Comparing the Accuracy of Resource Demand Measurement and Estimation Techniques

Felix Willnecker[1], Markus Dlugi[1], Andreas Brunnert[1], Simon Spinner[2], Samuel Kounev[2], Wolfgang Gottesheim[3], and Helmut Krcmar[4]

[1] fortiss GmbH, Guerickestr. 25, 80805 München, Germany
`{willnecker,dlugi,brunnert}@fortiss.org`
[2] Universität Würzburg, Am Hubland, 97074 Würzburg, Germany
`{simon.spinner,samuel.kounev}@uni-wuerzburg.de`
[3] Dynatrace Austria GmbH, Freistädter Str. 13, 4040 Linz, Austria
`wolfgang.gottesheim@dynatrace.com`
[4] Technische Universität München, Boltzmannstr. 3, 85748 Garching, Germany
`krcmar@in.tum.de`

Abstract. Resource demands are a core aspect of performance models. They describe how an operation utilizes a resource and therefore influence the systems performance metrics: response time, resource utilization and throughput. Such demands can be determined by two extraction classes: direct measurement or demand estimation. Selecting the best suited technique depends on available tools, acceptable measurement overhead and the level of granularity necessary for the performance model. This work compares two direct measurement techniques and an adaptive estimation technique based on multiple statistical approaches to evaluate strengths and weaknesses of each technique. We conduct a series of experiments using the SPECjEnterprise2010 industry benchmark and an automatic performance model generator for architecture-level performance models based on the Palladio Component Model. To compare the techniques we conduct two experiments with different levels of granularity on a standalone system, followed by one experiment using a distributed SPECjEnterprise2010 deployment combining both extraction classes for generating a full-stack performance model.

Keywords: performance model generation, resource demand measurements, resource demand estimations

1 Introduction

Performance models can be used to predict the performance of application systems. Resource demands are an important parameter of such performance models. They describe how an operation utilizes the available resources. A busy resource increases the time an operation needs to execute, therefore increasing the

Acknowledgment: This work has been supported by the Research Group of the Standard Performance Evaluation Corporation (SPEC).

response time of the operation and ultimately the time for the user accessing the system. When performance models are applied for capacity management, such information is essential as the available hardware must be sized according to the demand of the operations for a certain workload. Demands can be extracted from different sources. Expert guesses are used, especially when no running application artifact is available, to forecast the application's performance behavior. If running artifacts are available (e.g., in a test environment), measurement and estimation techniques can be applied. This work compares two direct measurement techniques and an adaptive estimation technique based on multiple statistical approaches and compares strengths and weaknesses of each technique.

Manually creating performance models often outweighs their benefits [6]. Therefore, automatic performance model generator (PMG) frameworks for running applications have been introduced in the scientific community [3,6]. Such PMGs create performance models, which include the software architecture, control flow and the resource demand of the application. These PMGs use either direct measurements by instrumenting the operations that are executed or resource demand estimations calculated from coarse-grained measurement data like total resource utilization and response time per transaction invocation.

Applying direct measurements requires to alter the installation of the system that is instrumented by applying an agent that intercepts invocations. This allows for extracting the software architecture and control flow, but causes overhead on the system running for every instrumented operation that is invoked [5]. Furthermore, such measurements require that for each instrumented technology and resource type, a dedicated measurement approach must be available. A number of industry solutions for direct measurements are already available and have been integrated into such a PMG previously [17].

As an alternative to direct measurements, resource demand estimation techniques can approximate the demand of a resource from coarse-grained monitoring data like Central Processing Unit (CPU) utilization of a system and response time of a transaction. Such data can be collected for a wide range of systems and technologies and requires no in-depth measurement of the application's technology stack. This coarse-grained monitoring data causes less overhead, produces less data to collect, and to process. However extracting the control flow of an application is not possible with such an approach.

The Library for Resource Demand Estimation (LibReDE)[6] provides different resource demand estimation approaches [15]. In order to do the estimations, LibReDE requires information about the resource utilization as well as about the response times of an operation or transaction during the same time frame. This work integrates LibReDE with the PMG introduced by Brunnert et al. [6] in order to be able to generate models based on direct resource demand measurements or estimations. This integration allows to compare the direct measurement and estimation approaches and to determine strengths and weaknesses for ex-

[6] http://se.informatik.uni-wuerzburg.de/tools/librede/

tracting resource demands using the SPECjEnterprise2010[7] industry benchmark as representative enterprise application for the evaluation.

We compare these two extraction classes for resource demands in a series of experiments evaluating the accuracy of automatically generated performance models in terms of CPU utilization and response times. Therefore, the main contributions of this work are as follows:

(i) An integration of resource demand estimation in a PMG.
(ii) A comparison of the accuracy of two direct measurement techniques with the most common resource demand estimation approaches used in practice.
(iii) An evaluation of an integrated PMG, utilizing the benefits of direct measurement and estimation techniques.

