Association for Information Systems
AIS Electronic Library (AISeL)

ECIS 2011 Proceedings

European Conference on Information Systems (ECIS)

Summer 10-6-2011

GENERIC PERFORMANCE PREDICTION FOR ERP AND SOA APPLICATIONS

Daniel Tertilt

Helmut Krcmar

GENERIC PERFORMANCE PREDICTION FOR ERP AND SOA APPLICATIONS

Tertilt, Daniel, fortiss – An-Institut of the Technische Universitaet Muenchen, Guerickestrasse 25, 80805 Muenchen, Germany, tertilt@fortiss.org

Krcmar, Helmut, Technische Universitaet Muenchen, Boltzmannstrasse 3, 85748 Garching, Germany, krcmar@in.tum.de

Abstract

Enterprise systems are business-critical applications, and strongly influence a company's productivity. In contrast to their importance, their performance behaviour and possible bottlenecks are often unknown. This lack of information can be explained by the complexity of the systems itself, as well as by the complexity and specialization of the existing performance prediction tools. These facts make performance prediction expensive, resulting very often in a "we fix it when we see it" mentality, with taking the risk of system unavailability and inefficient assignment of hardware resources.

In order to address the challenges identified above, we developed a performance prediction process to model and simulate the performance behaviour and especially identify performance bottlenecks for SOA applications. In this paper, we present the process and architecture of our approach. To cover a variety of applications the performance is modelled using evolutionary algorithms, while the simulation uses layered queuing networks. Both techniques allow a domain-independent processing. To cope with the resource requirements for delivering prediction results fast, EPPIC automatically acquires cloud resources for performing the modelling and simulation. With its slim user interface EPPIC provides an approach for easy to use performance prediction in a broad application context.

Keywords: Performance, Analysis, Prediction, Modelling, Simulation, SOA, ERP.

1 Introduction

The performance of a software system is very often ignored when designing the system (Menascé, 2002). This can be attributed to the invisibility of most parts of a software system and also of its weak points. Bad performance of for example an enterprise resource planning (ERP) system is not immediately visible and tangible, compared to a many kilometers traffic jam caused by a bridge that is too small. Nevertheless correcting the performance problems afterwards can be just as costly and difficult, as stated by Brebner et al. (2009). Moreover, the complexity of modern software systems makes it hard to understand how the system will perform under a changed load, or even after changes on the soft- or hardware. Existing tools are either tailored for a very special type of application, or they come with a variety of protocols and adapters and hundreds of configuration properties, resulting in the need of expert knowledge to operate them.

Not knowing the performance behaviour of an enterprise system though is a big risk. As enterprise systems are the backbones of many business processes, performance problems can not only block the scalability of these processes, but even bring down a department's or company's whole operational work.

In order to address these challenges, we develop a process and architecture of an integrated performance prediction tool for distributed enterprise applications, especially for enterprise service oriented architectures (SOA, (Dustdar, Gall and Hauswirth, 2003)). We called the tool EPPIC (Evolutionary Performance Prediction in the Cloud). As an exemplary implementation for an SOA we demonstrate the EPPIC process on an ERP system as an SOA service provider, i.e., a way of accessing the ERP system that becomes more and more crucial (Schneider, 2008).

2 Related Work[1]

Becker et al. (2007) introduce the Palladio Component Model (PCM) for predicting the performance of a component-based software system at design time. Kraft et al. (2009) estimate service resource demands based on response time measurement using linear regression and the maximum likelihood estimation, focusing on the analysis of ERP systems. Brebner et al. (2008) focus on the performance prediction of SOA, and perform extended case studies in governmental software development projects. Bögelsack et al. (2008) describe how to use a simulation model for simulating the performance behaviour of complex ERP systems, while Rolia et al. (2009) use Layered Queuing Models (LQM) for modelling ERP performance.

