Using Queuing Models for Large System Migration Scenarios – An Industrial Case Study with IBM System z

Robert Vaupel[1], Qais Noorshams[2], Samuel Kounev[2], and Ralf Reussner[2]

[1] IBM R&D GmbH, Böblingen, Germany
vaupel@de.ibm.com
[2] Karlsruhe Institute of Technology, Germany
{noorshams,kounev,reussner}@kit.edu

Abstract. Large IT organizations exchange their computer infrastructure on a regular time basis. When planning such an environment exchange, it is required to explicitly consider the impact on the Quality-of-Service of the applications to avoid violations of Service Level Agreements. In current practice, however, using explicit performance models for such estimations is frequently avoided due to scepticism towards their practical usability and benefits for complex environments. In this paper, we present a real-world case study to demonstrate that a queuing model-based approach can be effectively used to predict performance impact when migrating to a new environment in an industrial context. We first present a general modeling methodology and explain how we apply it for system migration scenarios. Then, we present a real-world industrial case study and show how the performance models can be used. The migration is planned for a System z environment running a large scale banking application. Finally, we validate the performance models after the system has been migrated, evaluate the prediction accuracy, and discuss possible limitations. Overall, the measurements show very high agreement with the prediction results.

Keywords: Business Transactions, Performance, Prediction.

1 Introduction

Large IT companies use their IT systems for a limited period of time. Typically, the main computing infrastructure is exchanged every two to three years and replaced with newer versions of the same computing architecture. These system upgrades are very expensive and require a thorough planning explicitly considering the performance implications on the existing applications and (business) transactions.

When planning such an environment exchange, i.e., a system migration, multiple questions arise as for instance: *i) What capacity is required to maintain comparable Quality-of-Service (QoS)? ii) How does the QoS of the main applications change in the migrated environment? iii) How does the QoS of the main applications change under higher workload intensity?* Moreover, there are multiple aspects that need to be considered posing further challenges: There can be many alternatives to choose from with regard to single processor speed and overall system capacity. Systems with higher processor speed require fewer processors to maintain the same total capacity. The higher speed of the processors might improve the transaction response times at a lower system utilization level, however, the response times might increase more drastically with increasing workload intensity. Furthermore, a system migration might initially target a system with lower capacity with the possibility of a stepwise capacity increase (as supported by many system architectures) if required.

In current practice, however, using explicit performance models to answer typical capacity planning questions is usually avoided. The main obstacle is the still existing scepticism in industry towards the practical usability, benefits, and return-on-investment of classical queuing models in the context of complex real-world scenarios.

To this end, in this paper, we present a real-world case study demonstrating the practicality and effectiveness of using queuing models to predict the performance impact when migrating to a new system environment in an industrial context. More specifically, we first present a general queuing model approach and explain how it is applied in system migration scenarios. Then, we present an industrial case-study and use our approach to plan a migration of a business transaction workload of a banking institute in an IBM System z server environment. We evaluate the queuing model and validate the results with real-world production workloads after the migration has been completed. The evaluation of the approach shows very high agreement with the predictions. Finally, we discuss practical challenges and possible limitations that need to be considered when applying our queuing model approach and under which conditions it can be used.

In summary, the contribution of this paper is a real-world case study in industrial context to show how a queuing model-based approach can be effectively used to project transaction response times for large business environments. Furthermore, we discuss limitations of the modeling approach and identify conditions that need to be considered when using such methodologies in real-world scenarios.

The remainder of this paper is organized as follows: Section 2 gives an overview of our modeling approach. In Section 3, we present our case study. Section 4 discusses limitations when using the modeling approach in complex scenarios. Finally, Section 5 summarizes and concludes the paper.

2 Modeling Approach

To evaluate system migration decisions providing capacity management support, we employ a queuing model-based approach to predict the performance after migrating to another system environment. Our methodology is based on established work [1,2] and comprises the following steps:

1. *System Environment Analysis*:
   We analyze the structure of the environment and, more specifically, we identify the important partitions that need to be analyzed in case of a virtualized environment.

