0,003	94,25%	99,89%	–0,056	8,32%	12,09%	–0,038

7 Conclusion

The many technical advances in corporate IT have led to short planning cycles and enormous price decreases in the past twenty years. Capacity and asset management are more difficult in an environment of fast technical change and have long been understudied. IT outsourcing has now achieved a level, where professional IT service providers provide application-level services to increasing numbers of customers, more and more based on shared infrastructures. Therefore, capacity and asset management are gaining increasing importance. In addition, "utility" and "on-demand" computing trends will further stimulate the demand for usage-based costing.

Technical difficulties and the complexity of modern IT infrastructures often lead to simple but biased cost allocation keys. In this paper we have proposed a method for the calculation of cost allocation keys based on estimated resource consumption of IT services. The approach satisfies a number of criteria that we consider essential for successful application in a professional IT service management organization. First, the estimators for the resource consumption of single service requests are of a high quality and can be used for capacity planning, as well as for accounting purposes. Second, the estimation process does not depend on certain hardware and software platforms. It can be broadly applied to different IT systems within an organization. Third, the process can be performed at low cost during standard approval tests.

We identified two major organizational success factors, which influence the practicability of the approach. On the one hand, the cost for integration into existing accounting and software testing processes should be minimal. On the other hand, business units and IT must define appropriate business services that can then be readily measured and priced. This constitutes a major task for cost accounting in general, not only for usage-based approaches [RoVB99].

Notes

[1] Exact figures: Full-time IT staff 28 %, Computer hardware 21 %, Software 18 %, IT Services 19 %, and Networking and communications hardware 14 % (Base: 270 IT executives at European enterprises were asked for their budget composition in 2006, September-December 2005).

References

[Aure97] *Aurenz, Heiko:* Controlling verteilter Informationssysteme: Client/Server-Architekturen. Peter Lang, Frankfurt am Main 1997.

[BCMP75] *Baskett, Forest; Chandy, K. Mani; Muntz, Richard R.; Palacios, Fernando G.:* Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. In: Journal of the ACM 22 (1975) 2, pp. 248–260.

[Bert01] *Bertleff, Claudia:* Einführung einer IT-Leistungsverrechnung zur Unterstützung des strategischen IT-Controllings. In: *Heilmann, H. (ed.):* Strategisches IT-Controlling. dpunkt.Verlag, Heidelberg 2001, pp. 57–66.

[BoGM06] *Bolch, Gunter; Greiner, Stefan; Meer, Hermann de:* Queueing Networks and Markov Chains. 2nd edition, Wiley-Interscience, Hoboken, New Jersey 2006.

[BoRi97] *Bolch, Gunter; Riedel, Helmut:* Leistungsbewertung von Rechensystemen mittels analytischer Warteschlangenmodelle. Teubner, Stuttgart 1997.

[BoSC] Borland Software Corporation: SilkPerformer. http://www.borland.com/us/products/silk/silkperformer/index.html, Last accessed: 2006-11-18.

[CIMS] CIMS Lab, Inc.: System Description Manual. http://www.cimslab.com, Last accessed: 2006-06-01.

[Drur97] *Drury, Donald H.:* Chargeback systems in client/server environments. In: Information & Management 32 (1997) 4, pp. 177–186.

[Econ] Econet AG: cMatrix DataXRay. http://www.econet.de/product/dataxray, Last accessed: 2006-07-13.

[Forr06] Forrester Research, Inc.: Global IT Budget Composition: 2006. http://www.forrester.com/Research/Document/Excerpt/0,7211,39632,00.html, Last accessed: 2006-09-12.

[Fuer94] *Fürer, Patrick J.:* Prozesse und EDV-Kostenverrechnung. Die prozessbasierte Verrechnungskonzeption für Bankrechenzentren. Paul Haupt, Bern 1994.

[Funk99] *Funke, Harald:* Kosten- und Leistungsrechnung in der EDV. Stand und Entwurf einer prozeßorientierten DV-Kostenverrechnung. Kassel University Press, Kassel 1999.

[GaJK05] *Gadatsch, Andreas; Juszczak, Jens; Kütz, Martin:* Ergebnisse der Umfrage zum Stand des IT-Controlling im deutschsprachigen Raum. Bd. 12, Fachhochschule Bonn-Rhein-Sieg. Fachbereich Wirtschaft Sankt Augustin, Sankt Augustin 2005.

[GaMa05] *Gadatsch, Andreas; Mayer, Elmar:* Masterkurs IT-Controlling: Grundlagen und Strategischer Stellenwert – IT-Kosten- und Leistungsrechnung in der Praxis. Vieweg, Wiesbaden 2005.

