Using Architecture-Level Performance Models as Resource Profiles for Enterprise Applications

Andreas Brunnert
fortiss GmbH
Guerickestr. 25
80805 München, Germany
brunnert@fortiss.org

Kilian Wischer, Helmut Krcmar
Technische Universität München
Boltzmannstr. 3
85748 Garching, Germany
{wischer, krcmar}@in.tum.de

ABSTRACT

The rising energy and hardware demand is a growing concern in enterprise data centers. It is therefore desirable to limit the hardware resources that need to be added for new enterprise applications (EA). Detailed capacity planning is required to achieve this goal. Otherwise, performance requirements (i.e. response time, throughput, resource utilization) might not be met. This paper introduces resource profiles to support capacity planning. These profiles can be created by EA vendors and allow evaluating energy consumption and performance of EAs for different workloads and hardware environments. Resource profiles are based on architecture-level performance models. These models allow to represent performance-relevant aspects of an EA architecture separately from the hardware environment and workload. The target hardware environment and the expected workload can only be specified by EA hosts and users respectively. To account for these distinct responsibilities, an approach is introduced to adapt resource profiles created by EA vendors to different hardware environments. A case study validates this concept by creating a resource profile for the SPECjEnterprise2010 benchmark application. Predictions using this profile for two hardware environments match energy consumption and performance measurements with an error of mostly below 15 %.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: measurement techniques, modeling techniques; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Measurement; Performance

Keywords

Performance Modeling; Palladio Component Model; Resource Profile; Energy Consumption; Capacity Planning

QoSA'14, June 30–July 4, 2014, Marcq-en-Baroeul, France.

ACM 978-1-4503-2576-9/14/06 ...$15.00.
http://dx.doi.org/10.1145/2602576.2602587.

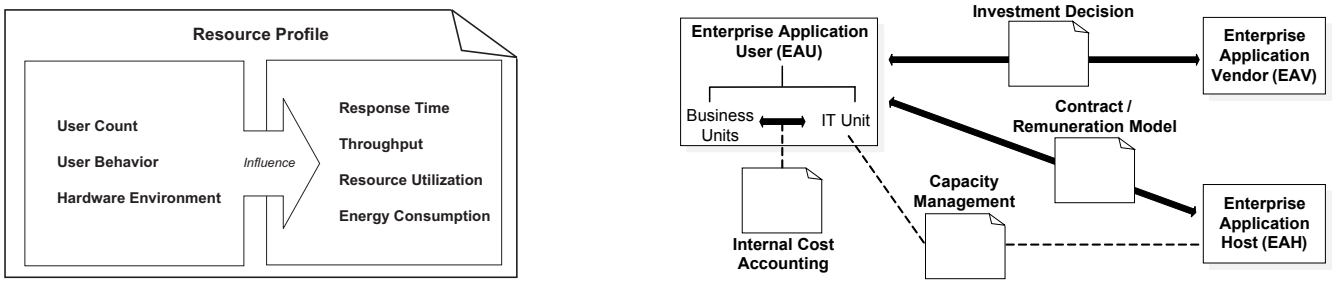
1. INTRODUCTION

Enterprise applications are the backbone of many business processes. These applications need to meet performance requirements (i.e. response time, throughput, resource utilization) to avoid problems during the process execution. Detailed capacity planning effort [17] is therefore required before new enterprise applications are deployed in a data center. This effort is driven by a basic question: How much hardware resources are required to fulfill performance requirements for the expected workload? This question leads to a more specific inquiry about the applications demand for hardware resources such as central processing units (CPU), hard disk drives (HDD) and memory. The capacity planning finally requires an assessment of the workload impact on the resource demand.

At the same time, the rising energy consumption is a major cost driver in data centers nowadays [8, 22]. The energy consumption is defined as the power consumption integrated over time. It would thus be beneficial to know how much power will be consumed by the resources utilized by an application in beforehand [6].

Estimating hardware requirements and energy consumption of new enterprise application deployments is only possible if different parties work together [4]. First of all, enterprise application vendors (EAV, i.e. software or consulting companies) need to quantify the resource demand of their software products. Enterprise application users (EAU, i.e. companies who source software from EAVs) need to specify the expected workload for their use cases. Finally, enterprise application hosts (EAH, i.e. data center providers) need to specify the characteristics of the hardware environment on which the applications can be deployed. Nowadays, the communication between these parties lacks a medium in which all these aspects are captured. The resource profile concept introduced in this work serves as such a communication medium and helps to answer the questions raised above.

This work proposes the use of architecture-level performance models to represent resource profiles. The contribution of this work therefore includes the resource profile concept as a use case for these models in between their traditional application domains: software performance engineering [24] and application performance management [18]. Additionally, an approach is presented to adapt these models to different hardware environments. Furthermore, a performance modeling approach is extended to allow predictions of the energy consumption. A case study finally evaluates the feasibility of the resource profile concept.



(a) Resource profile

(b) Use cases for a resource profile

Figure 1: Resource profiles for enterprise applications

2. RESOURCE PROFILES

Resource profiles are models that allow evaluating energy consumption and performance of enterprise applications (see figure 1(a)). In an ideal case, a resource profile is constructed once by an EAV and can then be used by EAUs and EAHs for several use cases as shown in figure 1(b). The intended use cases are similar to the ones of system requirements for end user desktop software (such as games or office tools). System requirements show whether a user is able to run a software on his current environment. If an end user needs to modify his soft- or hardware environment, this often has a huge impact on his purchasing decision. If EAUs would have similar information at hand during a software purchasing scenario, it would also influence their investment decisions.

2.1 Content and Structure

The energy consumption and performance of an enterprise application is influenced by several factors as depicted in figure 1(a). One of the key factors is the hardware environment on which an enterprise application is deployed. As hardware resources (e.g. the CPU) have different performance and power consumption characteristics, the resource demand (e.g. required CPU time) and energy consumption of single transactions is directly dependent on the components used within a server. Different hardware environments thus have a big impact on the performance and energy consumption of an application. Resource profiles therefore provide a means to represent different hardware environments.

Besides the hardware environment, the workload on a system directly influences the performance and energy consumption of an application. The workload of an enterprise application is typically specified by the amount of users accessing the system and their behavior [16]. Depending on the workload, the hardware environment is utilized on different levels, which lead to varying performance and energy characteristics. Consequently, the user count and their behavior can be represented in a resource profile.

