PELE - An MVA-based Performance Evaluation Methodology for Computer System Migration

Lei Zhang*[1], Issam Al-Azzoni[2], and Douglas G.Down†[1]

[1]Department of Computing and Software, McMaster University
[2]Department of Software Engineering, King Saud Universit

Abstract

Advances in technology and economic pressure have forced many organizations to consider the migration of their legacy systems to newer platforms. Legacy systems typically provide mission critical services vital for an organization's business needs. These systems are usually very large and highly complex with little or no documentation. As time passes, fewer people can understand and maintain these systems. While several techniques exist to verify the functionality of the migrated system, the literature is still lacking methods to effectively assess the performance impact of system migration. In this paper, we propose a new method designed specifically to address performance evaluation in system migration projects. The new method uses analytic queueing models and incorporates Mean Value Analysis (MVA) techniques for model validation and resource demand (or service rate) mapping for performance evaluation and capacity planning.

1 Introduction

Legacy systems and technologies continue to play a significant role in managing today's information systems. Many large organizations are still using

*Email address: zhangl64@mcmaster.ca
†Email address: downd@mcmaster.ca

legacy applications or technologies to provide mission critical services. McKendrick's survey found that nearly half of existing IT applications are based on legacy technologies, and billions of dollars are spent annually on maintaining and supporting these legacy systems [20].

Many legacy systems are becoming out-of-date, and most of them lack documentation or other means that support maintainability. After many years of evolution, these systems also become hard to understand and maintain. The high cost of maintenance is compounded by the fact that there is insufficient documentation to track the development or updates of the legacy systems, or there is a critical shortage of skilled developers who understand and thus can maintain these legacy systems. However, since these systems are vital for an organization's business needs, they can not simply be discontinued. In light of these factors, many organizations have opted for legacy system migration. Selected parts of the legacy system are migrated to modern platforms. In the migration of a legacy system, one or more of the following components are converted into newer, more modern technologies: the hardware platforms, the operating system, languages, data and databases. For example, data can be migrated from a flat file system to a database management system or from one database to another. Another example is the migration from a single processor computer to a multi-core processor platform. Several tools and strategies to assist the migration of legacy systems are presented in De Lucia *et al.* [13], Sahraoui [26], Teppe [27], and Zou [33].

The migration of a legacy system without any pre-evaluation often leads to high risks for organizations. Legacy systems are usually very large (sometimes with more than one million lines of code) and performance sensitive. Furthermore, these systems are highly complex with little or no documentation. The upgrade of such critical systems without some advance performance evaluation is not acceptable.

Carrying out a performance analysis study during the migration of a legacy system is useful and necessary for commercial organizations. It is important not only for managers, but also for developers and users. Traditional capacity planning and performance evaluation methods do not address some particularities for legacy system migration. For example, we note that performance evaluation for a legacy system must consider two systems: the legacy and target systems. In addition, stringent deadlines are typical in migration projects.

To analyze performance of system migration efficiently, we presented a method named PELE[1] in [1]. In PELE, a system is modeled as a parameterized queueing network model which is solved using *Mean Value Analysis* (MVA) (Reiser and Lavenberg [25]). In [1], we applied PELE with a multi-resource queueing network model and a single class MVA algorithm to evaluate performance in the migration of a standard web application, and demonstrated that PELE can be a very efficient method to address performance concerns during software migration.

We believe PELE could work for a variety of computer system migrations using the multi-resource queueing model and single class MVA algorithm. However, our initial work left us with two important issues to consider. First of all, we only simulated a scenario where the target system is fully operational in [1]. In other words, the production workload could be executed on the target system. In practice, the target system may be only partially operational, meaning that the production workload is not measurable on the target system, however, some non-production workload is executable. In this paper, we propose a solution for mapping the resource demands from a fully operational legacy system to a partially operational target system. We parameterize the performance model for the target system using estimated resource demands. Secondly, there are systems that cannot be modeled as a multi-resource queueing model. Characteristics of those systems include: simultaneous resource possession, blocking, high service time variability, *etc.* (see Chapter 15 in Menasce *et al.* [21]). In this paper, we consider a system where locking transactions occur in a database server, and as a result the assumptions of a *product-form* solution are violated. Thus, our previous performance model and the MVA algorithm are not applicable. Consequently, we find an approximate solution using the *Flow-Equivalent Server* (FES) method from Chandy *et al.* [6] to replace the non-product-form network with a single server, and then adapt PELE to address such cases. In summary, we hope the case studies of PELE in this paper not only serve to increase the utility of our methodology, but provide a guide for those who wish to plug in different performance models (appropriate to the application at hand) to PELE.

The organization of the paper is as follows. Section 2.1 discusses the limitations of existing methodologies for performance evaluation of legacy system migration. Section 2.2 proposes our PELE methodology. In Section 3.1, we present a case study of PELE using a calculation based mapping method

[1]PELE stands for Performance Evaluation method for LEgacy systems

to predict the resource demands of the target system, and then use a single class MVA algorithm to solve the performance model. In Section 3.2, we present a case study of PELE using a non-product-form performance model with the FES method and MVA algorithm. Section 4 gives a summary of related literature. Section 5 concludes the paper and discusses future work.