[Gart03] Gartner, Inc.: Chargeback: How Far Should You Go? (Executive Summary). http://www.gartner.com/DisplayDocument?ref=g_search&id=397166, Last accessed: 2006-08-06.

[GNMA02] *Gerlach, James; Neumann, Bruce; Moldauer, Edwin; Argo, Martha; Frisby, Daniel:* Determining the cost of IT services. In: Communications of the ACM 40 (2002) 9, pp. 61–67.

[Hlyn] *Hlynka, Myron:* List of Queueing Theory Software. http://www2.uwindsor.ca/~hlynka/qsoft.html, Last accessed: 2006-09-26.

[Horv06] *Horváth, Péter:* Controlling. 10th edition, Vahlen, München 2006.

[HWWB99] *Hübner, Dirk G; Waschbüsch, Christoph; Weinhardt, Christof; Bruhns, Peter; Koerner, Markus:* Prozessorientiertes IT-Kostenmanagement in Banken. State-of-the-art, Trends, Strategien. Fachverlag Moderne Wirtschaft, Frankfurt am Main 1999.

[IBCo] IBM Corporation: CIMS Chargeback System (now: Tivoli Usage and Accounting Manager). http://www.cimslab.com, Last accessed: 2006-07-13.

[IdOb] Ideal Observer: Einkaufsführer Web Analytics. http://www.idealobserver.de, Last accessed: 2006-09-04.

[InCo] Intel Corporation: Intel VTune Performance Analyzer. http://www.intel.com/cd/software/products/asmo-na/eng/vtune/239144.htm, Last accessed: 2006-11-18.

[JaPe] JavaPerformanceTuning.com: Tool reports. http://www.javaperformancetuning.com/tools, Last accessed: 2006-07-13.

[Karg99] *Kargl, Herbert:* DV-Controlling. 4th edition, Oldenbourg, München 1999.

[KeEi06] *Kemper, Alfons; Eickler, André:* Datenbanksysteme. 6th edition, Oldenbourg, München 2006.

[Kend53] *Kendall, David G.:* Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain. In: Annals of Mathematical Statistics 24 (1953), pp. 338–354.

[KoBu03] *Kounev, Samuel; Buchmann, Alejandro:* Performance Modeling and Evaluation of Large-Scale J2EE Applications. Proceedings of the 29th International Conference of the Computer Measurement Group (CMG) on Resource Management and Performance Evaluation of Enterprise Computing Systems. Dallas, USA 2003.

[KrBR00] *Krcmar, Helmut; Buresch, Alexander; Reb, Michael:* IV-Controlling auf dem Prüfstand. Gabler, Wiesbaden 2000.

[Krcm04] *Krcmar, Helmut:* Informationsmanagement. 4th edition, Springer, Berlin 2004.

[MaiJ96] *Mai, Jan:* Konzeption einer controllinggerechten Kosten- und Leistungsrechnung für Rechenzentren. Peter Lang, Frankfurt am Main 1996.

[McKa87] *McKinnon, William P.; Kallman, Ernest A.:* Mapping Chargeback Systems to Organizational Environments. In: MIS Quarterly 11 (1987) 1, pp. 5–20.

[MeAD04] *Menascé, Daniel A.; Almeida, Virgilio A. F.; Dowdy, Larry W.:* Performance by Design: Computer Capacity Planning by Example. Prentice Hall, Upper Saddle River, New Jersey 2004.

[MeAl00] *Menascé, Daniel A.; Almeida, Virgilio A. F.:* Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning. Prentice Hall, Upper Saddle River, New Jersey 2000.

[MeIC] Mercury Interactive Corporation: Mercury LoadRunner. http://www.mercury.com/us/products/performance-center/loadrunner, Last accessed: 2006-07-13.

[NaAp05] *Nagaprabhanjan, Bellari; Apte, Varsha:* A Tool for Automated Resource Consumption Profiling of Distributed Transactions. In: *Chakraborty, G. (ed.):* Proceedings of the Second International Conference on Distributed Computing and Internet Technology. Bhubaneshwar, India 2005, pp. 154–165.

[NeCh81] *Neuse, D.; Chandy, K.*: SCAT: A heuristic algorithm for queueing network models of computing systems. In: Proceedings of the ACM SIGMETRICS conference on Measurement and modeling of computer systems. Las Vegas 1981, pp. 59–79.

[Nice] Nicetec GmbH: netinsight. http://www.nicetec.de, Last accessed: 2006-07-13.

[OoOC01] Office of Government Commerce: Service Delivery. Stationery Office Books, London 2001.