2. *Workload Characterization*:
   We identify and characterize the main workloads for the system migration as well as possible workloads running in parallel affecting the main workloads.
3. *Metrics Measurements and Estimation*:
   We measure performance metrics for both the workload and the system infrastructure, e.g., the workload response time and the system utilization. Furthermore, we estimate the metrics that cannot be measured directly.
4. *Performance Modeling*:
   Finally, we model the transaction processing in a queuing model enabling to project and predict the response time as well as the system utilization of the target system environment depending on the workload.

After migrating the system infrastructure, the results can be evaluated and compared with new measurements to validate the predictions or to refine the approach and include further performance influences for future studies.

2.1 System Environment Analysis

In general, our approach is not limited to specific environments. In this paper, we apply our approach to *System z* mainframe computers. System z environments are state-of-the-art virtualized environments with logically partitioned, shared resources. The main operating system used by large organizations is predominantly the mainframe operating system *z/OS* hosting applications and databases.

Depending on the organization, the partition structure on System z mainframes may vary significantly. On the one hand, many large IT organizations – especially financial institutes – typically use a few partitions to host their applications. Such systems use many processors, large I/O subsystems and big memory environments. On the other hand, the system can be used for up to 60 partitions with few resources allocated for each partition. Such environments are mostly used by organizations hosting systems for other companies. The partition and resource characteristics employed in such environments are very different and need to be identified in the analysis.

2.2 Workload Characterization

Most z/OS-based production systems deploy *batch* and *online transaction processing (OLTP)* workloads. Usually, such workloads run in parallel, where either workload dominates in certain time periods. In some environments, the batch and the OLTP workloads are deployed in separate partitions. For the considered environment, we first identify the main workload and the respective performance characteristics of interest. Since the workload intensity varies over time, we then choose a representative time period in which the workload is running to parameterize the performance model.

The main workload may be a batch workload or an OLTP workload. For a batch workload, the total runtime and throughput (TP) are significant. Single request response times (RT) or the system utilization are usually less important, since the system is usually fully utilized during the batch runtime. For an OLTP workload, the transaction response times and throughput as well as the system utilization are significant. Moreover, the correlation between transaction throughput and system utilization is relevant as an indicator for resource efficiency and CPU cost per transaction.

Since the transaction response times are usually part of the *Service Level Agreements* with end users, our approach is specifically targeted at modeling and predicting the performance of the OLTP workload after system migration.

2.3 Metrics Measurements and Estimation

As previously mentioned, we focus on i) the transaction response time comprised by several components, ii) the transaction throughput describing the workload intensity, and iii) the total system utilization including load generated by workloads running in parallel. These metrics are measured in the existing environment. In general, the transaction response time is comprised of the following components, which are estimated on the existing system by measuring the execution states of the transactions and calculating their proportions of the total response time: i) CPU Processing Time, ii) CPU Wait Time, iii) I/O Data Transfer Time, iv) I/O Wait Time, and v) Other Time (e.g., due to software locking).

2.4 Performance Modeling

In this section, we model the system performance for transactional workloads. More specifically, we model the transaction response times by projecting the CPU processing times from the existing environment to the target environment. We assume that the general CPU processor architecture is the same for the existing and the target system. Furthermore, we assume that the I/O and Other components of the response time are not affected significantly by the migration. The assumption is reasonable for I/O if only the computing system is replaced. For the Other wait times, it is a simplifying assumption that needs to be validated after the migration. Overall, our performance modeling methodology is comprised of *Model Creation*, *Model Calibration*, and *Model Projection*.