This work begins with an introduction to the performance model generation workflow followed by introducing measurement technologies. We continue with an introduction to LibReDE and the approaches used to estimate resource demands including the selection of the most accurate estimation approach for meaningful resource demands. The experiment for comparing all three approaches is described and evaluated, followed by a hybrid setup where a combination of direct measurements and resource demand estimations is used. The work closes with related work, followed by the conclusion and future work section.

2 Extracting Resource Demands

In order to support resource demand measurement and estimation approaches, we extend the previously introduced Performance Management Work (PMW)-Tools' automatic PMG with LibReDE [6,15]. Generating a performance model is divided into three separate steps depicted in Figure 1. First monitoring data is gathered. This monitoring data is, in a second step, aggregated per operation and stored in a monitoring database (DB). The last step is the actual model generation, which uses the aggregated data and generates an architecture-level performance model based on the Palladio component Model (PCM) [1].

The PMG supports data from different data sources:

(i) PMW-Tools monitoring, a monitoring solution for Java Enterprise Edition (EE) applications to measure CPU, memory, and network demands and response times of Java EE components and its operations [4,6].
(ii) Dynatrace[8] Application Monitoring (AM), an industry monitoring solution for Java, .NET, PHP and other technologies [17].
(iii) System Activity Reporter (SAR), an Unix/Linux based tool to display various system loads like CPU utilization.

[7] SPECjEnterprise is a trademark of the SStandard Performance Evaluation Corp. (SPEC). The SPECjEnterprise2010 results or findings in this publication have not been reviewed or accepted by SPEC, therefore no comparison nor performance inference can be made against any published SPEC result. The official web site for SPECjEnterprise2010 is located at http://www.spec.org/osg/Enterprise2010.

[8] http://www.dynatrace.com

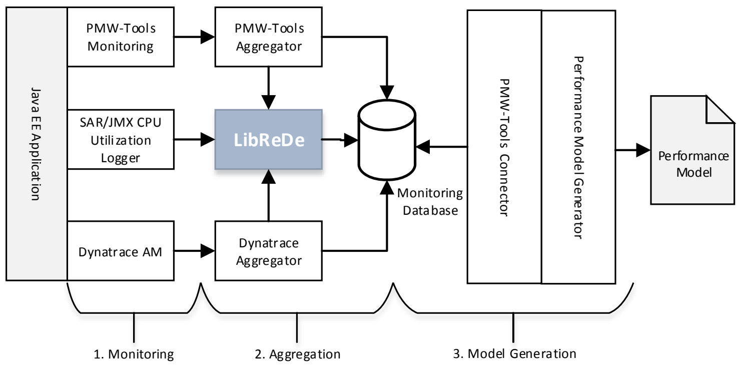

Fig. 1. Performance model generator framework (adapted from [5,17])

(iv) Java Management Extensions (JMX) Logger, a command line tool that reads CPU utilization values from Java Virtual Machines (JVMs) using the JMX interface.

The first two data sources are able to collect direct measurement data, but also response times for estimation techniques. The demand estimation is realized using LibReDE [15]. This library uses response times of an operation or transaction and utilization of a resource, collected by one of the last two data sources, to estimate the resource demands of an operation [15].

2.1 Performance Management Work - Tools Monitoring

PMW-Tools monitoring provides a Servlet Filter, an Enterprise JavaBean (EJB) Interceptor, a SOAP-Handler and a Java Database Connectivity (JDBC)-Wrapper for Java EE applications [4,6]. The aforementioned technologies allow to collect CPU time, heap allocation and network demand on the level of single operation invocations [4,5,6]. Furthermore, the PMW-Tools monitoring allows to collect information about the transaction control flow and about an application architecture on the level of components and their operations. All public operations within the instrumented system are extracted and combined to one transaction. The PMW-Tools monitoring agent is able to measure the response time of an operation. The start and end time of each operation invocation is measured. Subinvocations are removed from this time interval, so the actual response time of one operation invocation is calculated.

2.2 Dynatrace Application Monitoring

The Dynatrace AM solution allows for measurements on different levels of granularity. This ranges from measuring the response time on the system entry point level, through fine-grained measurements per operation invocation. Dynatrace AM uses, depending on the host system, various timers that measure the CPU utilization in different time intervals [7]. It furthermore traces a transaction throughout the instrumented system and can therefore determine the control flow as the PMW-Tools monitoring does [17]. The Representational State Transfer (REST) interface of this solution provides, among other metrics, the ability to access CPU time and response times of the instrumented operations. Thus, this approach, as well as the PMW-Tools monitoring approach can be used for direct measurements and estimation techniques.

2.3 Library for Resource Demand Estimation

Demand estimation approaches While the monitoring tools described in subsection 2.1 and subsection 2.2 are able to directly measure the CPU time per operation invocation, their usage is infeasible in certain situations, e.g., when using third-party or legacy applications that cannot provide the required instrumentation. For other scenarios, the costs for fine-grained instrumentation can be considered too high. Therefore, different statistical approaches have been proposed in the literature to estimate resource demands for individual operations based on aggregated measurements such as average response time or CPU utilization. These aggregated measurements are often collected by default in applications (e.g., in access log files) and in the operating system (OS). Therefore, resource demand estimation techniques can be applied in many situations where the usage of direct measurements is prohibitive.