To evaluate the impact of different hardware environments and workloads on energy consumption and performance, a model is required that describes the performance-relevant aspects of an enterprise application architecture independently from these influencing factors. These aspects include the components of an enterprise application, their interfaces, relationships, control flow, parametric dependencies and resource demands [12]. Resource profiles describe these aspects independently from the workload and hardware environment. To simplify their use, the aforementioned aspects are hidden for resource profile users (i.e. EAU, EAH). Resource profiles abstract these aspects on the level of detail of single deployment units of an enterprise application. These deployment units represent a collection of application components and thus reduce the complexity for the users. Instead of dealing with individual application components, they can use these deployment units for specifying allocations on different hardware environments.

To represent the workload, resource profiles describe external interfaces provided by an enterprise application (e.g. functionalities provided by user interfaces). These external interfaces can be used for specifying the workload. Based on these specifications, the influence of different workloads and hardware environments on performance and energy consumption can be evaluated as shown in figure 1(a).

The knowledge required for constructing a resource profile and for specifying the input variables for the evaluation is often distributed between different parties (i.e. EAV, EAU and EAH, see figure 1(b)). Resource profiles are therefore meant to be used differently depending on the available information. An EAV should create resource profiles for all enterprise applications sold (off-the-shelf and custom developments), which then can be adapted by EAUs and EAHs for their specific needs. They can modify the workload and the hardware environment but reuse the specifications provided by an EAV.

2.2 Use Case Examples

The primary use case of a resource profile is to estimate the required hardware resources for an enterprise application. As this task is required in different contexts, the following paragraphs explain some use cases in which the transferable nature of resource profiles helps to simplify the relationships of EAUs, EAVs and EAHs.

If an EAU is interested in a new enterprise application, he could use the corresponding resource profile as one component in an overall investment decision. The EAU can specify the expected amount of users, their behavior and his existing hardware environment to evaluate if this hardware would be sufficient to run the application for his needs. At the same time, the EAU could evaluate the impact of this particular application on his energy bill. If new hardware is needed for the application, the resource profile helps to compare different hardware configurations in terms of their impact on performance and energy consumption. Resource profiles can also be used to choose between different off-the-shelf software products with similar functionality with regard to the above mentioned criteria.

When software is purchased and hosted by an EAU internally, a resource profile supports the internal cost accounting between different business units and the IT unit [3]. Using resource profiles, the resource demand and power consumption of an enterprise application can be broken down to user profile levels or transaction classes. Brandl et al. [3] showed how such a breakdown of the resource consumption on the level of transaction classes can be used to allocate costs to different business units according to their workload.

If an EAU does not want to host an application himself, resource profiles can be used to negotiate a contract between an EAU and an EAH (e.g. cloud providers). As cloud computing is gaining more popularity, the demand for usage-based costing will increase [3]. Similar to the internal cost accounting approach explained above, EAUs and EAHs could agree on a remuneration model which is directly dependent on the resource and energy consumption of the hosted application [13]. Resource profiles help both parties to better estimate their costs in such a scenario.

If an enterprise application is already running in a production environment, resource profiles help in the capacity management process. For example, the impact of an increased user load on performance and energy consumption of an application can be examined in beforehand and appropriate conclusions can be drawn.

The next section explains the construction of resource profiles based on architecture-level performance models.

2.3 Performance Models as Resource Profiles

Evaluating the performance of an application is a common problem in the software engineering domain. Numerous performance modeling approaches have been proposed to address this challenge [1, 12]. A performance model of an enterprise application typically contains performance-relevant aspects of an application architecture, the hardware environment and the workload. Using these models as input for analytical solvers or simulation engines allows predicting response time, throughput and resource utilization for the modeled software system. Several case studies (see section 4) showed the applicability of performance models for predicting these performance metrics.

Performance models thus seem to be generally suitable to represent resource profiles. As resource profiles must be adapted to different target environments, a key requirement for their representation is that performance-relevant aspects of an application architecture, the hardware environment and the workload can be modeled independently from each other. In an ideal case, an EAV can model the performance-relevant aspects of an application architecture and distribute this incomplete model to an EAU who complements the model with the expected workload. Afterwards, an EAH can add the hardware environment to the resource profile.

Conventional performance modeling approaches [1] such as Queuing Networks, Layered Queuing Networks (LQN) or Queuing Petri Nets depict all these aspects nested in one single monolithic performance model. It is therefore hard to change a single aspect like the hardware environment or the workload without needing to substantially change the whole performance model. Architecture-level performance models [12] try to separate these aspects to simplify the modeling process. A popular architecture-level performance model is the Palladio Component Model (PCM) [20]. PCM separates all the aspects mentioned above and is thus used as meta-model to represent resource profiles.

PCM is described by Reussner et al. [20] as a software component model for business information systems to enable model-driven quality of service (QoS, i.e. performance) predictions. A software system is represented in PCM by several model layers which can reference each other [20]. The most important model within PCM is called repository model. This model contains the components of a software system and their relationships. The control flow of a component operation, its resource demand and parametric dependencies are specified in so called Resource Demanding Service Effect Specifications (RDSEFF). Components are assembled in a system model to represent an application. User interactions with the system are described in a usage model. The other two model types in PCM are the resource environment and allocation model. A resource environment model allows to specify available resource containers (i.e. servers) with their associated hardware resources (CPU or HDD). An allocation model specifies the mapping of system model elements on resource containers.