Model Creation. To model the CPU service demand, we use an open multi-server queuing model with general interarrival and service time, i.e., a G/G/C queuing model, to cover a wide range of modeling scenarios with arbitrary interarrival and service times, which are obtained by parameterization from real-world measurement data. The model is solved using the *Allen-Cunneen Approximation* [3] shown in Equation (1), where k is the Allen-Cunneen factor, U is the CPU utilization of the system, C is the number of servers (i.e., CPUs) and S is the service time. Furthermore, $P_W(U,C)$ is the probability for waiting in a system with C servers and utilization U expressed by the *Erlang-C formula* [4].

$$W = k \cdot \frac{P_W(U,C)}{C(1-U)} \cdot S, \quad P_W(U,C) = \frac{\frac{(UC)^C}{C!}}{\frac{(UC)^C}{C!} + (1-U) \cdot \sum_{i=0}^{C-1} \frac{(UC)^i}{i!}} \tag{1}$$

The Allen-Cunneen factor k is determined as

$$k = \frac{c_a^2 + c_s^2}{2}, \text{ where} \tag{2}$$

$$c_a = \textit{coefficient of variation} \text{ of the interarrival time distribution,}$$

$$c_s = \textit{coefficient of variation} \text{ of the service time distribution.}$$

Model Calibration. For model calibration, the Allen-Cunneen factor k is either determined based on measurements or estimated. Typically, a value of 1 assuming a M/M/C queuing environment is a reasonable assumption. The CPU processing time S is estimated based on measured transaction response time and observed execution states as

$$S = \frac{\text{CPU Processing States}}{\text{All States}} \cdot RT, \tag{3}$$

where RT is the transaction response time.

Model Projection. To predict the performance in the target environment, we project the transaction processing times of the existing environment to the target environment. Generally, this can be done by determining the relative CPU capacity of the two systems, e.g., using MIPS[1] comparison or CPU benchmarks.

In System z environments, the Large System Performance Reference (LSPR) [5] value is used to determine the capacity of the target system. In LSPR, the capacity of all systems is expressed as a multiple or a fraction of a base system. The capacity also depends on the considered workload and the system layout. IBM performs for each system generation a set of performance benchmarks covering batch workloads and different types of OLTP workloads as well as mixes of them. These benchmarks are used to obtain five performance values that characterize the systems when running different workloads. The values cover aspects from memory to I/O intensive workloads, batch environments and environments with very high transaction volumes. For many real-world environments, the mean or *Average* value of the performance numbers can be used as a representative value. This average value is also used when System z performance is expressed in MIPS. For a given environment, the exact LSPR value can be obtained using a tool called zPCR [6]. Thus, we project the CPU processing time of the existing system S to the CPU processing time in the target system S^* using the relative CPU capacity α as

$$S^* = \phi(S) = \alpha \cdot S, \ \alpha \in \mathbb{R}^+. \tag{4}$$

The system utilization in the target environment U^* is predicted using the *Utilization Law*

$$U^* = S^* \cdot TP, \tag{5}$$

where, in steady state, the arrival rate (or *transaction rate*) equals the throughput TP. The transaction response time in the target environment RT^* is

[1] Million Instructions Per Second

predicted by using the CPU processing time S^* from Equation (4) and the number of CPUs in the target environment C^* and applying Equation (1) to obtain W^*, thus

$$\begin{aligned} RT^* = S^* + k \cdot \frac{P_{W^*}(U^*, C^*)}{C^*(1 - U^*)} \cdot S^* \\ + \text{I/O Processing Time} + \text{I/O Wait Time} \\ + \text{Other Time} \end{aligned} \tag{6}$$

3 Case Study

In this section, we present a real-world migration study for a banking institute performed in 2012 with the following requirements of the installation:

– An existing System z10 with 33 processors should be upgraded to a target System zEC12. The I/O subsystem remains unchanged.
– The initial capacity of the target system should be *below* the existing system to allow incremental capacity upgrades if necessary. Moreover, the initial transaction rates were not expected to be that high to require the full capacity.

To support the migration, appropriate System zEC12 configurations, i.e., number of CPUs, should be identified with the following two prediction objectives:

– Prediction of the *transaction response times* of the main application on the target system as well as their *development upon increase in transaction rates.*
– Prediction of the *utilization of the target system* and, especially, the *increase in utilization* due to the migration to a system with less capacity.