LibReDE is a Java library providing different ready-to-use implementations of statistical approaches for resource demand estimation [15]. The library currently comes with implementations of six commonly used approaches: response time approximation [3], service demand law [3], linear regression [13], two variants of a Kalman filter [16,18] and an optimization-based approach [12]. Previous work [14] showed that the accuracy of the individual techniques strongly depends on the characteristics of the observations and the modeled system resulting in significant differences in the estimates. In order to evaluate the accuracy of the estimated resource demands, LibReDE supports the evaluation of the results using k-fold cross-validation: the input data is randomly partitioned into k equally large subsets and the estimation is repeated k times, each time using a different one of the k subsets as validation set and the others as training set. As the actual values of the resource demands are unknown, the estimation error is evaluated using the observed utilization U_{act} and the observed response times $R_{act,r}$ of operation r. The observed values are compared to the calculated ones, U_{calc} and $R_{calc,r}$, which are obtained using equations from operational analysis of queuing networks. Using the estimated resource demands, U_{calc} is determined based on the Utilization Law [8, Chap. 6]:

$$U_{calc}(\lambda) = \frac{1}{p}\sum_{r=1}^{n} \lambda^r D^r \quad (1) \qquad R^r_{calc}(\lambda) = D^r(1 + \frac{P_Q}{1 - U_{calc}(\lambda)}). \quad (2)$$

Assuming a M/M/k/PS queue for Equation 2 [8, Chap. 14]: n is the number of operations, D_r is the estimated resource demand of operation r, $\lambda = (\lambda_1, \ldots, \lambda_n)$ is a vector of arrival rates, p is the number of processor cores and P_Q is the probability that an arrival finds all servers busy (calculated using the Erlang-C formula [8, Chap. 14]).

The mean relative errors E_{util} for the utilization and $E_{rt,r}$ are then determined on the validation set V=$\{(\lambda_1^{(i)}, \ldots, \lambda_n^{(i)}, R_{act,1}^{(i)}, \ldots, R_{act,n}^{(i)}, U_{act}^{(i)}) : i = 1 \ldots m\}$:

$$e_{util} = \frac{1}{m}\sum_{i=1}^{m}\frac{|U_{act}^{(i)} - U_{calc}(\lambda^{(i)})|}{U_{act}^{(i)}} \quad (3) \qquad e_{rt}^{r} = \frac{1}{m}\sum_{i=1}^{m}\frac{|R_{act}^{(r,i)} - R_{calc}^{r}(\lambda^{(i)})|}{R_{act}^{(r,i)}} \quad (4)$$

The relative errors are calculated for each of the k validation sets and the result of the cross-validation is the mean relative error over all validation sets. Based on the relative errors, the PMG dynamically chooses an approach as described in the next section.

Estimation approach selection Selecting the right estimation approach for LibReDE makes a huge difference (in our experiments we observed differences in the range of 6% to 6000% relative response time error). Each approach has strengths and weaknesses depending on the application in place [14,15].

$$\begin{bmatrix} e_{util}^{(1)} \\ \vdots \\ e_{util}^{(i)} \\ \vdots \\ e_{util}^{(m)} \end{bmatrix} + \left(\begin{bmatrix} e_{rt}^{(1,1)} & \ldots & e_{rt}^{(1,j)} & \ldots & e_{rt}^{(1,n)} \\ \vdots & & \vdots & & \vdots \\ e_{rt}^{(i,1)} & \ldots & e_{rt}^{(i,j)} & \ldots & e_{rt}^{(i,n)} \\ \vdots & & \vdots & & \vdots \\ e_{rt}^{(m,1)} & \ldots & e_{rt}^{(m,j)} & \ldots & e_{rt}^{(m,n)} \end{bmatrix} \times \begin{bmatrix} \lambda^{(1)} & \ldots & 0 & \ldots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \ldots & \lambda^{(i)} & \ldots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & \ldots & 0 & \ldots & \lambda^{(n)} \end{bmatrix} \right) \times \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \vdots \\ 1 \end{bmatrix} \quad (5)$$

We are looking for the approach that calculates the most accurate resource demands, therefore we use both validators and select the one with the lowest relative error when combining both validation results provided by LibReDE. The utilization law validator provides a vector E_{util}, as we only use one resource, with the length of m, where m is the number of estimation approaches used. Each row in this vector contains the relative utilization error of one approach. The response time validator provides a $m \times n$ matrix E_{rt}, where m is the number of estimation approaches used and n the number of operations to estimate resource demands for. Each row i contains all relative response time errors of one approach and each column j contains the relative response time error of one operation. Therefore, the value at index i,j is the relative response time error of operation j using approach i.