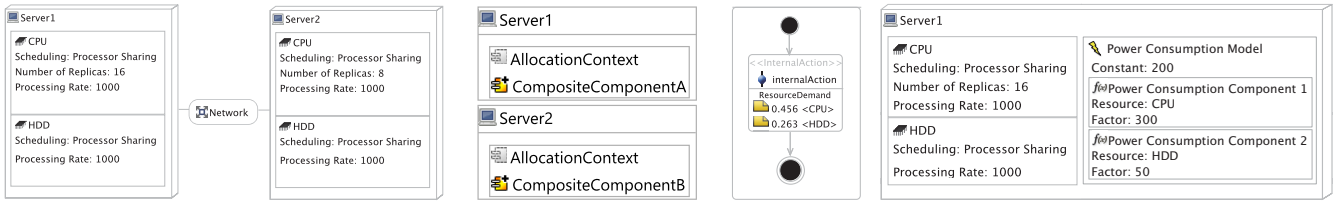
A resource profile can thus be represented using this meta-model by creating a system model. Such a system model describes the external interfaces of an enterprise application and references repository model components (including RDSEFFs) to describe the performance-relevant aspects of an application architecture. Several approaches to construct these models, either based on design model transformations [2] or dynamic analysis [5], already exist and are therefore not described in detail in this work. The workload for a PCM-based resource profile can be specified in a usage model. The hardware environment can be specified by the resource environment model whereas the deployment on this environment is specified in the allocation model.

Even though PCM provides a good foundation for building resource profiles, just creating a performance model is not sufficient for supporting the structure and use cases of a resource profile outlined earlier. The remainder of this section therefore focuses on questions that cannot be answered by the PCM modeling capabilities:

- How can resource profiles based on the PCM meta-model be adapted to different hardware environments?
- How can the PCM meta-model be extended to support energy consumption predictions?

2.4 Adapting Resource Profiles to Different Hardware Environments

Hardware environments are specified in PCM resource environment models. An example of a resource environment model is shown in figure 2(a). In this example, the resource environment consists of two hardware servers which are connected via a network connection. The model depicts the CPU and HDD of both servers. The CPU of *Server1* consists of 16 cores (number of replicas) and the CPU of *Server2* consists of 8 cores. Each core has a processing rate of 1000. The HDDs of both servers also specify a processing rate of 1000. A processing rate of 1000 means that one CPU core respectively the HDD can process 1000 units of work within a simulated time frame. In this example, one simulated time frame is interpreted as one second. Thus, each CPU core and HDD can process 1000 milliseconds (ms) of work per simulated time frame.



(a) Resource environment model (b) Allocation model (c) RDSEFF (d) Power consumption model

Figure 2: PCM models

An allocation model defines how system model elements are mapped on servers in the resource environment model. To simplify the use of a resource profile for evaluating different deployment options, an EAV should represent indivisible deployment units using composite components [20]. Composite components allow to combine several repository model components so that they can only be allocated as a whole and not individually. This representation ensures that EAUs or EAHs cannot evaluate deployment options which are not supported by an EAV. The allocation model in figure 2(b) shows the allocation of two composite components on the two servers modeled in the resource environment model. *CompositeComponentA* is mapped on *Server1* and *CompositeComponentB* on *Server2*.

A component implements several operations which can be invoked by users or other components. The resource demand of an operation is defined in internal actions of an RDSEFF. It is specified as the amount of units of work needed on a particular hardware resource to be processed. This resource demand is thus modeled relative to the processing rate of a hardware resource in the resource environment model. Figure 2(c) shows a simplified RDSEFF. The internal action consumes 0.456 units of work on the corresponding CPU resource and 0.263 units of work on the HDD. If this component is mapped on a resource container with a CPU core that can process 1000 units of work within one second, the 0.456 units of work can also be interpreted as 0.456 ms CPU time required by the component operation. The same interpretation is valid for the HDD demand.

This dependency between resource demands in RDSEFFs and processing rates of hardware resources in resource environment models avoids the need to adapt resource demands of every internal action in every RDSEFF if hardware resources are changed. It is only necessary to adapt the processing rate of modeled hardware resources if they are replaced by other types of the same resource. If more attributes of a hardware environment are changed (e.g. the number of servers), the allocation model must also be changed to map the (composite) components on the new resources. What needs to be done to model these structural changes is described by Reussner et al. [20].

If a resource profile is used to predict performance and energy consumption for a hardware environment that is not the one the resource profile was initially created with, the profile must be adapted to the target environment. To adapt the resource profile from one environment to another, the processing rate of resources must be scaled according to the performance of the hardware resources.

Following Menascé [16], the processing rate of a resource is scaled according to the hardware benchmark results of the initial and target hardware resource. After performing a suitable benchmark on the initial and the target server, we assume to get two benchmark scores of the investigated hardware resource. The benchmark score of the initial ($b_{initial}$) and target (b_{target}) server and the initial processing rate ($r_{initial}$) allow to calculate the new processing rate (r_{target}) for the target server's resource as follows:

$$r_{target} = \frac{b_{target}}{b_{initial}} * r_{initial} \quad (1)$$

For example, when benchmarking a specific hardware resource, an initial server gets a benchmark score of 40 whereas a score of 50 is achieved on the target server. By using formula 1 with an initial processing rate of 1000 a target processing rate of 1250 can be calculated. This calculation is possible for different hardware resources. For CPU benchmarks it is important that the benchmark can evaluate the performance of a single core, otherwise it is much harder to adapt the resource environment model from one server to another. If standardized benchmarks are used for this purpose, the benchmarks must not necessarily be performed by the user of a resource profile, as results for common hardware systems are often available on the web sites of the benchmark providers. Nevertheless, processing rates and benchmark scores of hardware resources used to derive resource demands need to be distributed along with a resource profile.

This approach assumes that all resource demands in the RDSEFFs of a repository model are initially derived from measurements on the same hardware types. Otherwise, it would be necessary to adapt the resource demands in an RDSEFF individually if a component is moved from one server to another in the allocation model. Similarly, the network traffic between all components needs to be represented in a resource profile even if the profile is created on a single machine. Without this information, it would not be possible to distribute components in a resource profile to different machines connected by a network.

2.5 Predicting Energy Consumption

To the best of our knowledge, there is no performance modeling approach available which is able to predict the energy consumption of an application. Hence, it can also not be predicted using PCM. As energy consumption is defined as the power consumption integrated over time, the PCM meta-model is extended by a *power consumption model* element. The PCM simulation engines SimuCom [2] and EventSim [19] are also extended to use the new *power consumption model* element. Simulation results now allow to evaluate the power consumption of servers in a resource environment over time. The integral of the resulting function can be used to predict the energy consumption of an appli-

cation. The construction of a *power consumption model* is explained in the following.