3.1 System Environment Analysis

The existing System z10, i.e., the *base system*, hosts two large partitions that process identical types of workload. For our analysis we summarize the data from both partitions. The operating system on both partitions is z/OS. The data collection is performed with a standard monitoring tool *Resource Measurement Facility* (RMF). The collected data is written to log files, which are managed by the z/OS component *Systems Management Facility* (SMF). The data analysis is performed using a set of tools created by the authors of this paper.

3.2 Workload Characterization

The workload analysis encompasses a three day period. Figure 1 depicts the total workload utilization summarized across both z/OS systems for the System z10. The main OLTP workload is produced by a banking application accessing DB2 databases. In addition, other transaction and batch processing workloads run on the system. We observe that most batch processing takes place during night

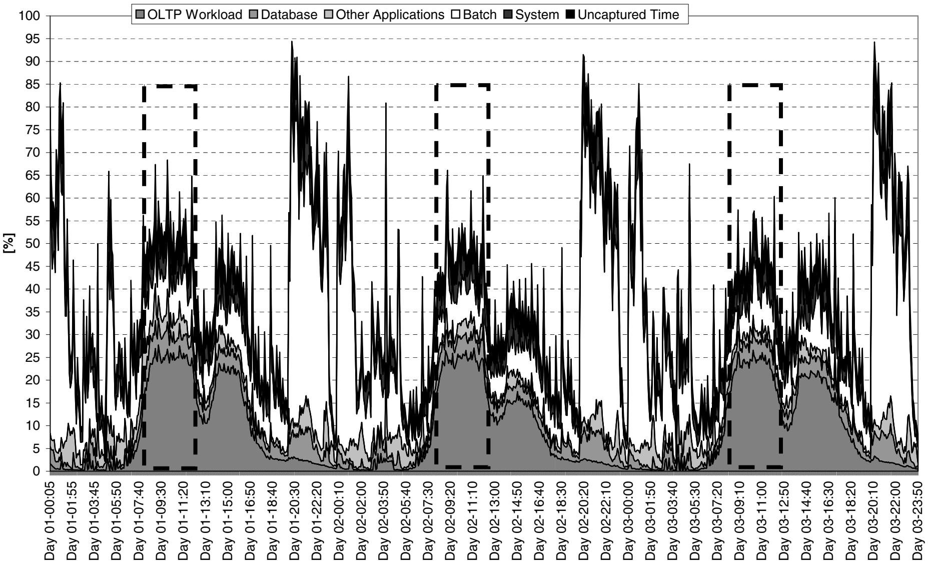

Fig. 1. Total Workload Utilization for Base Environment

Table 1. Measured Input Data of Base System

	Total CPU Utilization	Average Transaction Rate	Average Response Time (s)
Day 1	52%	320	0.096
Day 2	48%	343	0.087
Day 3	45%	336	0.087
Average	48%	333	0.090

time and OLTP during day time. Thus, the batch workload shows the highest utilization periods during night time[2]. For our analysis, we focus on the OLTP workload produced by the banking application and, more specifically, we focus on the period between 8:00 and 12:00 since it is the most critical period having the highest utilization for that workload, cf. dashed lines in Figure 1.

3.3 Metrics Measurements and Estimation

For our analysis, we summarize the system utilization and transaction rates. Furthermore, we calculate the average transaction response time (weighted over the number of transactions per day). Table 1 shows the measured input data for the specified time period for each day and the average values for all three days. We use the total utilization for the system, because we must consider the influence of the other workloads on the OLTP application. Even lower priority work (e.g., a parallel batch workload) shows influence. One main reason is that

[2] The analysis for the batch window has been omitted due to space constraints.

Table 2. Analysis of Execution States of Base System

Execution State	Number of Samples			Corresponding Time Value (s)			
	Day 1	Day 2	Day 3	Day 1	Day 2	Day 3	Average
CPU Using	10145	9919	9552	0.029	0.026	0.026	0.027
CPU Wait	2350	2251	2127	0.007	0.006	0.006	0.006
I/O Using	11248	10862	10422	0.032	0.029	0.028	0.030
I/O Wait	734	702	618	0.002	0.002	0.002	0.002
Other	9306	9302	9292	0.026	0.025	0.025	0.025

two partitions are used and the workloads running in different partitions are equally prioritized. Another reason is that the other workloads use the same cache structures and influence the execution of the considered workload.