2 Methodologies for Performance Evaluation of Legacy System Migration

2.1 The CAPPLES Methodology

A methodology called CAPPLES[2] (da Silva *et al.* [10, 11, 12]) is to our knowledge the only method developed specifically for the performance evaluation of target systems during the migration of legacy systems. CAPPLES uses a simulation approach, and outlines several novel strategies. One example is a strategy for the characterization of a synthetic workload for the target system. Another novel idea of CAPPLES is the concept of services, which are abstractions of their subtasks (*i.e.*, transactions) where actions and user interactions occur.

While CAPPLES is an attractive approach, we believe that it suffers from three significant limitations:

1. To model the target system, CAPPLES uses a simulation approach, which may be particularly expensive to develop, validate and run. This can be problematic given the tight deadlines typically present in a migration project. Furthermore, compared with queueing models, developing a simulation model requires detailed information about the behaviours of the transactions as well as underlying probability distributions, which may be difficult to obtain in a large system migration scenario.

2. CAPPLES is primarily designed to predict performance of the target system. In CAPPLES, there is no performance comparison between the legacy and target systems. Therefore, it does not model the legacy system. However, modeling the legacy system could be helpful in the

[2]CAPPLES stands for CApacity Planning and Performance analysis method for the migration of LEgacy Systems

validation process. The model of the legacy system could be used to model the target system for performance prediction, or as a basis for performance comparison between the legacy and target systems.

3. In CAPPLES, relevant services need to be identified and mapped from the legacy system to the target system. Therefore, the mapping step could be very time consuming and complicated. A better approach may be to skip the step of determining relevant services and instead, map the performance parameters (*e.g.*, resource demands, or service rates) directly based on the measurements of the legacy and target systems.

2.2 The PELE Methodology

As an alternative to CAPPLES, we present a new methodology called PELE. In PELE, a system is modeled as a queueing network which is solved using an MVA-based solution. Compared with the explicit solution of Markovian models which can require extensive computations and a large amount of information, the MVA algorithm focuses on performance metrics that are mean values, and requires minimal input parameters. This natural advantage of MVA is leveraged to make PELE an effective method for performance evaluation during system migration.

In PELE, input parameters of the queueing network model are obtained by analyzing the performance measurements collected from direct observation of the system. Typical input parameters include service rates, resource demands, and workload intensities (arrival rates, or think times). The parameterized model is solved using an MVA-based solution technique to obtain the mean response time or the throughput. The model can then be validated by comparing the mean response time or the throughput calculated by the model to the values measured on the actual system. Also, the model can be used to predict performance of the system under different scenarios and to compare the performance of different systems, *e.g.*, the legacy and target systems.

The PELE methodology is composed of eight steps:

1. **Characterization of the legacy system's workload.** In a migration project, it is important to describe the legacy system's workload, as in the following Steps 2 and 3. The required information, including transaction log files, could be supplied by communication with the

organization to characterize the workload of the legacy system. The workload consists of all transactions processed by the legacy system during an observation interval. The observation interval should be long enough to represent the actual system's workload. A discussion on how to construct a representative workload can be found in Weyuker and Vokolos [30].

2. **Measurement of the legacy system.** Two kinds of performance parameters are derived from the legacy system's workload characterization of Step 1: intensity parameters, and resource demand (or its inverse: service rate) parameters (see Menascé *et al.* [21]). The intensity (arrival rates, or think time) parameters provide a measure of the load placed on the system. The resource demand (*e.g.*, CPU, memory, disk, *etc.*) parameters provide a measure of the total amount of service time required by a transaction at a given resource. The measurement data presented in the transaction log files are transformed into input parameters for the model in Step 4.

3. **Identification of the relevant transactions in the legacy system.** Since we cannot assume the relevant transactions in the legacy system do not have any changes in the target system, it is necessary to identify the relevant transactions in the legacy system (later the target system too) in order to map the performance parameters in Step 5.

4. **Construction of a model for the legacy system. The model is analyzed and validated.** In this step, the resource demands (or service rates) and the intensity parameters of the relevant workload in the legacy system are ready. As a result, an appropriate queueing model is used to develop a performance model of the legacy system. The parameterized model is solved using an MVA-based solution technique to obtain the mean response time and/or the throughput. The model is then validated by comparing the mean response time and/or the throughput calculated by the model to the same quantities measured on the legacy system. As a rule of thumb, response time errors within 20% are considered acceptable (see Chapter 4 in [21]).

5. **Identify the relevant transactions and map the service demands to parameterize the model for the target system.** In this step, the model of the target system is parameterized. This is done

by mapping the resource demands or service rates of the legacy system workload into the target system. For example, consider a system in which data is migrated from a flat system into a relational database. The use of the new database system will impact how the resources are utilized. Thus, the new resource demands need to be computed. We focus on partially operational systems, and find that the resource demands (or their inverse - the service rates) may be mapped based on the ratio of corresponding parameters for the legacy and target systems (details in Section 3.2.2).