PCM is already capable of predicting the utilization rate of modeled hardware resources such as CPU or HDD. The full server power consumption can thus be modeled based on the utilization of the server's hardware resources as shown in several existing works on this topic [8, 21]. Full server power consumption refers to the power consumed by the power adapters of the server. As shown in section 2.2, this is the key figure from an economic point of view.

Rivoire et al. [21] and Fan et al. [8] showed that simple linear power models based on resource utilization metrics produce very accurate results. Even very simple models, which only capture the CPU utilization to predict the power consumption, are very accurate. A linear model with the predicted power consumption of a server as the dependent variable P_{pred} and multiple resource utilization metrics as the independent variables u_i can therefore be specified by the following equation [8, 21]:

$$P_{pred} = C_0 + \sum_{1}^{i} C_i * u_i \quad (2)$$

In PCM resource environment models a hardware server is represented by a resource container. The new *power consumption model* element is thus attached to the existing resource container meta-model element. It represents a linear model in the form of equation 2. Figure 2(d) shows an example of such a *power consumption model* for one server with 16 CPU cores and a HDD resource. A *power consumption model* contains a constant (C_0) which represents an approximation of the idle power consumption of a server. The independent variables of equation 2 are represented by multiple *power consumption components*. A *power consumption component* contains a multiplication factor (C_i) and a reference to the utilization of a hardware resource of the resource container (u_i). In the example of figure 2(d), the *power consumption model* represents the equation: $P_{pred} = 200 + 300 * u_{CPU} + 50 * u_{HDD}$. A CPU utilization of 50 % and a utilization of the HDD of 20 % would thus lead to a predicted power consumption of 360 watts (W).

The best way to construct such a linear *power consumption model* is a calibration run on the target hardware similar to the approach presented by Economou et al. [7]. In a calibration run, hardware resources are stressed independently from each other with changing intensity. Meanwhile the corresponding resource utilization metrics reported by the operating system and the resulting power consumption of the full server system are measured. Modern enterprise servers implement power measurement sensors for single hardware resources and the whole system. A common way to offer their measurements is to use the Intelligent Platform Management Interface (IPMI)[1]. Thus, there is often no need to use an external power meter device. After finishing the calibration run, a linear regression on the measured metrics is done to generate a linear *power consumption model*.

3. EVALUATION

In this section, a case study using the SPECjEnterprise-2010 industry standard benchmark demonstrates the feasibility of the resource profile concept. A resource profile based on the extended PCM meta-model is generated for the SPECjEnterprise2010[2] benchmark application on an initial hardware environment and is adapted to a target hardware environment by using the SPEC CPU2006[3] benchmark. Afterwards, workloads using varying amounts of users are executed both as a simulation using the resource profile and on deployments on the initial and target hardware environment. The simulated and measured response time, throughput, power consumption and resource utilization values are afterwards compared with each other. The evaluation steps and the quantitative validation ensure that the extended PCM meta-model provides a solid base for representing resources profiles with the properties explained in section 2.1.

3.1 SPECjEnterprise2010

The SPECjEnterprise2010 benchmark application represents business processes of an automobile manufacturer and is divided into three different domains: the Supplier domain, the Manufacturing domain and the Orders domain. The Orders domain is used by automobile dealers to sell and order cars. By doing so, they drive the demand for the Manufacturing domain. This domain simulates car manufacturing sites. It interacts with the Supplier domain to order parts required during the manufacturing process.

The evaluation in this paper focuses on the Orders domain as it is intended to be used by end users, whereas the other two domains are used by other applications as (web-) services. The communication between the domains is disabled in order to avoid the need to model and evaluate all domains.

The Orders domain is a Java Enterprise Edition (EE) web application that is composed of Servlet, JavaServer Pages (JSP) and Enterprise JavaBean (EJB) components. The automobile dealers access this application using a web interface over the hypertext transfer protocol (HTTP). The automobile dealers can perform three different business transactions: Browse (B), Manage (M) and Purchase (P). These three business transactions are composed of several HTTP requests to the system.

The dealer interactions with the system are implemented as benchmark driver in the Faban harness[4]. Faban is a load generation framework which is used to execute load on a SPECjEnterprise2010 deployment. For each benchmark run one can specify the number of dealer clients that interact with the Orders domain (benchmark scale). The official driver is implemented in a way that the benchmark scale not only influences the total number of clients but also the behavior of a single simulated client. As this is not typical for an enterprise application, the benchmark driver is patched so that the behavior of a dealer client is now independent of the total number of clients.

[1] http://www.intel.com/design/servers/ipmi/

[2] SPECjEnterprise is a trademark of the Standard Performance Evaluation Corp. (SPEC). The official web site for SPECjEnterprise2010 is located at http://www.spec.org/jEnterprise2010.

[3] The SPECjEnterprise2010 and SPEC CPU2006 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 results in this publication should thus be seen as estimates as the benchmark execution might deviate from official run rules. The official web site for SPEC CPU2006 is located at http://www.spec.org/cpu2006.

