Studying Sensitivities of an EJB Performance Model

Catalina M. Lladó
Universitat de les Illes Balears
Departament de Matemàtiques i Informàtica
07071, Palma de Mallorca, Spain
cllado@uib.es

Johannes Lüthi
FHS KufsteinTirol
Studiengang IT & Wirtschaft
Andreas Hofer Straße 7, 6330 Kufstein, Austria
Johannes.Luethi@fh-kufstein.ac.at

Peter G. Harrison
Imperial College
Department of Computing
180 Queens Gate, London SW7 2BZ, UK
pgh@doc.ic.ac.uk

Abstract

Often, at the time when a performance model is built, some of its input parameters are not known exactly. This can be for a variety of reasons, such as the system itself not having been completed, a benchmarking process not yet having been carried out, or that precise estimates for the parameters are very difficult to obtain. In these situations, interval values can be used to express the uncertainties of the input parameters and sensitivity studies of performance models have the potential to show the influence of input parameter uncertainties on the resulting performance measures. In this paper we study the sensitivities of an Enterprise JavaBeans performance model. This model is built using the standard method of model decomposition which gives various submodels. Solutions obtained for these submodels have been adapted to interval parameters and hence interval-based sensitivities are studied in the so-called container submodel and then the whole model. Using these techniques, parameters that require to be characterised more carefully than others can be identified.

Keywords: distributed systems, performance modelling, queueing, enterprise JavaBeans, parameter uncertainties, interval parameters, sensitivity analysis

1 Introduction

Enterprise JavaBeans (EJB) [8] has now become an established (software) architecture for developing, deploying and managing reliable and secure enterprise applications in production environments, e.g. e-commerce, e-science and engineering. It is therefore important to be able to model their performance in order to facilitate good architectural design options and choice of system parameters. Analytical models have been developed to this end in [3, 4], which show good approximations to simulations of a real EJB architecture.

As a robust, generic modelling methodology develops in this way, it becomes important to assess the sensitivity of a system with respect to its parameters. For example, what is the effect on user response time of a 10% increase in the total workload offered to a file server? Such questions can be answered by running a performance model several times with different parameter settings, but this is an expensive and unreliable way to proceed. Performance bounds provide a well-known and popular means for analysing sensitivity. Most often, performance bounds (i.e., intervals for obtained performance measures) are obtained by using simplified assumptions in order to calculate optimistic and pessimistic estimates for the desired performance measures in a (much) more efficient and cost-effective way than using a more exact model.

Performance bounds may also be obtained if bounds of the input parameters are known, i.e., if input parameters of a performance model are characterised by intervals [7]. Using intervals as input parameters may be motivated by existing uncertainties and incomplete information about the parameter values. Furthermore, the relationship between the output intervals obtained for performance measures and the input parameter intervals can be used to study the sensitivity of those performance measures. The interval adaptation of a submodel of our previous EJB architecture was presented in [5] as a concrete case study to compare different splitting algorithms that can be used to approximate interval computations (see for example [6]). In the present work, we extend this adaptation to the whole EJB architecture model of [4].

In section 2, we summarise the modified queueing network model of the complete EJB architecture that we adapt to intervals using the method described in section 3. We present and interpret numerical (interval) predictions of our adapted models in section 4, for the container sub-systems, and in section 5 for the whole EJB architecture. The paper concludes in section 6.

2 EJB Performance Model

Enterprise JavaBeans [8] is an architecture for component-based distributed computing. Enterprise beans are components of distributed, transaction-oriented enterprise applications and EJB servers reduce the complexity of developing middleware by providing automatic support for middleware services such as transactions, security and database connectivity. We focus on the method execution mechanism of the EJB architecture, which is the dominant factor influencing performance. A new form of blocking is the main feature of this mechanism, which is abstracted as a modified queueing network. The development of a queueing network model for the EJB architecture uses the general approximation method of model decomposition, in which queueing models are decomposed into submodels in a hierarchical way. This network is decomposed into a thread-manager node and a collection of *container sub-networks*, so-called since each of them represents a container and its *M* corresponding bean instances, see [3]. The thread-manager node models the scheduling of transactions in a closed, i.e. population-constrained, system. Transactions are held in a 'wait-set' if the system is full and one is only admitted when another transaction leaves. Thus, the population of the system remains constant for relatively long periods compared to individual node service times, and so the decomposition approach works well, as in the multi-access models of the 1970s and 1980s [1, 2]. The throughput functions of the sub-networks determine the service rate functions for the corresponding aggregated nodes in the overall network, (see [4]). Solutions for the overall network are subsequently obtained using load dependent mean value analysis (LD-MVA), see for example [2].