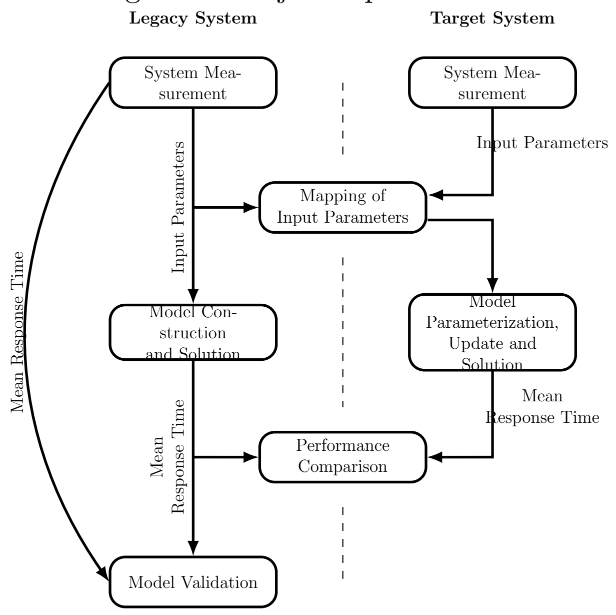
6. **Analyzing the model of the target system.** The parameterized model for the target system is solved using an MVA-based solution technique to obtain the mean response time, and also the throughput if necessary. Such a model may be verified after the analysis, for example, we may need to change the queueing network model from single class to multiple class, or we may need to change a resource model from load-independent to load-dependent.

7. **Comparing the performance of the legacy and target systems.** The mean response time or the throughput calculated from the model of the legacy system is compared with the corresponding values calculated from the target system model. We note that the mean response time of the target system may not be better than the legacy system. In some cases, upgrades may not lead to better performance, which makes the prediction even more crucial.

8. **Improving the migration.** In the final step, we have an overview of the performance on both the legacy and target systems. During the performance evaluation, the bottlenecks on both the legacy and target systems are identified, and various options could be evaluated to improve the performance, such as adding additional resources, or modifying demands of transactions, to give just two examples. A positive perspective of performance on the target system may lead to a confident system migration, while a negative perspective can also provide information to allow designs to be revised as necessary.

The following outlines how our PELE methodology addresses the limitations of CAPPLES:

Figure 1: Major steps of PELE



1. PELE uses MVA-based algorithms and does not require detailed simulation models. MVA is a simple recursion and several MVA-based solution packages are available (see Bolch *et al.* [4]). MVA does not require detailed distributional information. Also, by using MVA, generic tools can be developed for the performance evaluation of legacy systems.

2. In PELE, models for both the legacy and target systems are constructed and validated. This enables a performance comparison between both systems before the production workload is executable on the target system. In addition, the model of the legacy system could be reusable for the model of the target system, or become the basis model for the target system.

3. In CAPPLES, the mapping is not straightforward since simulation is used and hence the underlying distributions need to be mapped. In PELE, resource demands or service rates for services on the legacy system could be mapped into the target system. This is based on the measurements of the legacy and target systems. We note that using a queueing network model simplifies the mapping. In most cases, mapping averages (in the case of PELE) is easier to do than mapping

distributions (in the case of CAPPLES).

Figure 1 outlines the major steps of PELE. As we already discussed in Step 5 of PELE, we focus on partially operational target systems, in which the input parameter mapping step is carried out based on the measurements of both the legacy and target systems to parameterize the model for the target system. Note that if we were unable to obtain any measurements from the target system, the resource demands or service rates could potentially be estimated based on some well known information from similar situations. For instance, consider the database migration example. The use of the relational database can be shown using empirical data to speed up the query services, for example, by a factor of two [14]. Another example is Intel's Hyper-Threading Technology (HTT) which could gain a speedup of up to 30% with optimized programs [5], in other words, we may expect the average CPU demand on an HTT enabled and optimized platform to be 30% smaller than that on an HTT disabled platform.

3 Case Studies

In a previous case study in [1], we presented a scenario of MySQL [24] database server migration. We used the TPC-W [29] benchmark to generate workload, a product-form queueing network to model the system, and a single class MVA algorithm to solve the model. The queueing model has two resources, the CPU and disk. The model is parameterized by the resource demands, and the resource demands are calculated by the Service Demand Law (the Service Demand Law is an easy way to obtain the service demand of a resource by dividing the utilization of the resource by the system throughput, see Chapter 3 in Menasce *et al.* [21]).

In the rest of this section, we will use the notation given in Table 1.

3.1 TPC-W

In the first case study, we adopt the same queueing model and MVA algorithm as in [1]. We simulate a scenario where the target system is partially operational, and we would like to demonstrate how to map the resource demands of the CPU and the disk from the legacy to the target system.

Table 1: Notation

U_{CPU}	utilization of CPU
U_{Disk}	utilization of disk
D_{CPU}	resource demand of CPU
D_{Disk}	resource demand of disk
R	measured average response time
MRT	predicted mean response time
X	throughput
μ	service rate
Z	average think time

3.1.1 Mapping Method

We use the following notation to represent the parameters in this mapping step:

1. The non-production workload on the legacy system - A.

2. The production workload on the target system - B.

3. The CPU resource demands of A on the legacy and target systems respectively - $D_{A,CPU}$ and $D'_{A,CPU}$, respectively.

4. The disk resource demands of A on the legacy and target systems respectively - $D_{A,Disk}$ and $D'_{A,Disk}$, respectively.

5. The CPU resource demands of B on the legacy and target systems respectively - $D_{B,CPU}$ and $D'_{B,CPU}$, respectively.

6. The disk resource demands of B on the legacy and target systems respectively - $D_{B,Disk}$ and $D'_{B,Disk}$, respectively.

Let us assume that the resource demands D'_B for the target system are unknown. We assume that the resource demands only depend on the workload, in other words, as the resource demands increase in the legacy system, the corresponding resource demands in the target system increase proportionally. Thus, we have:

$$D'_B = D_B \times \frac{D'_A}{D_A}. \tag{1}$$

Table 2: TPC-W Mixtures

Web Interaction	Browsing Mix	Shopping Mix	Ordering Mix
Browse	95%	80%	50%
Order	5%	20%	50%

As can be seen from equation (1), the resource demand of the production workload on the target system is equal to the product of the resource demand of the production workload on the legacy system and the ratio of the resource demands of the non-production workload on the legacy and target systems.