[4] http://java.net/projects/faban/

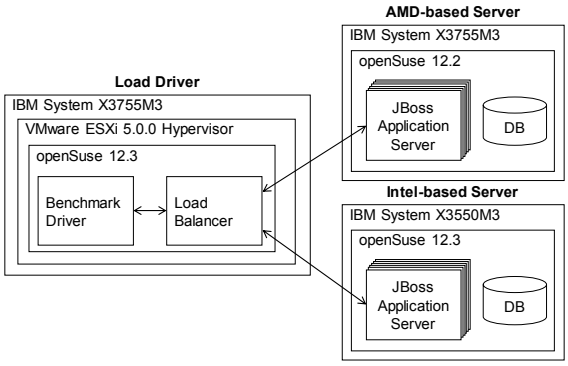


Figure 3: SPECjEnterprise2010 system topology

3.2 System Topology

An overview of the system topology is given in figure 3. The initial server, on which a resource profile represented as a system model is created, is an IBM System X3755M3 server. This machine, hereafter referred to as AMD-based server, contains 256 gigabytes (GB) random-access memory (RAM) and four AMD Opteron 6172 processors with four cores and a 2.1 GHz frequency each. The target server, which is represented by adapting the model as explained in section 2.4, is an IBM System X3550M3 server. This machine, hereafter referred to as Intel-based server, contains 96 GB RAM and two Intel Xeon E5645 processors with 6 cores and a 2.4 GHz frequency each. The hyper-threading capability of the Intel processors is disabled for this evaluation. The operating system on the AMD-based server is openSuse 12.2 whereas openSuse 12.3 is used on the Intel-based server. Six JBoss Application Server (AS) 7.1.1 instances are deployed on both servers, all running the SPECjEnterprise2010 benchmark application. Every AS instance uses its own Apache Derby DB in version 10.9.1.0 as persistence layer. The JBoss AS instances and the Apache Derby DBs are executed within a 64 bit Java OpenJDK Server virtual machine (VM) in version 1.7.0. The mod_cluster[5] web server module is used as load balancer for the JBoss AS clusters on a separate VM. The benchmark driver which generates the load on the systems under test (SUT) is also deployed on the same VM. The VM is mapped on the same type of hardware server as the AMD-based server explained above using the VMware ESXi 5.0.0 (build 469512) hypervisor. The VM runs openSuse 12.3 as operating system and is configured to have eight virtual CPU cores and 80 GB RAM. All servers are connected using a one gigabit per second (Gbit/s) network connection.

3.3 Creating & Adapting the Resource Profile

To construct a system model of the SPECjEnterprise2010 benchmark application on the AMD-based server, we use an automatic performance model generation approach presented in our previous work [5]. The PCM model generator is configured to represent the CPU resource demand in ms in the generated repository model components used within the system. A moderate load (∼50 % CPU utilization) is generated using the benchmark driver for a period of 20 minutes. During this time, data is collected and afterwards used to generate a system model for the SPECjEnterprise2010 benchmark application. Details of this process can be found in [5]. Afterwards, the system model is complemented with a usage model, which represents the workload generated by the benchmark driver, and a resource environment model that represents the hardware environment of the AMD-based server.

As outlined in section 2.4, the RDSEFFs of repository model components used within the system model contain CPU demand values specific for the AMD-based server. To adapt this information for the Intel-based server, the processing rate of CPU cores represented in the resource environment model needs to be changed. Additionally, the number of CPU cores needs to be reduced. SPEC CPU2006 is used to benchmark the CPU cores of both servers. The benchmark consists of an integer (SPECint) and a floating point (SPECfp) benchmark which again consist of several sub-benchmarks [9]. To calculate the adapted processing rate, the SPEC CPU2006 integer benchmark is executed on the AMD- and Intel-based servers. The AMD-based server achieved a benchmark score of 12.91 for the SPECint_base2006 metric. The Intel-based server achieved a benchmark score of 18.92. By using equation 1, a processing rate of 1464 is calculated for CPU cores in the adapted resource environment model of the Intel-based server.

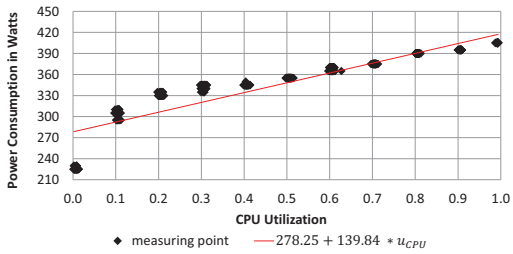
In a next step, *power consumption models* are added to the resource environment models of the AMD- and Intel-based servers, as explained in section 2.5. As these models only depict the CPUs of both servers, their *power consumption models* can only model the dependency between the CPU utilization rate and the power consumption. This constraint is acceptable as Capra et al. [6] state that the CPU consumes the majority of the overall power of a server and the power consumption is significantly dependent on the CPU's utilization rate. All other hardware resources consume roughly the same amount of power independent of their utilization rate [6].

To calibrate the *power consumption models*, the command line tool lookbusy[6] is used to stress the CPU. Using lookbusy the hardware resources CPU, HDD and memory can be utilized with a fixed utilization rate. To stress the CPU, lookbusy generates a consecutive CPU utilization in steps of 10 % starting from 0 % to 100 %. Each utilization step lasts for five minutes. After each step the server is kept idle for two minutes. A self written Java tool meanwhile captures the CPU utilization rate and the full server power consumption using IPMI interfaces provided by the servers every second. The IPMI power sensors used in this evaluation showed some ramp-up and ramp-down effects in every utilization step. To avoid these effects, only measurements taken between a ramp-up phase of two minutes and a ramp-down phase of another two minutes are used. This dataset is used to perform a linear regression to construct a *power consumption model* in the form of equation 2. The measurements and the thereof derived *power consumption models* for the AMD- and Intel-based servers are shown in figure 4(a) and figure 4(b).

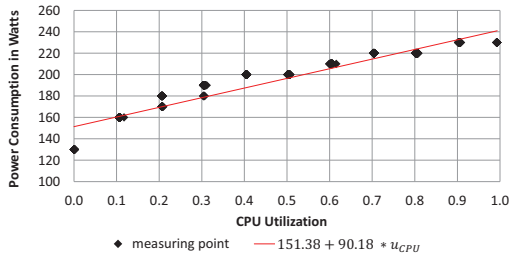
The system model of the SPECjEnterprise benchmark application, complemented with a usage model describing the benchmark driver workload and a resource environment model representing one of the two hardware environments, is hereafter called AMD- or Intel-based model respectively.

[5]http://www.jboss.org/mod_cluster

[6]http://www.devin.com/lookbusy/

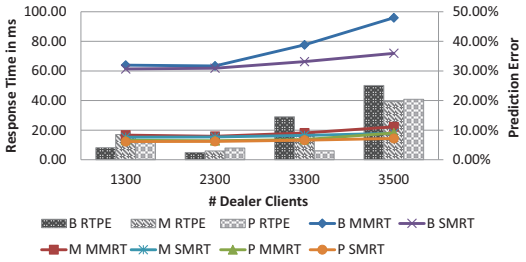


(a) AMD-based server

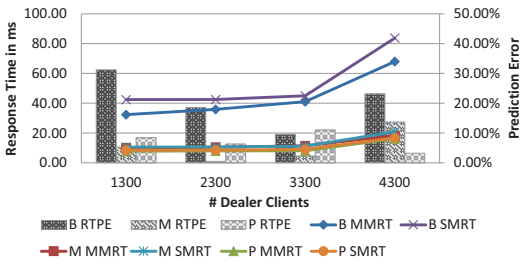


(b) Intel-based server

Figure 4: Power consumption models



(a) AMD-based server



(b) Intel-based server

Figure 5: Measured and simulated response times

3.4 Comparing Measurements & Simulations

In this section, the simulation results of the AMD- and Intel-based models are compared with measurements on the corresponding servers. The comparison is conducted for different load conditions. Four benchmark runs are executed on both SPECjEnterprise2010 server deployments with varying amounts of dealer clients. The number of clients is increased in steps of 1000 from 1300 to 4300. Benchmark runs on the AMD-based server showed that the system cannot handle 4300 concurrent dealer clients. In a benchmark run with 3500 clients the AMD-based server shows a CPU utilization of approximately 86 % which is equal to the utilization of the Intel-based server in a run with 4300 clients. Therefore, the highest load level is reduced from 4300 to 3500 dealer clients for the AMD-based server.