3 Interval Sensitivites of the EJB Performance Model

The sensitivity of a performance measure predicted by a model can be quantified by comparing the relative widths (characterising the degree of uncertainty) of input and output intervals for various parameter locations (mid-points of their intervals).

The adaptation of the EJB performance model presented in [3] was done in two steps. First, the short-circuited submodel consisting of a container server and M aggregated instance servers was adapted to interval parameters m_1 (mean service time of the container server) and μ (service rate of the bean servers). This adaptation process is presented in detail in [5]. Special care has been taken in reducing the number of occurrences of interval parameters as well as in exploitation of monotonicity of intermediate computational steps.

In the second step, the overall model is adapted to interval parameters. The thread manager is modelled as a load independent server and each of the aggregated submodels is represented by a load dependent server. The load dependent service rates correspond to the throughput results of the submodels.

For the overall model solution, LD-MVA is applied, as mentioned in section 2. In [9] it is proved that results from LD-MVA are monotonic with respect to the number of jobs in the network. Furthermore, Suri shows how monotonicity with respect to the service rates can be derived. The overall model is adapted to intervals as follows: using the interval adaptation of the submodel, intervals for the load dependent service rates of the submodel are computed. Monotonicity of LD-MVA is then exploited to compute intervals for the overall results efficiently (without interval splitting). However, due to the overestimation of the submodel results serving as input for LD-MVA, the overall results are also overestimated. Thus, selective interval splitting (see [6]) is applied to obtain overall results with the desired accuracy, i.e., intervals that enclose the desired range sufficiently tightly.

4 Submodel Experiments

Using the adaptation of the submodel to interval parameters described in section 3 (see [5] for details), different sensitivity analysis studies have been carried out with different combinations of parameter values in order to analyse distinct behaviours of the model. The subsequent parameters are however fixed for all the experiments: $M = 6$ bean servers per container, $I = 20$ different bean instances and a population of $N = 10$. The estimates for the service rate of the bean servers μ and the mean service time of the outer server m_1 are subject to uncertainty. Thus, these parameters might be characterised by intervals and are denoted as μ^{IV} and m_1^{IV} respectively.

The sensitivity studies have been done with respect to the position and interval width of the parameters m_1 and μ. The results show an (expected) dual view of the same problem, i.e. the model works in the same way keeping μ fixed and increasing the value of m_1 as keeping m_1 fixed and increasing the value of μ (i.e. decreasing the value of $1/\mu$). This is due to the fact that we are studying a two server submodel.

As a consequence, in this paper we only discuss sensitivities w.r.t. the parameter m_1 (for results and discussion of sensitivities with respect to μ see [4]). More specifically, we discuss sensitivity analysis of throughput w.r.t. the position of m_1. However, many other sentitivity studies have been carried out, as for example sensitivity analysis of throughput w.r.t. the position and relative width of m_1 and sensitivity analysis of equilibrium queue length probability distribution w.r.t. position of m_1 (see [4]).

We now investigate how the position of m_1 influences the width of the throughput intervals. The uncertainty of the input parameters or interval width is kept constant and equal to 5%.

In the first experiment carried out, both m_1 and μ are characterised by intervals. Their values are: $\mu^{IV} = 1/4.1 \pm 5\%$ and $m_1^{IV} = m_1 \pm 5\%$, where $m_1 \in [0.5, 12]$. Fig. 1 illustrates the interval relative width of the throughput results thus obtained, which is computed as: $RW = \frac{ub-lb}{(ub+lb)/2}$, where lb and ub are interval lower and upper bound respectively. The graph shows very little variation (between 0.1 and 0.112) meaning that the interval throughput computation is almost insensitive to the position of m_1, when using both m_1 and μ as interval parameters.