While (1) may be reasonable over a wide range of applications (it holds to an acceptable level of accuracy in our experiments), it may not hold in general. In particular, if either system has been configured for a particular workload while the other has not, there is no reason why (1) should hold. It still may be possible for a system designer to determine a relationship similar to (1) - for example, with a proportionally constant on the right hand side. This would be on a system by system basis, but note that the only quantities that would need to be estimated are the resource demands.

3.1.2 Experimental Results

We employ a J2EE implementation of TPC-W from the University of Virginia [15] as the benchmark. The specification of TPC-W defines 14 web interactions and three different mixtures of web interactions: browsing, shopping and ordering. As can be seen from Table 2, those three mixtures are classified by proportions of browsing and ordering interactions. In TPC-W, the number of users or consumers is represented by Emulated Browsers (EB), and this number is constant throughout an experiment. TPC-W also allows the configuration of the experimental environment through several input parameters. For example, the customizable think time (Z) is used to refer to the interval between receiving a reply from the last request and sending the next request.

We built this environment on a Dell desktop computer equipped with an Intel i7-2600 quad-core processor (octo-core logically with hyper-threading technology), 8GB RAM, and a 1TB disk (7200 RPM). For the Operating System, we use Ubuntu 12.04 LTS. For the Web Server, we use Apache HTTP Server 2.2.24 [28]. For the Application Server and Database Server, we use JBoss 3.2.7 [17] and MySQL 5.1.69, respectively.

Table 3: Platforms

Legacy system	Target system
dual-core platform with 1.6 GHz	quad-core platform with 3.4 GHz

Table 4: Workloads

	Non-Production Workload	Production Workload
First Scenario	Browsing Mix	Shopping Mix
Second Scenario	Browsing Mix	Ordering Mix

In our experiments, we consider a scenario of system migration among different numbers of CPU cores: migration from a dual-core platform to a quad-core platform (logically, with hyper-threading enabled) processor. At the same time, we scale the CPU frequency from 1.6 GHz to 3.4 GHz (see Table 3). We present the test results in two scenarios. In the first scenario, we assume the browsing mix is the non-production workload and the shopping mix is the production workload. In the second scenario, we assume the browsing mix is the non-production workload and the ordering mix is the production workload (Table 4).

We configure the number of users (EBs) to 100, mean think time to 3.5 seconds, and generate the database to contain 28,800 customers and 1,000 items. In addition, we set a measurement-interval (MI) of 20 minutes after 5 minutes of ramp-up (RU). The resource demands are calculated by the Service Demand Law.

We apply PELE in the following manner:

1. Measure the non-production and production workloads on the dual-core platform to obtain the average response times, while also measuring CPU and Disk utilizations.

2. Use MVA to calculate the mean response times and overall throughputs, and compare them with the measured average response times and overall throughputs, respectively.

3. Measure the non-production workload on the quad-core platform to obtain the average response time and overall throughput while also measuring CPU and Disk utilizations. Parameterize MVA to calculate the average response time and overall throughput to validate our model.

Table 5: Results of the First Three Steps

	dual-core (browsing)	dual-core (shopping)	quad-core (browsing)
$U_{CPU}\%$	38.48%	32.51%	18.50%
$U_{Disk}\%$	6.78%	9.31%	7.80%
D_{CPU} (sec.)	0.013531	0.011422	0.006456
D_{Disk} (sec.)	0.002384	0.003271	0.002722
R (sec.)	0.026383	0.020154	0.010494
X (/sec.)	28.4375	28.4625	28.6542
MRT-MVA (sec.)	0.0243	0.0204	0.0108
X-MVA (/sec.)	28.3744	28.4059	28.4832

Table 6: Mapping Results

	quad-core (mapping)	quad-core (shopping)
D_{CPU} (sec.)	0.005450	0.005454
D_{Disk} (sec.)	0.003735	0.003632
R (sec.)	0.0106 (MRT)	0.009469
X (/sec.)	28.4851	28.7467

4. Do the mapping step to calculate the resource demands of the production workload on the quad-core platform, and parameterize MVA to calculate the average response time and overall throughput.

5. Measure the production workload on the quad-core platform to validate our mapping method.

6. Compare the performance of the dual-core platform with that of the quad-core platform.

First Scenario: The data for the first three steps is shown in Table 5. There are two observations:

1. The MVA-based MRT and the measured average response times are quite close, so that our model is validated on both the legacy and target systems. This observation also holds for the throughput (X).

2. TPC-W achieves better performance with the browsing mixture on the quad-core platform, in terms of the average response time.

Next we apply the mapping step. We have already measured all of the required resource demands for the right hand side of equation (1), so we estimate:

$$\text{CPU demand with shopping mix on quad-core} = 0.006456 \times \frac{0.011422}{0.013531} = 0.005450\,\text{sec.}$$

and

$$\text{Disk demand with shopping mix on quad-core} = 0.002722 \times \frac{0.003271}{0.002384} = 0.003735\,\text{sec.}$$

We use these two resource demands to parameterize the MVA algorithm, and then run the shopping mixture on the quad-core platform to validate our mapping method. The results are shown in Table 6. As can be seen from the table, the estimated and measured resource demands are quite close. In addition, the MRT and throughput generated by the MVA algorithm with the estimated resource demands and the measured values are very close. We can conclude that our mapping method is validated here.