For each load level the AMD- and Intel-based models are used to predict performance and energy consumption for the respective amount of dealer clients. To simulate such high amounts of dealer clients, the event-oriented simulation engine EventSim [19] is used instead of the default process-oriented simulation engine SimuCom [2]. EventSim performs better under such load conditions.

For every load level a benchmark and simulation run of 30 minutes is executed. To avoid side effects during the benchmark and simulation runs, only results during a steady state between a five minute ramp-up and a five minute ramp-down phase are considered in the following.

To evaluate the accuracy of the simulation results, the measured and simulated results for each of the following metrics are compared: CPU utilization, power consumption as well as response time and throughput of the three business transactions. To measure the CPU utilization and the power consumption, the same Java tool is used as for the construction of the *power consumption models* in section 3.3.

The benchmark driver reports measurements for throughput and response time of the three business transactions performed by the dealer clients. However, the reported response times cannot be used for this evaluation since they contain the network overhead between the driver and the SUT whereas the generated model does not include this information. The simulated and measured response times can thus not be compared with each other. To measure comparable response times, a Servlet filter is used to log the response time of each HTTP request executed during a benchmark run. The mean response times of these HTTP requests are used to calculate the mean response times of the three business transactions. The measurement has an influence on the CPU utilization. To reduce this distortion, the Servlet filter is only deployed on one of the six application server instances. The throughput is taken directly from the benchmark driver. It is included in this comparison as using different thread pool configurations could lead to good response time measurements while the throughput is very low. Therefore, the throughput comparison is important to ensure the validity of the evaluation results.

Table 1 and figure 5(a) show the comparison of the measured and simulated results for the AMD-based server and model. Table 1 shows for every load level specified by the number of dealer clients (C) and for every business transaction (T) the Measured Throughput (MT), the Simulated Throughput (ST), the Throughput Prediction Error (TPE), the Measured Mean CPU Utilization (MMCPU), the Simulated Mean CPU Utilization (SMCPU), the CPU Prediction Error (CPUPE), the Measured Mean Power Consumption (MMPC), the Simulated Mean Power Consumption (SMPC) and the Power Consumption Prediction Error (PCPE). Figure 5(a) shows the Measured Mean Response Time (MMRT), the Simulated Mean Response Time (SMRT) and the Response Time Prediction Error (RTPE). Table 2 and figure 5(b) show the same measurement and simulation results for the Intel-based server and model.

The AMD-based model predicts the response times of the business transactions for low (1300 clients) and medium (2300 clients) load conditions with an error below 10 %. Under high load conditions (3300/3500 clients) the error

Table 1: Measured and simulated results for the AMD-based server

C	T	MT	ST	TPE	MMCPU	SMCPU	CPUPE	MMPC	SMPC	PCPE
1300	B	78623	78557	0.08 %	33.37 %	30.04 %	9.97 %	367.55 W	320.26 W	12.87 %
	M	39378	39245	0.34 %						
	P	39259	39135	0.32 %						
2300	B	139328	138383	0.68 %	57.38 %	52.89 %	7.82 %	403.87 W	352.22 W	12.79 %
	M	69685	70186	0.72 %						
	P	69932	69514	0.60 %						
3300	B	200174	199166	0.50 %	82.53 %	76.00 %	7.92 %	433.76 W	384.52 W	11.35 %
	M	99643	99930	0.29 %						
	P	99673	99315	0.36 %						
3500	B	211585	211314	0.13 %	86.10 %	80.59 %	6.40 %	436.47 W	390.95 W	10.43 %
	M	105454	105506	0.05 %						
	P	105708	105557	0.14 %						

Table 2: Measured and simulated results for the Intel-based server

C	T	MT	ST	TPE	MMCPU	SMCPU	CPUPE	MMPC	SMPC	PCPE
1300	B	78502	78333	0.22 %	24.05 %	27.24 %	13.26 %	197.05 W	175.94 W	10.71 %
	M	39464	39376	0.22 %						
	P	39367	39297	0.18 %						
2300	B	139603	138961	0.46 %	45.08 %	48.29 %	7.12 %	220.47 W	194.93 W	11.58 %
	M	69314	69490	0.25 %						
	P	69266	69576	0.45 %						
3300	B	199517	199646	0.06 %	64.86 %	69.34 %	6.92 %	241.67 W	213.91 W	11.49 %
	M	99254	99936	0.69 %						
	P	99890	99355	0.54 %						
4300	B	259548	259591	0.02 %	86.03 %	90.16 %	4.80 %	264.29 W	232.69 W	11.96 %
	M	130239	129641	0.46 %						
	P	129909	129293	0.47 %						

stays below 26 %. The response time prediction of the Intel-based model is slightly less accurate. The response times are mostly predicted with an error below 20 %. Only the prediction of the browse transaction in the case of 1300 clients and at the highest load level with 4300 clients show deviations of 31.23 % and 23.19 % respectively compared to the measurement results.

Both models predict the throughput with an error below 1 %. This low error is caused by the fact that the average think time of a dealer client between two transactions with approximately 9.9 seconds is much higher than the response time of a single transaction. Thus, the prediction errors of the response times only have a low impact on the prediction accuracy of the throughput. The CPU utilization is mostly predicted with an error below 10 % by both PCM models. Only the CPU utilization prediction for the Intel-based server in a setting with 1300 clients has a higher error of 13.26 %. The power consumption of both servers is predicted with an error below 13 %. As the power consumption values are relatively stable during the steady state of all load levels, the energy consumption can be predicted by multiplying the mean power consumption values by time.