The information shown in Table 1 can be measured directly. To obtain the CPU processing time for our model, we use execution state samples to apportion the measured response time. RMF samples execution states of all workloads in the system. These samples are taken every second and detect whether an execution unit is i) using CPU, ii) waiting on CPU, iii) using I/O, iv) waiting on I/O, or v) whether the execution unit is in a state not known to the operating system, e.g., waiting on a database lock. We summarize the samples of the OLTP workload and use them to apportion the response time so that we can calculate the CPU processing time. Table 2 shows the sample breakdown for the time frame from 08:00 to 12:00 of the three days being analyzed.

We will also use the sample states of Table 2 later when we compare the results of our model with data from the target system. We then evaluate whether our assumption that the influence of the I/O subsystem has not changed and that the I/O load is the same is correct. The same applies for the *Other* samples.

3.4 Performance Modeling

Model Creation, Calibration, and Projection. The next step is to select the possible target system configurations. Thus, we calibrate the model with the results from the previous section and project it using the relative capacity of the base and target systems. The relative capacity of each possible target system is taken from LSPR as described in Section 2.4. For our analysis we use the *Average* value, which applies to most installations. We also performed the analysis with the values obtained from the zPCR tool, which provide a slightly more accurate relative capacity, but in order to simplify the study we omit this step. Also, the results showed no significant difference.

Our base system has 33 processors. The target system is supposed to have less capacity and we compare five zEC12 that provide from 85% of the base system capacity up to a slightly higher capacity. The target system configurations have between 14 and 18 processors. The single processor speed of the target systems is nearly twice as fast as for the base system. As described in Section 2.4, for our model we use the number of processors and the relative capacity of the base and target system configurations. In addition, initially we assume an Allen-Cunneen

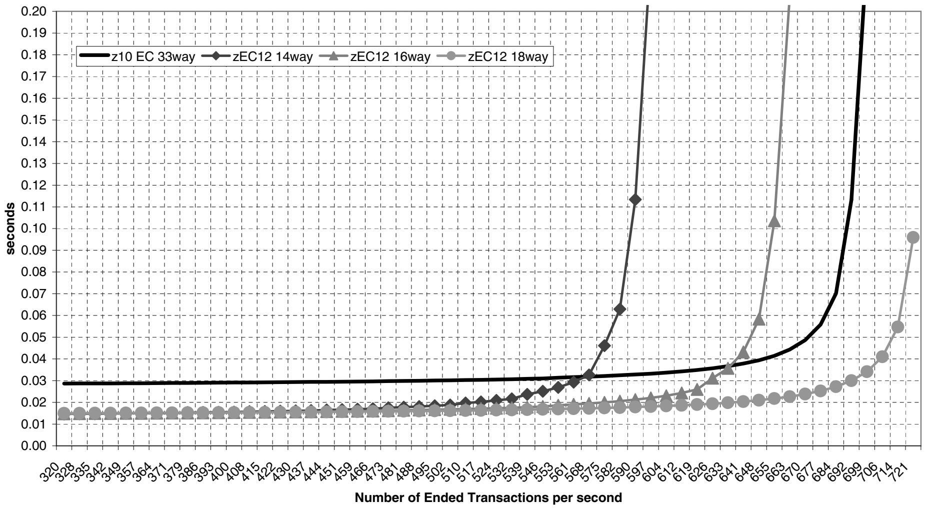

Fig. 2. Change in CPU Response Time

factor of one and later modify the model explicitly with factors of two and four to provide different estimations.