Finally, we predicted the performance metrics with the production workload on the target system. Then we compare these performance metrics with those on the legacy system, and we find that TPC-W performs better on the quad-core platform with both the browsing and shopping mixtures.

Second Scenario: We repeat the first scenario, with the modification that we choose the ordering mixture as the production workload. The data for the first three steps is shown in Table 7. Since the non-production workload is the same as for the first scenario, we focus on the column which presents the results with the ordering mixture on the dual-core platform. As can be seen from the table, the MVA-based MRT and the measured average response time are quite close, so that our model is validated with the ordering mixture on the legacy system.

For the mapping step, we estimate:

$$\text{CPU demand with shopping mix on quad-core} = 0.006456 \times \frac{0.003448}{0.013531} = 0.001645\,\text{sec.}$$

and

$$\text{Disk demand with shopping mix on quad-core} = 0.002722 \times \frac{0.012363}{0.002384} = 0.014116\,\text{sec.}$$

We use these two resource demands to parameterize the MVA algorithm, and then run the ordering mixture on the quad-core platform to validate our

Table 7: Results of the First Three Steps

	dual-core (browsing)	dual-core (ordering)	quad-core (browsing)
U_{CPU}%	38.48%	9.82%	18.50%
U_{Disk}%	6.78%	35.21%	7.80%
D_{CPU} (sec.)	0.013531	0.003448	0.006456
D_{Disk} (sec.)	0.002384	0.012363	0.002722
R (sec.)	0.026383	0.023530	0.010494
X (/sec.)	28.4375	28.4800	28.6542
MRT-MVA (sec.)	0.0243	0.0227	0.0108
X-MVA (/sec.)	28.3744	28.3872	28.4832

Table 8: Mapping Results

	quad-core (mapping)	quad- core (ordering)
D_{CPU} (sec.)	0.001645	0.001882
D_{Disk} (sec.)	0.014116	0.012538
R (sec.)	0.025 (MRT)	0.019115
X (/sec.)	28.3687	28.5925

mapping method. The results are shown in Table 8. As can be seen from the table, the estimated and measured resource demands are quite close. Note that the predicted MRT with the estimated resource demands is slightly off from the measured one (with acceptable error) in this case. Comparing the average response times of the browsing and ordering mixtures on the legacy system (which are quite close), the MRT of the ordering mixture will be roughly two times larger than that of the browsing mixture on the target system. The reason behind this is that the ordering mixture is I/O intensive with 35% disk utilization. In other words, the disk is the bottleneck for the ordering mixture. As a consequence, increasing the processor number (more precisely, enabling HTT) or CPU frequencies may not affect the performance. In contrast, faster processors will send requests to the disk more frequently, which results in more requests waiting in the queue of the disk. Fast processors turn out to spend even a larger proportion of their busy time on I/O waiting (compared with slower processors), which increases the average response time. We will see a similar case, in which we predict the MRT more precisely, in Section 3.2.3.

We actually simulated another scenario in which the system is also migrated from a dual-core platform to a quad-core platform, but the CPU frequency decreases from 3.4 GHz to 1.6 GHz. We did this to demonstrate that our methodology does not rely on a faster target system. Since this is not a typical case of migration (migrating a fast platform to a slow one), and in the interest of space, we did not include the experimental results, but the mapping method is also applicable in that scenario.

3.2 SysBench

So far, we have discussed systems that can be modeled using product-form solutions. Those solutions are only applicable under some assumptions, *e.g.*, non-priority scheduling, infinite queue capacity, non-blocking factors, state-independent routing, *etc.* However, product-form assumptions can be violated. This could be due to simultaneous resource possession, blocking, high service time variability, *etc.* In this case study, we will deal with the transaction latency issue (Mi *et al.* [23, 22]) which is caused by locking operations in a database. A locking transaction can deny all requests that try to access the locked data. Such a locking condition on a database could cause simultaneous resource possession, and slow down service during a locking period. Blocked requests will remain idle until the resource is unlocked. Thus, service times could be much longer than usual due to these locking behaviours. When this transaction latency issue occurs, the burstiness of both service times and resource utilizations is observed. The burstiness in the service times violates the product-form condition, and the measured resource utilizations can be very misleading. Since the measured utilizations under such service time burstiness can be very inaccurate, we cannot simply use the Service Demand Law to calculate the service demands as we did before. The MVA algorithm is thus inaccurate in such a scenario.

To produce such a scenario, we adopt a different benchmark called SysBench [19], which generates on-line transaction processing workloads to evaluate the performance of the database server (we still use MySQL). In our experiments with SysBench, we observe that the service rate increases as the number of jobs in the system increases. Eventually, the service rate saturates and the system is under heavy load. Thus, we make one assumption here: we are only interested in the performance under heavy load for both systems. The reason for this assumption is that SysBench itself is designed for stress tests. The statistics generated by SysBench with a small number of jobs are quite unpredictable, *i.e.*, the overall throughput may vary significantly in consecutive tests.

Table 9: Results Using Multi-resource Queueing Network Model with Load Independent MVA

Benchmark	U_{CPU}	U_{Disk}	R(predicted)	R(measured)
SysBench	16.72%	55.54%	0.06946 sec.	0.132.87 sec.