This evaluation shows that energy consumption and performance of two deployments of the same enterprise application can be predicted with an accuracy that is acceptable for capacity planning purposes [17]. The approaches to adapt resource profiles to different hardware environments and to predict energy consumption using an extended PCM metamodel could thus be validated. It is therefore technically feasible to realize the resource profile concept.

4. RELATED WORK

The resource profile concept relates to several existing research directions. This section is therefore structured according to different directions that contribute to our work. First, we review existing approaches to support capacity planning for enterprise applications using performance models. Afterwards, research in the area of energy consumption of enterprise applications is presented. Finally, approaches that predict energy consumption and performance of enterprise applications are outlined. The review of related work concludes with approaches to improve the relationships of EAVs, EAUs and EAHs using resource demand data.

Capacity planning using performance models

A model-driven capacity planning tool suite for component- and web service-based applications is proposed by Zhu et al. [25]. The tool suite can be used in early software design phases to support the performance evaluation. It consists of tools to transform existing design models into performance models and benchmark drivers to derive resource demands for the performance models. These performance models and benchmarks can then be used to support capacity planning tasks. Their tooling is intended to be used early in the development process and not as a final capacity planning tooling, as the implementation might have different characteristics as the generated benchmark code.

Liu et al. [14] show how LQN models can be used to support the capacity sizing for EJB applications. In a later work, Liu et al. [15] use LQN models to realize a capacity sizing tool for a business process integration middleware by taking different CPU configurations into account. The au-

thors introduce a way to deal with different hardware environments in the context of LQN models. They implemented a model transformation tool which dynamically constructs LQN models from XML documents representing the application model on the one hand and the hardware configuration on the other hand. However, in their current implementation one is limited to only change the processing speed of the CPU of a server and one is not able to change the hardware environment more radically, for example from a one server deployment to multiple servers.

Tiwari and Nair [23] are using LQNs to predict the performance of two deployments of the same Java EE sample application. Similar to the approach in this work, they show how the SPEC CPU benchmark can be used to adapt a LQN model to a different hardware environment. However, as already discussed in section 2.3, LQN models do not allow to change the workload or the hardware environment without reconstructing the whole model. Their approach is thus less flexible than the one proposed in this work.

Energy consumption of enterprise applications

Capra et al. [6] developed energy benchmarks for enterprise resource planning (ERP), customer relationship management (CRM) and database management system (DBMS) applications. They showed that under the same workload different applications with similar functionality have a significant divergent energy consumption. They concluded that energy efficiency is a quality metric that should be considered when buying or developing new software.

An overview of existing power models and metrics to describe the energy consumption of computer systems can be found in the work of Rivoire et al. [22]. The authors show several metrics that can be used to model the energy consumption of computer systems. This work is thus not focused on the energy consumption of a specific enterprise application but rather on the system level.

Johann et al. [10] propose methods to measure the energy efficiency of software. The authors define energy efficiency as the ratio of useful work done relative to the energy required for performing the work. The authors suggest to calculate the energy efficiency of single methods or components throughout the software development process to create energy efficient applications. However, even though this is a very promising research direction it is challenging to really measure the efficiency if the system on which an application will be deployed has a different power profile than the one on which the application is developed.

Jwo et al. [11] propose an energy consumption model for enterprise applications. The authors calculate the overall energy consumption by multiplying the time a transaction spends on a machine with the machine's mean power consumption. As the time spend on a machine and its power consumption is workload dependent, the resource profile approach is more flexible as it allows to include this perspective.

Combination of performance and energy prediction

One of the few examples that combines energy consumption and performance prediction approaches with a business perspective can be found in the work of Li et al. [13]. The authors propose a sizing methodology for ERP systems which is based on closed queuing networks. Their methodology allows to optimize sizing decisions using multiple dimensions. They allow to perform Total Cost of Ownership (TCO) decisions that include hardware purchasing as well as energy consumption costs for new ERP systems. Unfortunately, their approach is limited to ERP systems with a predefined set of deployment options and is thus not transferable to other types of applications. However, their multi objective optimization (MOO) approach might be an interesting enhancement for the resource profile concept introduced in this work as resource profiles could be used as the input for such a MOO solver.

Relationships between EAV, EAU and EAH

The term resource profile was already used by Brandl et al. [3] in their work on cost accounting for shared IT infrastructures. The authors introduce an approach to associate resource demands to specific IT services (e.g. email) and to store these resource demands for different user types in service-specific vectors (called resource profiles). Using these resource demand vectors they propose an approach to exactly bill the service consumers by the number of users and types of services they are using. Compared to the approach presented in this work, their resource profile concept is mainly intended to be used to allocate costs for existing applications and services more precisely. The approach presented in this work is intended for new applications and services that should be integrated into a data center. However, the data in our resource profile can also be used for the cost accounting approach presented by Brandl et al. [3].

5. CONCLUSION & FUTURE WORK

This work introduced the concept of resource profiles for enterprise applications. Their main purpose is to simplify the integration of new enterprise applications into data centers with given performance requirements. To achieve this goal, resource profiles support the capacity planning process by allowing to evaluate energy consumption and performance for different workloads and hardware environments before an enterprise application is deployed.

The required information to specify a resource profile and the input parameters for the evaluation can be provided by different parties such as EAVs, EAUs and EAHs. This is achieved by leveraging PCM as meta-model to represent resource profiles. This meta-model is enhanced to allow predictions of the energy consumption. Additionally, an approach is presented to adapt these enhanced PCM models to different hardware environments. The evaluation showed that a resource profile for the SPECjEnterprise2010 benchmark application predicts energy consumption and performance for two hardware environments with high accuracy.

Additional case studies are required to validate further use cases outlined in section 2.2. To make resource profiles better applicable in different scenarios, several extensions are required. First of all, the representation of external systems (e.g. CRM or ERP systems) reused by an enterprise application needs to be investigated. As resource profiles are intended to describe a specific enterprise application, approaches to represent external systems as black-box components which can be replaced by EAUs or EAHs need to be introduced.