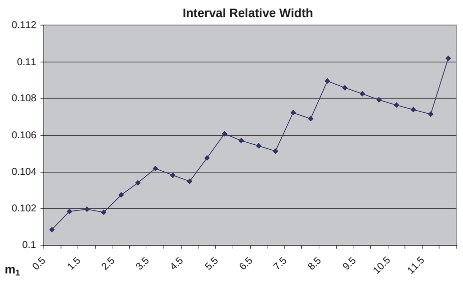

Figure 1. Throughput interval relative width when $m_1 \in [0.5, 12]$.

Looking at detailed throughput results, it can be seen that when $m_1 \simeq 1.7$ the throughput approaches $1/m_1$, indicating that at this point the container server becomes the bottleneck of the subsystem. Therefore, we can infer that, given the already stated parameter values and for $m_1 < 1.7$ the aggregated server – M parallel servers – is the bottleneck of the subsystem. Otherwise (i.e. for $m_1 \geq 1.7$), the container server is the bottleneck and in this case the throughput of the sub-model is equal to $1/m_1$. As a consequence, three different regions can be distinguished depending on the value of m_1. The different regions would be very difficult to guess intuitively. For instance, when considering the system without blocking, we would find that the bottleneck changes around $m_1 \simeq 2/3$ ($6\mu = 1/m_1$), which is very far from $m_1 \simeq 1.7$. We observe that when $m_1 < 1.7$ and the aggregated server is the bottleneck, the throughput still depends on the value of m_1 because it increases as m_1 decreases. On the contrary, when $m_1 \geq 1.7$ the throughput value aproaches $1/m_1$ and is not influenced by the service rate of the parallel servers.

Each one of the different regions (depending on the value of m_1) has been analysed in detail. Fig. 2 illustrates the behaviour of the model in an area where the bottleneck is clearly the aggregated server. It compares interval evaluation results when both m_1 and μ are characterised by intervals against the case when only m_1 is characterised by an interval. As can be seen from the figure, when using only interval values for m_1 the throughput interval obtained is very thin, suggesting that when the container server is not the bottleneck, small variations in the value of m_1 do not influence the subsystem throughput, as expected. On the other hand, when comparing the two-interval parameter case against using only interval values for μ, the interval results for the two cases are almost the same. Hence, in this region, only the characterisation of μ as an interval significantly influences the throughput interval values.

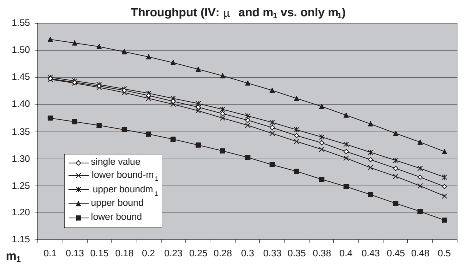

Figure 2. Throughput interval bounds: m_1 and μ against only m_1 characterised by intervals, $m_1 \in [0.1, 0.5]$.

Similar experiments have been performed at the other end of the throughput curve ($m_1 \in [1.6, 2]$), where the container server is the bottleneck of the subsystem (with throughput asymptotically equal to $1/m_1$). These experiments clearly show that the only parameter that influences the throughput interval width – and, moreover, the throughput value – is m_1.

In between these two regions there is a transition zone (when $m_1 \in [0.6, 1.5]$) in which the characterisation of both m_1 and μ as an interval influence the throughput interval width. This zone corresponds to the 'load-balanced' area that surrounds the point where the bottleneck device changes from one server to the other and therefore both service times influence the throughput.

We can conclude that, when only characterising one of the input parameters as an interval (either m_1 or μ), the interval throughput obtained is sensitive to the position of m_1.

5 Overall Model Experiments

Similar sensitivity studies as described in Section 4 for the aggregated submodel have also been performed for the overall model. In this model, an additional service time parameter is introduced, namely the service rate ν for the thread manager server. The value used for the parameter ν is 16 with an uncertainty of $\pm 5\%$. The population chosen for the overall model in this section is $N_{total} = 40$ threads.