For instance, when we use a two-resource (CPU and disk) queueing network to model the system, and the Service Demand Law to parameterize the single class MVA algorithm, we have the predicted MRT as shown in Table 9. As can be seen, the result has completely unacceptable error (around 200%). The reason is that the resources may be possessed by one request while others are blocked due to the frequent locking behaviours. As a result, the average resource utilizations we measured remain at a low level. Those average values are misleading. They present a situation in which the resources are idle most of the time as the average arrival rate is small. However, the real situation is that many requests were sent by users, but they are blocked in the queue. Consequently, the predicted MRT based on the Service Demand Law is much smaller than the measured one.

3.2.1 Choice of Performance Model

We choose an approximate solution for non-product-form queueing networks, called the Flow-Equivalent Server (FES) method. By this method, we replace a non-product-form system by a flow-equivalent server. The service rate of the flow-equivalent server is equal to the throughput of the system.

In this case study, the whole system is modeled by a queueing network which has M resources. Regardless of the value of M, all of the servers in the network are replaced by a single FES server. By measuring the overall throughput of the system, we have the service rate of the FES server, and then we can solve the corresponding queueing model using a single class MVA algorithm, which yields the mean response time and the mean queue length. By this method, we avoid measuring resource utilizations and computing the resource demands by the Service Demand Law. The details of the FES method and the MVA algorithm that we use can be found in Chapter 36 in Jain [16].

The think time is exponentially distributed with mean Z, which can be configured in SysBench. The number of users N can also be configured in the benchmark. The default FES method requires an iterated calculation of service rates from 1 to N. In our case, due to the heavy load assumption, we only calculate the average service rate μ of the flow-equivalent server under heavy load.

3.2.2 Mapping Method

We have already seen the mapping method in Section 3.1.1. The mapping step here is almost the same as that in the previous case study. However, there are two differences. Firstly, we are mapping service rates rather than resource demands, due to the difference in parameterization methods. So we change the parameters in equation (1) in Section 3.1.1 from resource demands to service rates under heavy load. Secondly, we assume that both the legacy and target systems are under heavy loads (because of the limitation of SysBench). Therefore, we only need to consider one service rate - the service rate for each system. Note that it may be different workloads that bring both systems to heavy load.

3.2.3 Experimental Results

We present a case study for a database server system migration. The legacy system has slow processors and an out-of-date Linux kernel, while the target system has faster processors and an up-to-date Linux kernel. We present the test results in two scenarios. The intensity parameters of the workload do not change from the legacy to the target system in the first scenario. However, the intensity parameters of the workload change significantly from the legacy to the target system in the second scenario.

We built this environment on the same Dell desktop computer as the one in the case study with TPC-W. For the Operating System, we use Ubuntu 10.04 LTS server edition. We use the on-line transaction processing (OLTP) test mode in SysBench, which by default generates parameterized transactions consisting of 14 read operations, 5 write operations and 3 other operations (*i.e.*, lock/unlock table statements). We employ the default workload as our non-production workload. The system migration scenario is given in Table 10, while the two scenarios are given in Table 11.

We configure that 20 customers run simultaneously in SysBench, so that the system is under heavy load. We also configure the number of transactions to 2,000,000, and generate a database whose table size is equal to 1,000,000. A single test runs for around 20 minutes.

Table 10: Platforms

Legacy system	Target system
dual-core platform with kernel 2.6.32-35	octo-core platform with kernel 3.0.0-23

Table 11: Workloads

	Non-Production Workload	Production Workload
First Scenario	14-read, 5-write, 3-other	23-read, 14-write, 3-other
Second Scenario	33-read, 5-write, 3-other	63-read, 5-write, 3-other

We apply PELE in this case study as follows:

1. Configure SysBench with non-production workload A.
2. Measure SysBench with zero think time on kernel 2.6.32 and the dual-core platform to obtain overall throughput, which is the service rate of the FES server.
3. Measure SysBench with positive think time on kernel 2.6.32 and the dual-core platform to obtain the average response time.
4. Use the service rate from Step 1 as input to the MVA algorithm to calculate the mean response time, and compare it with the measured average response time from Step 2.
5. Repeat Steps 1 to 4 on kernel 3.0.0 and the octo-core platform.
6. Repeat Steps 1 and 2 with production workload B on the legacy system. Then we will have the service rate with workload B on the legacy system.
7. Based on the ratio of the two service rates with workload A on the legacy and target systems, and the service rate with workload B on the legacy system, apply the mapping step to calculate the service rate with workload B on the target system.
8. Repeat Steps 3 and 4 with workload B on the target system to validate our mapping method.
9. Compare the performance metrics of the target and legacy systems.

Table 12: Results of the First Five Steps

	2.6.32 (2 cores)	3.0.0 (8 cores)
Service Rate	140.52/sec.	162.52/sec.
Measurement (Z=0.02 sec.)	R=0.12704 sec. X=136.02/sec.	R=0.10580 sec. X=158.96/sec.
Prediction (Z=0.02 sec.)	R=0.1236 sec. X=139.23/sec.	R=0.1049 sec. X=160.19/sec.