Abstract

Cost Accounting for Shared IT Infrastructures – Estimating Resource Utilization in Distributed IT Architectures

IT infrastructure, such as servers and networking equipment, accounts for a large proportion of the IT costs in modern organizations. Typically, this IT infrastructure is shared among multiple applications and customers. Cost allocation of shared IT infrastructure is difficult and regularly based on biased cost allocation keys, which often causes free-rider problems. Measuring usage is technically difficult and incurs high costs. In this paper we propose a method to derive adequate estimators for the resource consumption of a customer-oriented service. These so-called resource profiles can then provide a basis for cost allocation keys. The estimators are derived from a series of load tests, as they are typically done before an application is launched in larger organizations. Such profiles need to be unbiased and precise even in cases of varying workloads and in rather heterogeneous environments. We describe the results of a set of experiments in an infrastructure provided by the BMW Group, and illustrate how the estimation can be integrated into existing IT service management processes. In our experiments we use Queuing Networks to validate the estimated resource profiles under different workloads.

Keywords: IT Infrastructure Cost Accounting, Usage-based Cost Allocation, Capacity Planning, Queuing Network Theory

[OoOC02] Office of Government Commerce: ICT Infrastructure Management. Stationery Office Books, London 2002.

[ReLa80] *Reiser, Martin; Lavenberg, Stephen S.:* Mean-Value Analysis of Closed Multichain Queuing Networks. In: Journal of the ACM 27 (1980) 2, pp. 313–322.

[Rieb94] *Riebel, Paul:* Einzelkosten- und Deckungsbeitragsrechnung. Gabler, Wiesbaden 1994.

[RoVB99] *Ross, Jeanne W.; Vitale, Michael R.; Beath, Cynthia Mathis:* The untapped potential of IT chargeback. In: MIS Quarterly 23 (1999) 2, pp. 215–237.

[Sche05] *Scheeg, Jochen Michael:* Integrierte IT-Kostentabellen als Instrument für eine effiziente IT-Leistungserbringung im Informationsmanagement: Konzeption und praktische Umsetzung. Difo-Druck, Bamberg 2005.

[Schw] *Schwichtenberg, Holger:* Tools and Software Components for the .NET Framework. http://www.dotnetframework.de/dotnet/tools.aspx, Last accessed: 2006-07-13.

[SoFo] SourceForge.net: The Grinder: A Java Load Testing Framework. http://grinder.sourceforge.net, Last accessed: 2006-07-13.

[SPECa] Standard Performance Evaluation Corporation: CPU2000 (CPU Benchmark). http://www.spec.org/cpu2000, Last accessed: 2006-08-31.

[SPECb] Standard Performance Evaluation Corporation: SPECjAppServer2002 (Java Application Server Benchmark). http://www.spec.org/jAppServer2002, Last accessed: 2006-12-14.

[Spit00] *Spitta, Thorsten:* Kostenrechnerische Grundlagen für das IV-Controlling. In: Kostenrechnungspraxis 44 (2000) 5, pp. 279–288.

[SuMi] Sun Microsystems, Inc.: Java Pet Store Sample Application. http://java.sun.com/reference/blueprints, Last accessed: 2006-07-13.

[SyCo] Symantec Corporation: Application Performance Management. http://www.symantec.com/Products/enterprise?c=prodcat&refId=1021, Last accessed: 2006-07-13.

[Sysk02] Syskoplan AG: Optimierung der Total Cost of Ownership in IT-Abteilungen scheitert an der fehlenden Leistungsverrechnung. http://www.syskoplan.de/content/pressemitteilungen/sysko_pres_260802.pdf, Last accessed: 2006-11-30.

[TeKu98] *Tempelmeier, Horst; Kuhn, Heinrich:* Flexible Fertigungssysteme. Entscheidungsunterstützung für Konfiguration und Betrieb. Springer, Berlin 1998.

[TPPCa] Transaction Processing Performance Council: TPC-App (Application Server and web services benchmark). http://www.tpc.org/tpc_app, Last accessed: 2006-06-26.

[UCCL] University of Cambridge Computer Laboratory: The Xen virtual machine monitor. http://www.cl.cam.ac.uk/Research/SRG/netos/xen, Last accessed: 2006-08-29.

[USU] USU AG: Costing/Charging Manager. http://www.usu.de/it_management_solutions/finance_management/costing_charging_manager.html, Last accessed: 2006-07-13.

[VeTB96] *Verner, June M.; Toraskar, Kranti; Brown, R.:* Information systems chargeout: a review of current approaches and future challenges. In: Journal of Information Technology 11 (1996) 2, pp. 101–117.

[Vmwa] VMware, Inc.: Virtualization software. http://www.vmware.com, Last accessed: 2006-08-29.