Some operations might get a small amount of calls, misleading the approach selection when just selecting the approach with the smallest relative error. We weight the relative error of each operation according to the arrival rates of the input data as the number of values used for the estimation varies due to different workload on each operation. We therefore multiply the arrival rates matrix λ with the relative response time error matrix E_{rt}. The result is a weighted matrix that considers the operation call probability. To select the best suited approach we need to reduce this matrix to a vector, where each value contains a meaningful relative error for one approach considering all operations. We calculate the sum over each row of the matrix resulting in a relative response time error vector. Both vectors, containing either the response times or the CPU utilization error, are added up as shown in Equation 5.

We finally select the approach with the minimum total error in the resulting vector. The resource demands D_r of this approach are stored in the monitoring DB of the PMG. The model generation then uses these resource demands for building an architecture-level performance model.

3 Evaluation

In order to evaluate the accuracy of resource demand measurement and estimation approaches, we used two environments. The first evaluation compares the three presented approaches (PMW-Tools monitoring, Dynatrace AM and LibReDE) with each other on two levels of granularity in a virtualized environment. In the second evaluation, we use a distributed bare-metal installation and combine direct measurement and estimation approaches.

For both evaluations, we use the orders domain application of the SPECjEnterprise2010 (Version 1.03) industry standard benchmark as exemplary enterprise application. Since the benchmark defines a workload and a dataset for the test execution, the results are reproducible for others. The orders domain application is a Java EE web application comprised of servlet, JavaServer Pages (JSPs) and EJB components. The application represents a platform for automobile dealers to sell and order cars; the dealers (henceforth called users) interact with the platform using the Hypertext Transfer Protocol (HTTP). There are three basic business transactions which describe how users interact with the system: Browse, Manage and Purchase.

3.1 Standalone evaluation

For the standalone evaluation, we installed the SPECjEnterprise2010 benchmark and its corresponding load test driver on two Virtual Machines (VMs), each deployed on separate hosts (IBM System X3755M3) to avoid interferences between the two systems. The system under test (SUT) VM contains the application server, hosting the orders domain application. The other VM executes load tests on the SUT using the Faban[9] harness driver of the benchmark. Both virtual machines run openSUSE 12.3 64-bit as OS and have access to 40 gigabytes of Random Access Memory (RAM). The application server VM uses six CPU cores while the driver VM has access to four CPU cores.

The benchmark is deployed on a JBoss Application Server (AS) 7.1 in the Java EE 7.0 full profile. The DB on the test system VM is an Apache Derby DB in version 10.9.1.0. The JBoss AS and the Apache Derby DB are both executed in the same 64-bit Java OpenJDK VM (JVM version 1.7.0_17).

[9] https://java.net/projects/faban/

The first step of the evaluation is to obtain the relevant performance metrics (response time, utilization and throughput) of the SUT under different workloads by performing measurement runs. As the network overhead between the Faban harness and the SUT is not considered in the first step, the response time measurements are conducted by measuring the system entry point response times with the PMW-Tools monitoring. For this purpose, a workload of 600, 800, 1000 and 1200 concurrent users is put on the SUT, resulting in a mean CPU utilization of 39%, 56%, 69% and 79% on the server. Each measurement run lasts for sixteen minutes while data is only collected between a five minute ramp-up and a one minute ramp-down phase.

The standalone evaluation is conducted on two levels of granularity. We compare system entry point level, where only the boundaries of the system are monitored, with a component operation level monitoring, where each public operation of each used component is instrumented. This results in different performance models as resource demands are only measured or estimated for either servlet invocations (system entry point) or servlet calls and EJB operation invocations. For both cases we execute a load test with 600 concurrent users and collect monitoring data. Depending on the approach selected, this monitoring data contains either fine-grained measurements of CPU demanded time per operation invocation or only response times and total CPU utilization of the VM.

The performance models generated with this monitoring data are used for simulating the same and higher amounts of concurrent users (800 - 1200). We compare the simulated CPU utilization and the response times with actual measurements on the system. For the utilization we compare the measured mean CPU utilization (MMCPU) with the simulated mean CPU utilization (SMCPU) and calculate the relative CPU utilization prediction error (CPUPE).

When examining the CPU utilization prediction results shown in Table 1, it is visible that LibReDEs prediction is very accurate, especially in the replay case with 600 concurrent users and the upscaled case with 1200 concurrent users. The two monitoring solutions only measure the CPU time of the actual request thread while LibReDE also takes the overhead of the application server and CPU time for other processing like garbage collection (GC) into account. Dynatrace AM can use different CPU timers optimized for specific environments (i.e., VM, Windows OS, etc.) and the here used POSIX Hi-Res timer produces more accurate results than the PMW-Tools monitoring [7].