Table 13: Workload increases but minor change

	2.6.32 (2 cores)	3.0.0 (8 cores)
Service Rate (non-production workload)	140.52/sec.	162.52/sec.
Service Rate (production workload)	41.55/sec.	48.06/sec. (predicted) 48.41/sec. (measured)
MRT (Z=0.02 sec.)	null	R=0.39610 sec. (predicted) R=0.39435 sec. (measured)

First Scenario: In this scenario, the difference between the non-production workload and the production workload is minimal - the non-production workload has 14 read operations, 5 write operations, and 3 other operations per transaction, while the production workload has 23 read operations, 14 write operations and 3 other operations per transaction. In other words, the typical workload increases from the legacy to the target system, but the nature of the workloads is the same in the sense that both workloads are still I/O intensive.

The experimental results of the first five steps with the non-production workload are shown in Table 12. The first row shows the service rates under the non-production workload on both the legacy and target systems. These service rates will be used as input parameters to the MVA algorithm, with which we could calculate the performance metrics with different think times, *e.g.*, we use these values to compute the MRT with 0.02 second think time. The second row shows the measured metrics, while the third row shows the predicted numbers. We have two observations from the information in Table 12:

1. The MRT and throughput generated by our performance model are close to measurements, so that our performance model is validated on both the legacy and target systems.

2. The performance of the target system with the non-production workload is better than the legacy system. We expect a similar result with the production workload.

We then measure the production workload on the legacy system, and calculate the service rate. We already have the service rates for the non-production workload on both the legacy and target systems. Using the mapping method, we predict the service rate for the new workload on the target system as: $41.55 \times (162.52/140.52) = 48.06$ per second. Results can be seen in Table 13. The measured service rate is also presented in the second row for validation.

Using this predicted service rate, we could calculate the performance metrics with increased workloads and different think times on the target system using the MVA algorithm. In the third row of Table 13, the predicted response time of 20 threads on the target system is 0.39610 seconds while the measured value is 0.39435 seconds. We also have two observations from Table 13:

1. The predicted service rate of the production workload on the target system is very accurate. Our mapping method is validated.

2. The predicted MRT generated by our performance model together with the predicted service rate is very accurate. Our performance model is validated with the production workload.

Second Scenario: So far we have seen the test results in a scenario in which workloads increase from the legacy to the target system. In the tests, the non-production and production workloads are I/O intensive and the predicted results are quite accurate compared with the measured results. To extend the test cases, we design a new test where the workload changes significantly. For instance, we increase the number of read operations per transaction by 49, while keeping the number of other operations the same. In other words, the production workload on the target system will become CPU intensive (with a large proportion of read operations and heavy load). The results for our mapping method are given in Table 14.

As shown in Table 14, the predicted service rate is 72.48 per second, while the measured rate is 68.21 per second. The error is 6.26%. Compared with the error in Table 13 (0.72%), the error increases due to the significant change in workloads. Although this error could be considered reasonable, we

Table 14: Significant Workload Change

	2.6.32 (2 cores)	3.0.0 (8 cores)
Service Rate (14-5-3)	140.52/sec.	162.52/sec.
Service Rate (63-5-3)	62.67/sec.	72.48/sec. (predicted) 68.21/sec. (measured)

Table 15: Choosing a Better Base Transaction Mix

	2.6.32 (2 cores)	3.0.0 (8 cores)
Service Rate (33-5-3)	102.76/sec.	113.90/sec.
Service Rate (63-5-3)	62.67/sec.	69.46/sec. (predicted) 68.21/sec. (measured)

still would like to find a better solution for such a scenario. The basic idea is to design a base workload whose features are similar to the production workload. In addition, this base workload must be operational on the target system to replace the non-production workload. Suppose that we choose a CPU intensive workload (transactions) as our base test comprising 33 read operations, 5 write operations, and 3 other operations. The test results can be seen in Table 15. With the new measurements of the base workload, the error decreases from 6.26% to 1.83%.

Finally, as can be seen from Table 12, the target system has better performance than the legacy system. Consequently, we suggest that the system migration is reasonable.

3.3 Discussion

It is very important to adopt appropriate analytic queueing models for both the legacy and target systems. The decision is made based on empirical measurements of the legacy system (the target system may not be operational). Product-form solutions are only applicable under certain assumptions, *e.g.*, non-priority scheduling, infinite queue capacity, non-blocking factors, state-independent routing, *etc.* On the other hand, product-form solutions are no longer valid when we observe certain characteristics in a system, *e.g.*, simultaneous resource possession, blocking, high service time variability, *etc.* Consequently, non-product-form solutions should be employed. To solve performance models, suitable MVA algorithms are adopted based on different requirements for product-form networks. For example, a single class MVA algorithm is easy to parameterize because it only requires service demands for one class. However, it cannot offer class-level performance metrics. For more performance details about services, a multiple class MVA algorithm could be considered to offer class-level performance metrics (if meeting Service Level Agreement (SLA) requirements is an issue). Approximate solutions, such as FES and Diffusion Approximations [4], may be considered for non-product-form networks. We have considered closed models in the paper, as they are typically used for the client/sever style systems that we have considered. It would not be difficult to extend this technique to open network models.

4 Related Work

Instead of measuring the performance with real workloads on the target system, analytic models together with synthetic workloads can be used to predict performance for the target system. Avritzer and Weyuker [2] introduced a synthetic approach of performance testing for distributed systems. Instead of porting software from the legacy to the target system to evaluate performance, they considered designing a synthetic workload for the target system. The synthetic workload can be generated and adjusted by commercial benchmarks to approximate the real resource demands for the target system. This novel method does not depend on the development of the software on the target system, so that it could provide early suggestions related to performance concerns for system designers before running the target system.