3 Research Design

All the previously mentioned approaches have one thing in common – their application requires extended manual effort and expert knowledge. In contrast to that our research aims on developing an easy to use performance prediction framework. We use a design science approach as defined by Hevner, March, Park and Ram (2004) to answer the following research question.

How can a performance prediction framework be designed using evolutionary algorithms and layered queuing networks to efficiently forecast the performance behaviour of SOA enterprise applications using cloud resources?

[1] Due to the limited space in a research in progress paper, the related work section is strongly shortened. An extended literature review can be requested from the author.

Our design science artefact is the EPPIC tool. In several iterations the tool and its components are developed and evaluated. The evaluation is done by applying the artefact to load-testable enterprise systems and by comparing the predicted performance behaviour with the measured performance data.

4 Prediction Process

Before describing the developed architecture, we give an overview of the performance prediction process. The performance prediction process consists of three steps – measurement, modelling, and simulation. EPPIC builds upon existing measured performance data, so that we will keep the measurement section small and focus on the aspects of modelling and simulation. Performance models are fundamental to predict the scalability of software and hardware systems, either by analytical methods or simulation (Menascé and Almeida, 1998). Following this advice, we develop performance models for each component in the analyzed software system. As we want to support different types of components and different patterns of input data, we use an evolutionary algorithm approach to model the component's performance behaviour. The evolutionary algorithm is used to perform a multi-objective optimization (Zitzler and Thiele, 1999) on the given performance data, resulting in an approximation of the performance behaviour represented by a continuous mathematical formula. The performance models are used for simulating the behaviour of the analyzed system. For simulation we use Layered Queueing Networks (LQN) as defined by Franks et al. (2009).

4.1 Process Step 1: Performance Measurement

In the EPPIC architecture we focus mainly on the performance modelling and simulation. It is assumed that measured performance data of the system components exist, or is obtained by the use of an external tool. In our evaluations we used the tool PEER (Performance Evaluation Cockpit for ERP Systems) developed by Jehle (2010), as this tool provides a suitable way to gather performance data of a software system without having a visible performance impact on the tested system.

4.2 Process Step 2: Performance Modelling

For every service of the SOA, a performance model is created. The performance model is an approximation of the component's response time, based on various input parameters like the request type and size and the number of parallel requests. The performance models are used as input for the simulation, and represent the response time behaviour of a service.

The evolutionary algorithm used for performance modelling consists of a population of individuals competing for a limited resource, in this case simply the number of allowed individuals. After random model initialization, the individuals compete by comparing their fitness value, in this case the negative geometrical distance of the model to the measured performance data. The individual with the better fitness passes its model to the loser, where it is, to a given chance, mutated (Goldberg, 1989), and crossover (Goldberg, 1989) is performed to a given chance by the exchange of a random part of the winner's model by a random part of the loser's model. The mutation of the passed model allows the model to converge towards a maximum in the search space, which means a model approximating well the measured performance data. As this maximum might be a local maximum crossover allows jumping in the search space, which enables the algorithm to leave a local maximum and to jump to a global one. The advantage of an evolutionary algorithm for modelling is that it can be efficiently executed on any set of measured performance data, independent of its structure and size (Gwozdz and Szlachcic, 2009). This allows the creation of performance models even for services with few measured data available (i.e. cost-intensive and/or externally provided services), while the exactness of the model can be strongly increased by the consideration of any kind of available input data. Furthermore the evolutionary algorithm provides first results very fast (dependent on the underlying hardware

resources, as described in the following chapter), while it can continue optimizing the model continuously.

4.3 Process Step 3: Performance Simulation

For simulating the performance behaviour of the system, we use Layered Queuing Networks (LQN). LQN offer flexibility in modelling software entities using the task as its main concept. A task can be either a hardware resource or a software entity. Each task has its own (infinite) queue to store incoming requests until they can be processed. Both, software entities and hardware, can be single- or (infinite) multi-servers depending on the number of requests that can be processed concurrently. A task may provide more than one service; therefore a task can contain different entries. For instance, a database may be modelled as a task offering services like “insert”, “update” and “delete”.