Table 1. Measured and simulated CPU utilization for system entry point level

System		**PMW-Tools monitoring**		**Dynatrace AM**		**LibReDE - estimation**	
Users	**MMCPU**	**SMCPU**	**CPUPE**	**SMCPU**	**CPUPE**	**SMCPU**	**CPUPE**
600	39,33%	36.66%	6.80%	38.73%	1.53%	39.73%	1.01%
800	55,69%	48.68%	12.58%	51.41%	7.68%	52.69%	5.37%
1000	69,28%	60.92%	12.06%	64.02%	7.58%	65.56%	5.36%
1200	79,31%	73.21%	7.69%	77.33%	2.50%	78.66%	0.82%

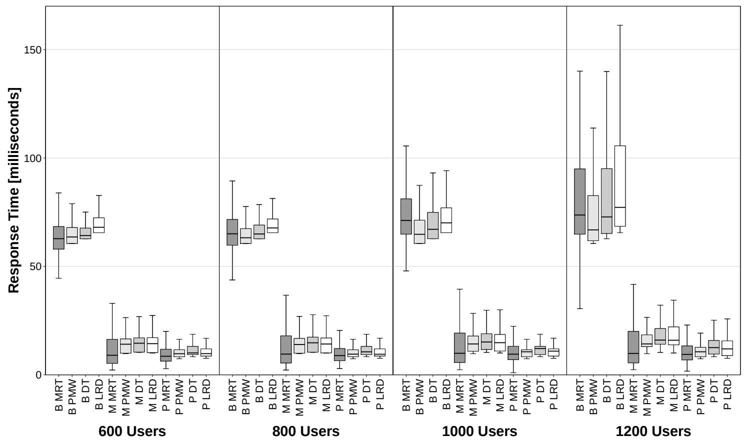

Fig. 2. Measured and simulated response times on system entry point level

Figure 2 shows the response times for system entry point level granularity using box plots. Each box depicts one measurement/simulation series. The figure is divided into four sections, distinguishing between different user amounts. In each section, three measured response time (MRT) box plots are shown, one for each business transaction: Browse (B), Manage (M), Purchase (P). The sections are completed by nine simulation box plots, one for each of the three business transactions times the three techniques: PMW-Tools monitoring (PMW), Dynatrace AM (DT) and LibReDE (LRD).

We see that LibReDE tends to overestimate the resource demands, leading to a higher median and broader Interquartile range (IQR) for the Browse and Manage transaction, but delivers good results in general. The differences between PMW-Tools monitoring and Dynatrace AM are minimal in most cases. All approaches have in common that they cannot predict the lower quartiles. However, this is most likely caused by the fact, that only mean values for CPU demands are represented in the resource demands of the generated performance models.

The CPU utilization results and errors are similar for component operation level compared to system entry point level. Table 2 shows that LibReDE again produces the most accurate resource demands when simulating and comparing

Table 2. Measured and simulated CPU utilization for component operation level

System		**PMW-Tools monitoring**		**Dynatrace AM**		**LibReDE - estimation**	
Users	**MMCPU**	**SMCPU**	**CPUPE**	**SMCPU**	**CPUPE**	**SMCPU**	**CPUPE**
600	39,33%	36.39%	7.49%	37.21%	5.39%	39.61%	0.69%
800	55,69%	48.42%	13.04%	49.83%	10.51%	52.77%	5.24%
1000	69,28%	60.26%	13.01%	61.89%	10.67%	65.71%	5.15%
1200	79,31%	71.78%	9.49%	74.07%	6.60%	79.32%	0.01%

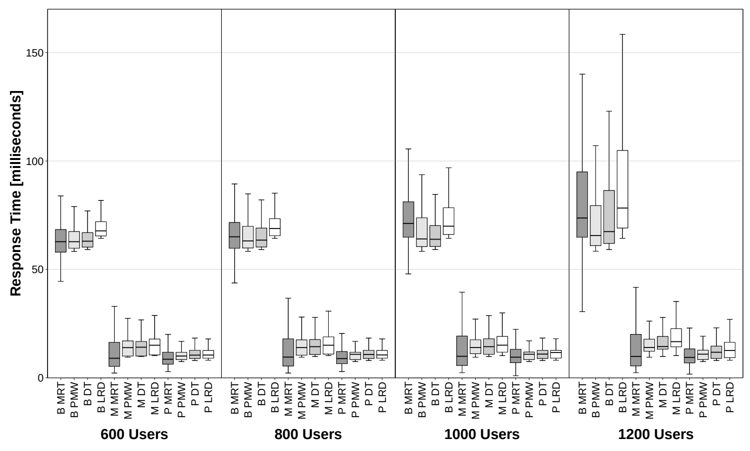

Fig. 3. Measured and simulated response times on component operation level

the CPU utilization with actual measurements. Dynatrace again is more accurate than PMW-Tools monitoring but the differences are smaller compared to the system entry point level.

The response time errors presented in Figure 3 are best predicted with direct measurements. The differences between the two monitoring approaches are rather small. LibReDE overestimates in most of the cases. The upper quartiles are better predicted using estimation than direct measurements, but the median and IQR are worse with estimation approaches. Again all approaches have in common that they cannot predict the lower quartile.