Prediction Results. We model the base system and, as example, three of the possible target system with 14, 16, and 18 processors. The interesting question is how the CPU processing time will change when the transaction rate is increased and how many transactions can be processed before the capacity of any of the target systems is exceeded. Figure 2 depicts the change in CPU processing time for the base system and the three possible target system configurations. On the z10 base system, 320 to 350 transactions per second are processed on average. With the same rate, no negative impact can be expected on any of the target systems, because the overall utilization on the base system is well below 60% as shown in Figure 1. Furthermore, Figure 1 shows that even peak utilizations are always below 70% during the main online processing time from 08:00 to 12:00.

More interesting is the question how many transactions can be processed before the CPU response time shows a significant increase. We define the threshold for a significant increase as two times the measured or projected CPU processing time. We choose two times, because this means that the CPU wait time at this point equals approximately the CPU processing time. At this point, we can expect that a slight change on the system can cause significant queuing and a high disturbance of transaction response times. We also take into account that the total system utilization includes other workloads being processed on the system, which may increase as well.

The base system is able to process at least 680 transactions per second before the CPU wait time has the same magnitude as the CPU processing time. The target system with 14 processors will be able to process 570 transactions per

Table 3. Maximum Capacity in Number of Transactions

Allen Cunneen Factor	z10 33 CPUs	zEC12 14 CPUs	zEC12 16 CPUs	zEC12 18 CPUs
1	680	570	630	690
2	660	530	610	665
4	605	500	550	610

Table 4. Measurement Data for New System with 14 Processors

	Total CPU Utilization	Average Transaction Rate	Average Response Time	Workload CPU Processing Time
Day 4	59%	370	0.075	0.014
Day 5	59%	364	0.093	0.017
Day 6	63%	382	0.075	0.014
Average Measured	60%	372	0.081	0.015
Modeling Results	62%	—	0.080	0.015

Table 5. Analysis of Execution States of New System

	Number of Samples			Corresponding Time Value			
State	Day 4	Day 5	Day 6	Day 4	Day 5	Day 6	Average
CPU Using	5114	5249	5462	0.014	0.017	0.014	0.015
CPU Wait	787	870	864	0.002	0.003	0.002	0.002
I/O Using	12328	12558	13208	0.033	0.041	0.034	0.036
I/O Wait	580	589	621	0.002	0.002	0.002	0.002
Other	9271	9469	9460	0.025	0.031	0.024	0.027

second. Table 3 shows the influence of different interarrival and service time distributions on the maximum number of transactions that can be processed on the considered systems.

The utilization law shown in Equation (5) allows us to predict the change of the total CPU Utilization on the target system when the number of transaction increases. When we use the results in the first row of Table 3 assuming an exponentially distributed transaction rate, we can determine that the maximum number of transactions corresponds to a system utilization of around 95%.

Finally, based on the modeling results we can recommend that for the OLTP workload, a target system with 14 CPUs should replace the existing system. We previously mentioned that another analysis was performed to estimate the total execution time of the batch workloads running during night time (cf. Figure 1). This analysis also suggested that a 14 CPU system was initially sufficient to accommodate the workload.

3.5 Evaluation

We took another set of measurements for a three day period after replacing the 33 processor system with the newer 14 processor zEC12 system. The measurement results are now compared to the modeling results in order to evaluate whether the model predictions were accurate. The measurement were taken for the same time period from 08:00 to 12:00 and summarized in Table 4.

Table 4 also shows the modeling results when we assume around 370 transactions for the new system. A comparison of the measurement data between Table 1 and Table 4 shows that around 11% more transactions were processed on the new system for our evaluation period than on the previous z10 system. Based on the utilization law we can determine that 360 to 380 processed transactions will cause a CPU utilization between 61% and 63%. Our measurements show a CPU utilization between 59% and 63%. Figure 2 shows that for 360 to 380 transactions the CPU processing time is around 0.015s, which also agrees with the measurements.

Finally, Table 5 depicts the processing states on the new system, which we compare with Table 2 from the previous system. We observe that the number of *Other* states has not changed, which means that the processing time for non-OS-related resources has not changed. I/O processing is slightly higher by 13%, however, we also observed 11% more transactions.

4 Discussion

Next, we discuss encountered caveats and practical challenges when using the proposed modeling and analysis approach.

