Performance Prediction of Applications using Software Engineering

[1] J. Usha Sreevani, School of Information Technology, VIT University, Tamilnadu, India.
[2] A. Charishma, School of Information Technology, VIT University,Tamilnadu, India.

Abstract:-*Enterprise systems are business-critical applications, and strongly influence a company's productivity. On the other hand, software development techniques like waterfall, iterative and incremental approach etc are the heart of the development process that drive to the success of the system developed. These techniques can be applied in various contexts and for applications and systems too. In contrast to their importance, their performance behavior 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. To overcome the costs that are incurred when tools are used, genetic algorithm based approach has been proposed, but that in turn has its own disadvantages. In order to address the challenges identified above, we developed a performance prediction process to model and simulate the performance behavior and especially identify performance bottlenecks for applications. In this paper, we present the process and architecture of our approach.*

Keywords: ERP,EPPIC,Performance Prediction,Software Engineering.

1.INTRODUCTION:

The performance of a software system is very often ignored when designing the system. 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 behavior 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).

Existing System:

1.INTRODUCTION:

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 behavior. 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 behavior represented by a continuous mathematical formula. The performance models are used for simulating the behavior of the analyzed system. For simulation we use Layered Queueing Networks (LQN) as defined by Franks et al. (2009).

Process Step 1: Performance Measurement

In this 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. Again here we cannot escape the usage of a tool.

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 behavior 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 (Gwozdzand 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.

Process Step 3: Performance Simulation for simulating the performance behavior of the system, Layered Queuing Networks (LQN) are used . 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.

Limitations:

1. Using genetic algorithm makes you wait days for a solution.

2.The existing solution is not much suitable for parallel systems.

3.Fine tuning all the parameters for the GA, like mutation rate, elitism percentage, crossover parameters, fitness normalization/selection parameters, etc, is often just trial and error.

4.The way you communicate your desires to the system is through the fitness function. But GAs will take it literally, with no common sense. You have to be very careful when designing your fitness function.

5.no guarantee of finding global maxima.

Detailed study:

The more we know a system,the more precise the model of a system we may develop,the easier and more successful the performance prediction may be. Performance measurement may be conducted on an existing parallel system to identify current performance bottlenecks,correct them and identify and prevent potential future performance problems. A majority difficulty faced by highly dynamic analysis is efficiently and reliably forming inferences from performance measurements. Gathering data enough to conclusively to verify a performance function in the presence of noise and error can be too expensive to provide a viable basis for performance prediction. The runtime of parallel applications is dependent on many factors. It is important to understand the range of factors that can mainly affect performance , and to predict the performance of a potential execution. As there are many applications that are running at the same time , the performance has to be predicted for many. Hence it will pose a greater difficulty in predicting the performance of the applications. In this paper our approach for predicting the performance of various applications as a whole is described.

2.PROPOSED SYSTEM

In the BSP model,the system is described in three elements:the data modules,the interconnecting network and the synchronizing facility. This model is to bridge a gap between theoretical work and practical considerations. Under this model,the applications are considered as a sequence of computation steps separated by global synchronization. In each computation step,there is certain time unit during which the application receives L/g requests,where g is the communication bandwidth. Given the computation and communication requirements for an algorithm ,this model gives an upper bound on the parameters (L and g) that allow optimal execution of the algorithm. This algorithm is applicable in a variety of techniques and several algorithms can be directly implemented using this model. If L and g are considered for each application separately , the working of the algorithm with these given parameters fetches us the required output that is our performance measurement not accurately but approximately which is far better ,efficient ,easier and less costly means of predicting the performance of applications. Though at the application level this algorithm does not fit in the bulk synchronous framework.

As an alternative method we can also use this approach. Firstly , measured performance data of system components is obtained from using an external tool. Hence each application has its own data. Data needs to be added/updated for new applications. This is the approximate response times of different applications with different data including parallel requests from the application requester. We devise an algorithm to test the applications if they run according to these estimated response times. Moreover when a failure occurs in accessing the application , it is informed through the algorithm itself. By this way, much of the performance prediction can be done escaping heavy costs.

There is another model that extends this BSP model. This is the Log P model which in-turn has four parameters. This model aims at capturing the bottlenecks in parallel systems. It considers communication costs and system parameters. Here sending a small message takes o cycles on the sending processor and L cycles for the communication latency, and o cycles on the receiving processor. Hence the total time needed for a small message is 2o+L cycles. If sending a small message of m bytes requires sending of m/w messages between the nodes,where w is the underlying short message size,then the total time required to send a m bytes message is o+([m/w]-1) x max{g,o} + L + o cycles. Thus the communication performance of a parallel system can be predicted by this mechanism which is more realistic unlike the previous BSP model. Beyond predicting accurately ,the performance of applications, this model is useful in evaluating parallel systems.

3. CONCLUSION:

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. The above mentioned algorithms need to be properly devised in order to obtain good results.

REFERENCES:

[1].Performance prediction 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).

[2]. 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)

[3].Performance Modeling Evolving Enterprise Service Oriented Architectures, Joint Working IEEE/IFIP Conference on Software Architecture 2009 &European Conference on Software Architecture 2009 Cambridge. Brebner, P. C. (2008).

[4]. Performance modeling for service oriented architectures, Companion of the 30th international conference on Software engineering ACM, Leipzig, Germany, pp. 953-954.Dustdar, S., Gall, H. and Hauswirth, M. (2003) Software-Architecture für Verteilte Systems.