Key advantages of LQNs are, beside others, the natural mapping of LQN to componentized, multi-tiered, and multi-layered enterprise level software stacks and its extensibility to include newly discovered bottleneck resources or devices into existing models (Ufimtsev and Murphy, 2006). The LQN formalism maps very well to the ERP system architecture as it supports modelling of the internal hierarchical and parallel activities (Woodside, 2002). The formalism is very flexible due to the reusable sub models.

5 Architecture

In this chapter we describe the architecture of our performance prediction tool EPPIC for SOA applications. The architectural design of EPPIC is focused on maximum generality to allow the prediction of multiple types of applications. Furthermore the architecture is strongly scalable to allow the prediction of small systems up to very large enterprise systems. Figure 1 depicts the architecture of the EPPIC tool.

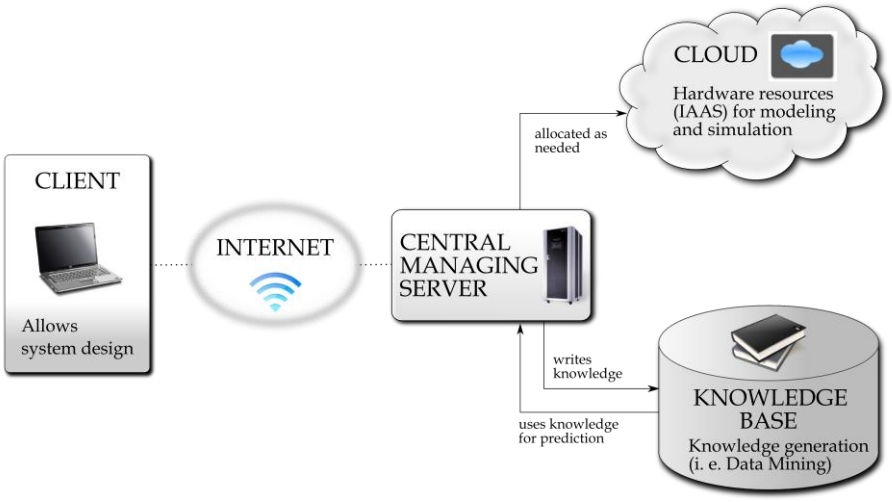

Figure 1. Architecture of the EPPIC Tool

A client application executable on any standard device, such as e.g. standard PCs, laptops and tablet PCs provides the user interface for system design, measured data input, and the prediction result. The central managing server manages the cloud resources as well as the knowledge base. Resources for modelling and simulation are allocated temporarily in an Infrastructure as a Service (IAAS) cloud, and already modelled component performance behaviours and designed systems are stored in the knowledge base. In the following, we will describe all the architecture’s components in detail.

5.1 Client

The client application serves three purposes. First, it is used for configuring the tested system. For this the application provides a drag & drop interface for defining the system components and their relation. For standardization and readability reasons the system design view is oriented at the UML component diagram.

Second, the client interface in this view allows the attachment of measured performance data to system components. This data is used for creating the performance models.

Third, the performance prediction results are shown in the client interface. In the first version of EPPIC we will focus mainly on bottleneck detection. For this, we mark bottlenecks (components which queues run full during simulation) in yellow or red (dependent on the speed the queues get filled), while uncritical components stay green. The interface allows re-runs of the simulation, i.e. after changes on the load behaviour has been performed.

5.2 Central Managing Server

While the client application provides the user interface, the central managing server is the managing backend component. It provides a web service interface for the client applications and manages the cloud resources used for modelling and simulation. Furthermore it manages the knowledge base and decides whether a performance model is taken from there, or newly modelled using the cloud resources.

The central managing server provides a SOAP/HTTP web service interface to the client. Using this interface, the clients can transfer system configurations and request performance prediction.