3.2 Distributed Setup

The previous evaluation showed that resource estimation techniques provide sufficiently accurate results for most of the evaluated scenarios. However, in order to use these estimations, it is important to be able to measure control flows and response time on the level of granularity that needs to be represented in a model. Furthermore, estimations work only as long as response time and throughput values for all requests are available for a measurement interval. Therefore, there are a lot of cases in which it is desirable to mix direct measurements with resource estimation techniques.

This evaluation validates a distributed deployment scenario for SPECjEnterprise2010 in which direct measurements and estimations are used in combination. This is necessary to be able to properly account for the resource demands and times spent on different layers of the architecture (e.g., what portion is spent in the DB tier). It is important to note that the following models also account for network resource demands which was not done for the previous evaluations as the standalone setup was deployed on a single server. The models for this evaluation are automatically generated using the PMG by providing input from multiple sources (PMW-Tools monitoring, Dynatrace AM, SAR and LibReDE).

The SPECjEnterprise2010 benchmark is deployed in a multi-tier architecture consisting of a presentation, application and a data tier. As we do not have an in-depth monitoring for the data tier, we use estimation here while the presentation and application tier are instrumented using the PMW-Tools monitoring as well as the Dynatrace AM. The resulting resource demands are used to build a performance model based on PCM. In order to model the data tier, the data collection solution (i.e., PMW-Tools monitoring, Dynatrace AM) gathers the tier's response times, CPU utilization on the DB is gathered using SAR. These values are used as input for a resource demand estimation using LibReDE [15]. The generated performance model is then enriched with the data tier's estimated resource demands. Finally, the model is used to perform simulations with increasing workloads; the results are then compared to measurements of the real system to gauge the prediction performance of the approach.

To obtain a multi-tier architecture, the standard orders domain application is modified by converting the EJB components to web services. This allows for the application's deployment on two different machines. In addition, the application tier is connected to a PostgreSQL DB located on a third machine.

The different tiers of the application are deployed on three different machines which in the following will be called User Interface (UI) server, Web Service (WS) server and DB server. Additionally, a benchmark driver is deployed on one VM to generate load on the whole system by accessing the UI server using the three business transactions. To achieve a moderate load on each system, the CPU core count of each system has been modified by disabling some cores. All of the systems' technical specifications are listed in Table 3.

The distributed evaluation also begins with performing similar measurement runs using minimal instrumentation. Executing the same workload (600 - 1200 users), as in the previous evaluation results in a maximum CPU utilization of 77%, 59% and 68% on the UI, WS and DB server, respectively. The benchmark driver has been modified to collect the response time of the three business transactions for each invocation, instead of measuring them directly on the SUT as in the previous evaluation.

Table 3. Software and hardware configuration of the SUT

Server	**UI Server**	**WS Server**	**DB Server**
Application	SPECjEnterprise2010 (version 1.03) orders domain		
AS/DB	GlassFish 4.0 (build 89)	JBoss AS 7.1.1	PostgreSQL 9.2.7
JVM	64-bit Java HotSpot JVM version 1.7.0_71	64-bit Java OpenJDK JVM version 1.7.0_40	-
OS	openSUSE 12.2		openSUSE 12.3
CPU Cores	2 x 2.1 GHz	6 x 2.1 GHz	4 x 2.4 GHz
CPU Sockets	4 x AMD Opteron 6172		2 x Intel Xeon E5645
RAM	256 GB		96 GB
Hardware System	IBM System X3755M3		IBM System X3550M3
Network	1 gigabit-per-second (GBit/s)		

Table 4. Measured and simulated CPU utilization using PMW-Tools monitoring

	UI server			**WS server**			**DB server**		
Users	**MMCPU**	**SMCPU**	**CPUPE**	**MMCPU**	**SMCPU**	**CPUPE**	**MMCPU**	**SMCPU**	**CPUPE**
600	39.97%	40.36%	0.96%	30.96%	26.93%	14.96%	34.51%	40.77%	15.35%
800	53.11%	54.05%	1.74%	41.86%	36.11%	15.94%	45.89%	54.54%	15.86%
1000	65.27%	67.37%	3.11%	48.39%	44.99%	7.57%	56.51%	68.02%	16.93%
1200	77.01%	80.52%	4.36%	59.71%	53.81%	10.96%	68.38%	81.42%	16.01%

Table 5. Measured and simulated CPU utilization using Dynatrace AM

	UI server			**WS server**			**DB server**		
Users	**MMCPU**	**SMCPU**	**CPUPE**	**MMCPU**	**SMCPU**	**CPUPE**	**MMCPU**	**SMCPU**	**CPUPE**
600	39.97%	33.29%	20.06%	30.96%	30.54%	1.36%	34.51%	34.25%	0.77%
800	53.11%	44.47%	19.43%	41.86%	40.82%	2.55%	45.89%	45.80%	0.20%
1000	65.27%	55.55%	17.49%	48.39%	51.03%	5.17%	56.51%	57.20%	1.21%
1200	77.01%	66.82%	15.25%	59.71%	61.34%	2.66%	68.38%	68.92%	0.79%

Afterwards, the UI and WS server are instrumented and another benchmark run with a workload of 600 concurrent users is performed. The collected data is used to generate a performance model using the PMG. Simultaneously, the response times per invocation and aggregated utilization of the DB server are collected. These are automatically used by the PMG as input for the LibReDE resource demand estimation. The model is further enhanced by adding latency and throughput values of the network connecting the individual servers as shown in [4]. These values are gathered using the lmbench[10] benchmark suite. Finally, the finished model is used to simulate the SUT with a workload of 600, 800, 1000 and 1200 concurrent users; the duration and steady state times correspond to the ones used for the measurements.