Enterprise applications are often executed in runtime environments (e.g. Java EE servers) that are available by different vendors. It would be helpful for the capacity planning if these platforms were represented explicitly in a resource profile and could also be changed by EAUs or EAHs.

Further extensions are required in the underlying PCM meta-model to support the representation of varying workloads and to represent memory demands. The meta-model should also be enhanced to allow representations of the power consumption using nonlinear models. Additionally, the simulation engine EventSim needs to be enhanced to support more elements of the PCM meta-model.

Another area of future research is better automation to create resource profiles. Our evaluation shows how a resource profile can be created and adapted for Java EE applications. Other enterprise application frameworks need similar automation. Even though the underlying performance models could be created using different means such as static or dynamic analysis, our future research will focus on better dynamic analysis as these approaches tend to generate more accurate models from a resource demand perspective.

6. REFERENCES

[1] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: A survey. *IEEE Transactions on Software Engineering*, 30(5):295–310, 2004.

[2] S. Becker. *Coupled Model Transformations for QoS Enabled Component-Based Software Design*. Karlsruhe Series on Software Quality. Universitätsverlag Karlsruhe, 2008.

[3] R. Brandl, M. Bichler, and M. Ströbel. Cost accounting for shared it infrastructures. *Wirtschaftsinformatik*, 49(2):83–94, 2007.

[4] A. Brunnert, C. Vögele, A. Danciu, M. Pfaff, M. Mayer, and H. Krcmar. Performance management work. *Business & Information Systems Engineering*, http://dx.doi.org/10.1007/s12599-014-0323-7, 2014.

[5] A. Brunnert, C. Vögele, and H. Krcmar. Automatic performance model generation for java enterprise edition (ee) applications. In *Computer Performance Engineering*, pages 74–88. Springer, 2013.

[6] E. Capra, G. Formenti, C. Francalanci, and S. Gallazzi. The impact of mis software on it energy consumption. In P. M. Alexander, M. Turpin, and J. P. van Deventer, editors, *ECIS*, 2010.

[7] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan. Full-system power analysis and modeling for server environments. In *Workshop on Modeling, Benchmarking, and Simulation*, Boston, Massachusetts, USA, 2006.

[8] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. *Computer Architecture News*, 35(2):13–23, 2007.

[9] J. L. Henning. Spec cpu2006 benchmark descriptions. *Computer Architecture News*, 34(4):1–17, 2006.

[10] T. Johann, M. Dick, S. Naumann, and E. Kern. How to measure energy-efficiency of software: Metrics and measurement results. In *Proceedings of the International Workshop on Green and Sustainable Software*, pages 51–54, Zurich, Switzerland, 2012.

[11] J.-S. Jwo, J.-Y. Wang, C.-H. Huang, S.-J. Two, and H.-C. Hsu. An energy consumption model for enterprise applications. In *Proceedings of the IEEE/ACM International Conference on Green Computing and Communications*, pages 216–219, Washington, DC, USA, 2011. IEEE.

[12] H. Koziolek. Performance evaluation of component-based software systems: A survey. *Performance Evaluation*, 67(8):634–658, 2010.

[13] H. Li, G. Casale, and T. Ellahi. Sla-driven planning and optimization of enterprise applications. In *Proceedings of the First Joint WOSP/SIPEW International Conference on Performance Engineering*, pages 117–128, New York, NY, USA, 2010. ACM.

[14] T.-K. Liu, S. Kumaran, and Z. Luo. Layered queueing models for enterprise javabean applications. In *Proceedings of the IEEE International Conference on Enterprise Distributed Object Computing*, pages 174–178, Washington, DC, USA, 2001. IEEE.

[15] T.-K. Liu, H. Shen, and S. Kumaran. A capacity sizing tool for a business process integration middleware. In *Proceedings of the IEEE International Conference on E-Commerce Technology*, pages 195–202, Washington, DC, USA, 2004. IEEE.

[16] D. A. Menascé and V. A. F. Almeida. *Capacity Planning for Web Services: Metrics, Models, and Methods*. Prentice Hall, Upper Saddle River, New Jersey, 2002.

[17] D. A. Menascé, V. A. F. Almeida, F. Lawrence, W. Dowdy, and L. Dowdy. *Performance by Design: Computer Capacity Planning by Example*. Prentice Hall, Upper Saddle River, New Jersey, 2004.

[18] D. A. Menascé. Load testing, benchmarking, and application performance management for the web. In *Proceedings of the Computer Measurement Group (CMG) Conference*, pages 271–282, Reno, Nevada, USA, 2002. CMG.

[19] P. Merkle and J. Henss. EventSim – an event-driven Palladio software architecture simulator. In *Palladio Days 2011 Proceedings (appeared as technical report)*, Karlsruhe Reports in Informatics ; 2011,32, pages 15–22, Karlsruhe, 2011. KIT, Fakultät für Informatik.

[20] R. Reussner, S. Becker, E. Burger, J. Happe, M. Hauck, A. Koziolek, H. Koziolek, K. Krogmann, and M. Kuperberg. The Palladio Component Model. Technical report, KIT, Fakultät für Informatik, Karlsruhe, 2011.

[21] S. Rivoire, P. Ranganathan, and C. Kozyrakis. A comparison of high-level full-system power models. In *Proceedings of the Conference on Power Aware Computing and Systems*, Berkeley, CA, USA, 2008. USENIX Association.

[22] S. Rivoire, M. Shah, P. Ranganathan, C. Kozyrakis, and J. Meza. Models and metrics to enable energy-efficiency optimizations. *Computer*, 40(12):39–48, 2007.

[23] N. Tiwari and K. Nair. Performance extrapolation that uses industry benchmarks with performance models. In *International Symposium on Peformance Evaluation of Computer and Telecommunication Systems*, pages 301–305, Ottawa, ON, USA, 2010.

[24] M. Woodside, G. Franks, and D. C. Petriu. The future of software performance engineering. In *Future of Software Engineering (FOSE)*, pages 171–187, Washington, DC, USA, 2007. IEEE.

[25] L. Zhu, Y. Liu, N. B. Bui, and I. Gorton. Revel8or: Model driven capacity planning tool suite. In *ICSE*, pages 797–800. IEEE, 2007.