For the computation-intensive performance modelling and simulation, we use automatically allocatable cloud resources. The central managing server allocates and de-allocates the required cloud resources automatically in a cost-optimized manner. As the chosen cloud resource provider (see also the following section) allows resource allocation on an hourly base, the managing server allocates resources as needed, and keeps them up for multiples of full hours. In this way, an allocated resource can be used for multiple modelling or simulation rounds.

Already modelled components and system structures are stored in the knowledge base. The storing and fetching of the knowledge base objects is done by the central managing server. For identifying a user's systems and components (and also for anonymizing the data in the cloud, as described later on), each model and system design is identified by a unique identifier. The central managing server maintains the mapping between the unique identifier and the user.

5.3 Usage of Cloud Resources

The algorithms used for performance modelling and simulation scale nearly linear with the available hardware resources. For providing fast feedback to the user we need huge CPU and memory resources for short periods of time. As it is not economic to host this amount of hardware resources for just some peaks a day we decided to use the Amazon Elastic Compute Cloud (EC2). The EC2 provides Infrastructure as a Service (IAAS), which allows us to allocate dynamically as many hardware resources as needed. The hardware allocation is done programmatically using the EC2 API, initializing the allocated machines with a predefined basic Linux template which brings the Java runtime as well as the modelling and simulation applications.

The data sent to the cloud for modelling and simulating is anonymized. To identify the correlation between prediction results and systems an identifier is attached to the input data and results. These identifiers are managed by the managing server. Using the anonymization helps to mitigate concerns of transparency and security of the data in the cloud.

5.4 Knowledge Base

Modelling the component performance models and designing the system structure is a time consuming process. To avoid duplicated effort once modelled components and designed systems are stored in the knowledge base. After a component's behaviour is modelled once, or a system is designed, it can be reused for modelling alternative systems, or for the simulation of scenarios like the adding or removing of hardware, increased load behaviour, or hardware and software failure.

6 First Results

As a first step for implementing the presented EPPIC architecture we developed a prototype for the evolutionary algorithm for modelling the performance behaviour of the system's components, called Darwin. First results gathered by applying the prototypical implementation of Darwin on performance data of an industrial ERP system (SAP ERP) won by load testing the work material creation process of the Production Planning Integration Case Study (Weidner, 2006) are very promising. The evolutionary model generation results in performance models that are very close to the real component performance behaviour with errors between 10 to 15 percent. Surprisingly, but well interpretably when looking at the storage of the generated model, the performance of the evolutionary modelling algorithm is strongly correlated to the available memory, and less to the CPU resources. The analysis of this fact showed that the generated models become complex, requiring too much memory when stored as object trees. Further improvements can be made to the performance of the modelling algorithm by analyzing different model storage formats, like for example a string representation.

As already stated in section 3.3 the resource consumption using LQN based approaches differ between the used evaluation technique. When using the Mean Value Analysis (MVA) based solver, the solving time is extremely short, even with very low usage of hardware resources. Evaluation of LQN models using the simulation tool takes a lot of more computing power. As this approach is not based on a mathematical solution the accuracy of the results rises with the number of iterations, which leads to longer runtimes and therefore to a need for more computing power. Nevertheless both evaluation techniques are needed, and with the usage of automatic resource allocation we cope well with the volatile resource consumption. The algorithm to decide which and how much resources are allocated will be one of the big challenges in the EPPIC project.

7 Conclusion and Future Work

The increased accessing of ERP systems as SOA services will allow software performance engineers to merge the approaches for predicting the performance of ERP and SOA applications. When ERP provided in a Cloud gain more attention in research and practice, the approaches will be very similar. With EPPIC we already developed an approach towards this development by creating a unified performance prediction technique for ERP as well as SOA.

In the presented version of the architecture, the knowledge base simply stores a user's component performance models and its designed systems for follow-up usage. The performance of the prediction process might be improved by, when modelling a component's performance, identifying similar, already modelled components, and use their models either as the component's model (potentially with modifications), or at least as initial model for the evolutionary algorithm. Furthermore the case that no measured performance data is available for a system's component can be covered by the knowledge base. To fulfil this goal, a matching algorithm has to be developed that is able to identify similar components.