Single value results as well as performance measure intervals for three situations have been studied: (a) all three parameters μ, m_1, and ν are characterised by intervals, (b) only the submodel parameters μ and m_1 are characterised by intervals, (c) only ν is an interval. These results show that uncertainty in the parameter ν is not influent since the graphs of interval results where all three parameters are represented as intervals and those where only μ and m_1 are intervals are almost identical. This effect has also been observed in the very thin (i.e., approximately zero width) interval results for the case where only ν is characterised by an interval, which are almost identical to the single value results.

Fig. 3 illustrates once more the fact that for the parameter ranges considered in our experiments, uncertainties in ν need not be taken into account. If only ν is represented by an interval with a relative width of $\pm 5\%$ ($\pm 20\%$), the relative width of the corresponding overall model throughput intervals is always less than 0.04% (0.14%).

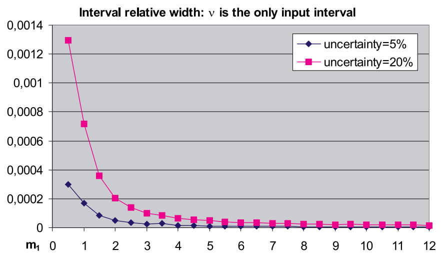

Figure 3. Interval relative width of overall model throughput, $m_1 \in [0.5, 12]$

In analogy with the results presented in Fig. 2 experiments have been carried out to see how representation of only one of the parameters μ, m_1 changes the overall model throughput intervals in comparison to characterising both μ and m_1 by intervals in the region $m_1 \in [0.1, 0.5]$. The qualitative behavior of results such obtained is identical to the corresponding submodel results depicted in Fig. 2. It is interesting to see how the sensitivity results for the submodel directly correspond to almost identical sensitivities in the overall model results.

6 Conclusion

The use of interval arithmetic to evaluate uncertainties in system parmeterisations has been shown to be effective in a highly non-trivial example, namely an EJB architecture model representative of many large distributed component-based client-server applications. The approach clearly identified those parameters in which errors or deliberate changes, for example arising from evolution of the system, have a significant effect on performance – and, conversely, those to which system performance is insensitive.

There are several additional experiments that could be carried out on our EJB model to investigate the sensitivity of other parameters. For example, it would be interesting to consider asymetric systems, in which the container subsystems were non-homogeneous and unequally utilised. A model for this problem appears in [4]. We speculate that sensitivity would be largely determined by the dominating submodel, but dependent on the degree of domination.

More plans for future work point in the direction of integrating interval-based techniques into performance engineering tools. This could be especially useful for software performance models derived from UML (unified modeling language) diagrams which is currently a major theme of general interest.

References

[1] P. G. Harrison and N. M. Patel. *Performance Modelling of Communication Networks and Computer Architectures*. Addison-Wesley, 1993.

[2] E. D. Lazowska, J. Zahorjan, G. S. Graham, and K. C. Sevcik. *Quantitative System Performance – Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1984.

[3] C. M. Lladó and P. G. Harrison. Performance evaluation of an enterprise JavaBean server implementation. In *Proc.* 2^{nd} *Int. Workshop on Software and Performance (WOSP 2000, Sept. 17–20, Ottawa, Canada)*, pages 180–188, Sept. 2000.

[4] C. Lladó Matas. *Performance Evaluation of Enterprise JavaBeans Architectures*. PhD thesis, Imperial College of Science, Technology and Medicine, University of London, 2002.

[5] J. Lüthi and C. M. Lladó. Interval parameters for capturing uncertainties in an EJB performance model. *Performance Evaluation Review*, 29(1):291–300, June 2001.

[6] J. Lüthi and C. M. Lladó. Splitting techniques for interval parameters and their application to performance models. *Performance Evaluation*, 2002. In print.

[7] S. Majumdar and R. Ramadoss. Interval-based performance analysis of computing systems. In *Proc. MASCOTS 1995 (Durham, NC, USA, Jan. 18-20, 1995)*, pages 345–351. IEEE Computer Society Press, January 1995.

[8] S. Microsystems. Enterprise JavaBeans 1.1 Architecture. http://java.sun.com/products/ejb.

[9] R. Suri. A concept of monotonicity and its characterization for closed queueing networks. *Operations Research*, 33:606–624, 1984.