CPU Parking. When we take a look at Table 2 and Table 5, we observe that the CPU wait samples and CPU wait times are higher for the 33 processor system compared to the 14 processor system. This is surprising and in general the times look too high for systems that are utilized by less than 70%. It is also not possible to model these waiting times in Table 2 with the existing queuing formula readily. In fact, the relatively high CPU queuing times result from an optimization for System z. The queuing formula assumes that the processors are always active and ready to process work, however, in the real environment, this is not the case. Many processors are placed in a parking state especially when the system is not too highly utilized. As a result, the system is optimized to reduce cache conflicts and, thus, the CPU processing time decreases and exhibits lower variability. The CPU wait time, however, is slightly higher, because the CPU dispatcher queues are longer than expected on a highly parallel system. In the end, this optimization provides much better throughput, however, the effect diminishes when the system utilization approaches 90% and can be ignored for our purposes.

Secondary Workload. It is difficult to use the model for subordinated workloads that typically run with lower priorities than the main workload. Especially when the transaction rates of the subordinate workloads are low compared to the main workload, the results may become questionable.

Virtualization. Another limitation arises from the environment setup. Our case study showed a fairly simple virtualized environment consisting of two partitions running identical workloads. This is not always the case. Especially many small partitions that process different types of workloads can be very disturbing and cause that no reliable results can be derived from the queuing model.

Clusters. In our case study, we used the methodology to replace a single hardware system. In many large IT installations, the target workload does not only run on a single system, but is spread across a cluster of systems. There are various difficulties arising from clusters to predict transaction response times and utilization of the systems. The most disturbing factors are that the workload distribution is often unequal between the systems and that the systems are configured differently. A different configuration means both that the hardware can be different, for example, that the systems have different number of processors, as well as that the number of partitions executed on the systems is different. Such influencing factors can cause inaccuracies when applying the modeling approach. For such environments, the queuing model needs to be extended to a queuing network modeling both the workload scheduler and the cluster systems.

5 Conclusion

We presented an industrial migration case study for a banking institute in a real-world environment based on IBM System z server technology. Our general goal in this paper was to show how a queuing model-based approach can be effectively used in a complex state-of-the-art real-world context.

In the study, an existing System z10 should be replaced with a newer System zEC12 model. Our approach was used to determine the appropriate number of processors to support the main OLTP workload even during peak periods. We used an open multi-server queuing model with general interarrival and service time distributions calibrating the service times with measurements on the existing system. The service times were projected to the new system using relative capacity information to predict the workload performance in the new environment. With this model, the recommended number of processors was determined. After the system migration, the prediction accuracy was evaluated by comparing the model predictions against measurements with a real-world production workload. Both the average response time and the total system utilization exhibited very high agreement with the predictions.

Finally, we discussed practical challenges and the conditions under which the queuing model would be inaccurate or require a more fine-grained extension to provide reliable predictions. Such challenges typically arise when there are system-specific optimizations, e.g., CPU parking, when the workload is running under low priority, or when the environment is highly distributed and heterogeneous, e.g., in large-scale virtualized and cluster environments.

Acknowledgements. This work was partially supported by the German Research Foundation (DFG) under grant No. RE 1674/5-1 and KO 3445/6-1.

References

1. Menascé, D., Almeida, V., Dowdy, L., Dowdy, L.: Performance by Design: Computer Capacity Planning by Example. Prentice Hall science explorer. Prentice Hall (2004)
2. Bolch, G., Greiner, S., de Meer, H., Trivedi, K.: Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications. Wiley (2006)
3. Allen, A.O.: Probability, Statistics, and Queueing Theory with Computer Science Applications. Academic Press (September 1978)
4. Kleinrock, L.: Queueing Systems: Theory. In: Queueing Systems. Wiley (1975)
5. IBM: Large Systems Performance Reference for IBM System z, `https://www-304.ibm.com/servers/resourcelink/lib03060.nsf/pages/lsprindex`
6. Shaw, J., Walsh, K.: J.F.: zPCR, IBM's Processor Capacity Reference. IBM, `http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS1381`