Next steps will have to validate the EPPIC approach by performing case studies on real enterprise applications using an EPPIC prototype covering all of the described functionality. Further effort will

be focused on the optimization of the evolutionary modelling, as on a complex system under test it requires most of the hardware resources and time.

8 References

Becker, S., Koziolek, H. and Reussner, R. (2007) Model-based performance prediction with the palladio component model, ACM, pp. 54-65.

Bögelsack, A., Jehle, H., Wittges, H., Schmidl, J. and Krcmar, H. (2008) An Approach to Simulate Enterprise Resource Planning Systems, 6th International Workshop on Modelling, Simulation, Verification and Validation of Enterprise Information Systems, MSVVEIS-2008, In conjunction with ICEIS 2008(Eds, Ultes-Nitsche, U., Moldt, D. and Augusto, J. C.) INSTICC PRESS, Barcelona, Spain, pp. 160-169.

Brebner, P., O'Brien, L. and Gray, J. (2009) Performance Modeling Evolving Enterprise Service Oriented Architectures, Joint Working IEEE/IFIP Conference on Software Architecture 2009 & European Conference on Software Architecture 2009Cambridge.

Brebner, P. C. (2008) Performance modeling for service oriented architectures, Companion of the 30th international conference on Software engineeringACM, Leipzig, Germany, pp. 953-954.

Dustdar, S., Gall, H. and Hauswirth, M. (2003) Software-Architekturen für Verteilte Systeme, Springer-Verlag, Berlin, Heidelberg.

Franks, G., Al-Omari, T., Woodside, M., Das, O. and Derisavi, S. (2009) Enhanced Modeling and Solution of Layered Queueing Networks, IEEE Transactions on Software Engineering, 35 (2), pp. 148-161.

Goldberg, D. E. (1989) Genetic algorithms in search, optimization, and machine learning, Addison-Wesley Professional, Upper Saddle River,NJ, USA.

Gwozdz, P. and Szlachcic, E. (2009) An Adaptive Selection Evolutionary Algorithm for the Capacitated Vehicle Routing Problem, Logistics and Industrial Informatics, 2009. LINDI 2009. 2nd International, 10-12 Sept. 2009, pp. 1-6.

Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004) Design Science in Information Systems Research, MIS Quarterly, 28 (1), pp. 77-105.

Jehle, H. (2010) Performance-Messung eines Portalsystems in virtualisierter Umgebung am Fallbeispiel SAP, Lehrstuhl für Wirtschaftsinformatik, München.

Kraft, S., Pacheco-Sanchez, S., Casale, G. and Dawson, S. (2009) Estimating service resource consumption from response time measurements, Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and ToolsICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Pisa, Italy, pp. 1-10.

Menascé, D. A. (2002) Software, performance, or engineering?, Proceedings of the 3rd international workshop on Software and performanceACM, Rome, Italy, pp. 239-242.

Menascé, D. A. and Almeida, V. A. F. (1998) Capacity planning for Web performance: metrics, models, and methods, Prentice Hall.

Rolia, J., Casale, G., Krishnamurthy, D., Dawson, S. and Kraft, S. (2009) Predictive modelling of SAP ERP applications: challenges and solutions, Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and ToolsICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Pisa, Italy, pp. 1-9.

Schneider, T. (2008) SAP-Performanceoptimierung, Galileo Press, Bonn.

Ufimtsev, A. and Murphy, L. (2006) Performance Modeling of a JavaEE Component Application using Layered Queuing Networks: Revised Approach and a Case Study, SAVCBS 2006.

Weidner, S. (2006) Integrations-Fallstudie PP (SAP ECC 5.0).

Woodside, M. (2002) Tutorial Introduction to Layered Modeling of Software Performance.

Zitzler, E. and Thiele, L. (1999) Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, IEEE Transactions on Evolutionary Computation, 3 (4), pp. 257 - 271.