When examining the CPU utilization values in Table 4 and Table 5, we see that the SMCPU of the DB server is predicted with very high accuracy using Dynatrace AM, with the highest error being 1.21% at 1000 concurrent users. The PMW monitoring does not intercept all JDBC calls, leading to an overestimation of CPU demands on the calls that are intercepted. Furthermore, the accounting of this calls is also missing in the WS server, leading to an underestimation of the CPU demands in the business tier. The CPU utilization of the WS server is predicted very well using Dynatrace AM, while the UI server's utilization is predicted too low. Dynatrace distributes the processing time to all active operations. We have more running operations on the WS server, leading to better results for this tier compared to the UI server. The PMW monitoring instruments the CPU demands of the UI server better, because its servlet interceptor measures each operation individually. Overall, the results show that the approach is well suited for predicting the performance of a multi-tier application.

The response time values are illustrated in the box plots in Figure 4. The figure is divided into four sections, one section for each user amount. Each section again contains three MRT series (Browse, Manage, Purchase) and six simulation

[10] http://lmbench.sourceforge.net/

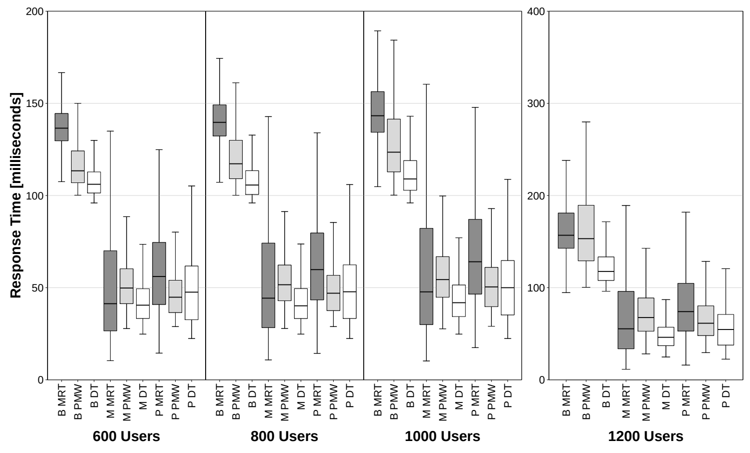

Fig. 4. Measured and simulated response times

box plots. Three plots for the combination PMW-Tools monitoring and LibReDE (PMW) and three plots for the combination Dynatrace AM and LibReDE (DT). Note that the last section uses another scale as the first three sections, as the response times are significantly higher with 1200 concurrent users.

The comparison shows that the combination of resource demand measurement and estimation techniques leads to a good representation of the real system. The median of the simulated response time is close to the actual measurements. The prediction error for the median response time values is at most 25.02% for the browse transaction at 1200 concurrent users. The IQR prediction using PMW is usually a bit closer to the real system measurements than DT.

4 Related Work

This section presents related work that is concerned with measurement accuracy in different environments or the overhead caused by such measurements.

CPU accounting on VMs can be error prone due to sharing the same physical resource over multiple machines. Hofer et al. [9] discovered that malicious accounting, so called steal time, can be detected and calculated in a VM. If not corrected, CPU utilization measurements produce wrong resource demands. Wrong CPU utilization accounting decreases the quality of performance models created either using direct measurement or estimation methods. We avoid this by isolating the SUT VM on a single host. However, virtualized environments need to correct this steal time in order to calculate accurate resource demands.

Estimating the overhead of virtualized environments has been described by Brosig et al. [2] and Huber et al. [10]. These approaches estimate, among others, virtualization overhead based on monitoring data using a queuing network. Such calculations can increase the accuracy of resource demands of such environments.

Kuperberg compared different timers and measurement approaches for a number of systems [11]. While the Dynatrace AM already offers different timers to select the most suitable one, the other two approaches rely on either the ThreadMXBean, JMX monitoring or SAR. The accuracy of these approaches can vary depending on the underlying system monitored and therefore the calculated resource demands accuracy may vary.

Measurement approaches cause overhead on the SUT. Brunnert et al. [5] measured and discussed this effect for the PMW-Tools monitoring solution in previous work. This overhead effect turns out to be at around 0.003 ms for each measurement when only CPU no other resource demands are collected. This overhead can effect the system at its capacity limits, while an estimation approach can use coarse-grained monitoring data with less overhead.

5 Conclusion and Future Work

This work compared three different techniques for deriving resource demands for performance models. We compared a monitoring approach from academia, an industry monitoring solution and a library combining six different estimation approaches. These techniques have been integrated into a single automatic PMG. The evaluation compared all techniques in a standalone and a distributed setup, as well as in a virtualized and a bare-metal environment for two levels of granularity: system entry point level and component operation level.

All techniques deliver good results for both granularity levels and in all environments. Estimation techniques deliver better results for the system entry point level, but fall short behind direct measurements for the component operation level. Furthermore, direct measurements can extract resource demands on any level of detail, while estimation techniques must calculate demands for the complete system to distribute the measured utilization among the components. Estimation techniques can be applied to a broad variety of technologies as the requirements for data collection are lower. We demonstrated accurate results using a hybrid setup, where measurement approaches are used to extract resource demands for the UI and WS combined with estimations for the DB.

The evaluation uses a Java EE application. Industry monitoring like Dynatrace AM are capable of observing other technologies. Demonstrating the applicability of the framework for other technology stacks as well as extending the monitored resources are interesting challenges for further research.

References

1. Becker, S., Koziolek, H., Reussner, R.: The Palladio Component Model for Model-Driven Performance Prediction. Journal of Systems and Software 82(1), 3–22 (2009), special Issue: Software Performance - Modeling and Analysis
2. Brosig, F., Gorsler, F., Huber, N., Kounev, S.: Evaluating approaches for performance prediction in virtualized environments. In: Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2013 IEEE 21st International Symposium on. pp. 404–408 (Aug 2013)

3. Brosig, F., Kounev, S., Krogmann, K.: Automated Extraction of Palladio Component Models from Running Enterprise Java Applications. In: Proceedings of the 1st International Workshop on Run-time Models for Self-managing Systems and Applications (ROSSA 2009). ACM, New York, NY, USA (2009)
4. Brunnert, A., Krcmar, H.: Continuous Performance Evaluation and Capacity Planning Using Resource Profiles (under review). Journal of Systems and Software (2015)
5. Brunnert, A., Neubig, S., Krcmar, H.: Evaluating the Prediction Accuracy of Generated Performance Models in Up- and Downscaling Scenarios. pp. 113–130 (2014)
6. Brunnert, A., Vögele, C., Krcmar, H.: Automatic Performance Model Generation for Java Enterprise Edition (EE) Applications. In: Computer Performance Engineering, Lecture Notes in Computer Science, vol. 8168, pp. 74–88. Springer Berlin Heidelberg (2013)
7. Dynatrace: Dynatrace Agent Timers. `https://community.compuwareapm.com/community/display/DOCDT60/Agent+Timers`, accessed: 14.05.2015
8. Harchol-Balter, M.: Performance Modeling and Design of Computer Systems. Cambridge University Press, New York, NY, USA (2013)
9. Hofer, P., Hörschläger, F., Mössenböck, H.: Sampling-based Steal Time Accounting Under Hardware Virtualization. In: Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering (ICPE 2015). pp. 87–90. ACM, New York, NY, USA (2015)
10. Huber, N., von Quast, M., Hauck, M., Kounev, S.: Evaluating and Modeling Virtualization Performance Overhead for Cloud Environments. In: Proceedings of the 1st International Conference on Cloud Computing and Services Science (CLOSER 2011). pp. 563–573. SciTePress (May 2011)
11. Kuperberg, M.: Quantifying and Predicting the Influence of Execution Platform on Software Component Performance, vol. 5. KIT Scientific Publishing (2010)
12. Menascé, D.A.: Computing Missing Service Demand Parameters for Performance Models. In: Proceedings of the 2008 Computer Measurement Group Conference (CMG 2008). pp. 241–248. Las Vegas, NV, USA (2008)
13. Rolia, J., Vetland, V.: Parameter Estimation for Performance Models of Distributed Application Systems. In: Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research (CASCON '95). p. 54. IBM (1995)
14. Spinner, S.: Evaluating Approaches to Resource Demand estimation. Master's thesis, Karlsruhe Institute of Technology (KIT) (2011)
15. Spinner, S., Casale, G., Zhu, X., Kounev, S.: LibReDE: A Library for Resource Demand Estimation. In: Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering (ICPE 2014). pp. 227–228. ACM, New York, NY, USA (2014)
16. Wang, W., Huang, X., Qin, X., Zhang, W., Wei, J., Zhong, H.: Application-Level CPU Consumption Estimation: Towards Performance Isolation of Multi-tenancy Web Applications. In: Proceedings of the 5th International Conference on Cloud Computing (CLOUD). pp. 439–446. IEEE (2012)
17. Willnecker, F., Brunnert, A., Gottesheim, W., Krcmar, H.: Using Dynatrace Monitoring Data for Generating Performance Models of Java EE Applications. In: Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering (ICPE 2015). pp. 103–104. ACM, New York, NY, USA (2015)
18. Zheng, T., Woodside, M., Litoiu, M.: Performance Model Estimation and Tracking Using Optimal Filters. IEEE Transactions on Software Engineering 34(3), 391–406 (2008)