Several methods for performance evaluation and prediction have been integrated with the software development process (see Balsamo *et al.* [3] and Woodside *et al.* [31] for surveys). The majority of these methods require creating a performance model early in the development cycle. The performance models are used to provide quantitative results in order to adjust the architecture and design with the purpose of meeting performance requirements. These methods aim to support performance engineering early in the development process. However, when applied to a legacy system, it is of interest to explore how these methods can be used to address the particularities of performance evaluation during system migration.

The MVA algorithm is often used to analyze performance models. Chen *et al.* [7, 8, 9] proposed performance modeling approaches with different MVA algorithms (*e.g.*, single class, multiple class, *etc.*) for diverse environments and purposes. For example, in [8], the authors present a performance model using MVA during the translation from Service Level Objectives (SLOs) into lower-level policies that can then be used for design and monitoring purposes. They use a modified multi-resource multi-station multi-class MVA algorithm for performance analysis, and combine the performance model with regression analysis in the decomposition process.

Performance evaluation for transaction-based workloads may suffer from transaction latency. Mi *et al.* [23, 22] develop performance models to address this issue. In [23], they develop an application signatures based approach for performance evaluation using various system metrics. The application signature is built based on transaction latency profiles. They claim that there could be additional bottlenecks in the system under high load that contribute to transaction latency. In addition, there could be some inefficiencies and lock contention within the transaction code itself. These application metrics can be further used for efficient capacity planning, anomaly detection, and provisioning of multi-resource applications.

Different methods are adopted to estimate service demands to parameterize the MVA algorithm. Zhang *et al.* [32] rely on system measurements to derive CPU utilizations, and then use a Least Squares Regression (LSQ) approach to estimate the mean service time of different transaction classes. However, LSQ suffers from the multicollinearity problem, so its effectiveness will be affected when the linearity assumption is violated. Kalbasi *et al.* [18] present another estimation technique called Demand Estimation with Confidence (DEC), and verify the effectiveness of the DEC approach compared with LSQ and Least Absolute Deviations regression (LAD) using the TPC-W benchmark.

5 Conclusion

The PELE methodology that we have presented is adaptable to different MVA-based performance models. We have previously validated it with product-form queueing networks and a single class MVA algorithm, and in this paper we propose an efficient mapping method to estimate the resource demands or service rates for the target system. In addition, we validate PELE involving non-product-form networks and an associated FES method. The latter offers an option to evaluate system performance when a product-form solution is not applicable. Our methodology offers a novel approach for those who would like to adopt their own MVA-based performance models to evaluate computer system performance for a migration project.

There are several potential directions for future work. The models studied in this paper provide a widely applicable methodology to evaluate performance. In the future, we would like to investigate using multi-class queueing network models in PELE, which could offer more detailed performance information, in particular for fulfillment of class-level SLAs. Secondly, to date we have only used benchmarks to generate artificial workloads for performance evaluation. Our next goal is to characterize the workload and analyze performance for real datacenter migrations. Thirdly, all of the performance models used to date have been solved by MVA-based techniques. We would like to investigate other potential techniques that can be incorporated into the PELE framework.

Acknowledgement

The work reported in this paper was supported by the Ontario Research Fund and the Natural Sciences and Engineering Research Council of Canada.

References

[1] I. Al-Azzoni, L. Zhang, and D. G. Down. Performance evaluation for software migration. In *Proceedings of the second joint WOSP/SIPEW International Conference on Performance Engineering*, pages 323–328, 2011.

[2] A. Avritzer and E. J. Weyuker. Deriving workloads for performance testing. *Software – Practice and Experience*, 26(6):613–633, 1996.

[3] S. Balsamo, A. D. 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.

[4] G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications.* John Wiley & Sons, second edition, 2006.

[5] S. Casey. How to determine the effectiveness of hyper-threading technology with an application, 2011.

[6] K. Chandy, U. Herzog, and L. Woo. Parametric analysis of queueing networks. *IBM J. Research and Development*, 19(1):36–42, January 1975.

[7] Y. Chen, S. Iyer, X. Liu, D. Milojicic, and A. Sahai. SLA decomposition: Translating service level objectives to system level thresholds. Technical report HPL-2007-17, HP, 2007.

[8] Y. Chen, S. Iyer, X. Liu, D. Milojicic, and A. Sahai. Translating service level objectives to lower level policies for multi-tier services. *Cluster Computing*, 11(3):299–311, 2008.

[9] Y. Chen, A. Sahai, S. Iyer, and D. Milojicic. Systematically translating service level objectives into design and operational policies for multi-tier applications. Technical report HPL-2008-16, HP, 2008.

[10] P. P. da Silva, A. H. F. Laender, and P. B. Golgher. A simulation model for the performance evaluation when migrating legacy systems. In *Proceedings of the Conference on Software Maintenance and Reengineering*, pages 210–215, 2001.

[11] P. P. da Silva, A. H. F. Laender, R. S. Resende, and P. B. Golgher. CAPPLES - A capacity planning and performance analysis method for the migration of legacy systems. In *Proceedings of the Workshops on Evolution and Change in Data Management, Reverse Engineering in Information Systems, and the World Wide Web and Conceptual Modeling*, pages 198–212, 1999.

[12] P. P. da Silva, A. H. F. Laender, R. S. Resende, and P. B. Golgher. Characterizing a synthetic workload for performance evaluation during the migration of a legacy system. In *Proceedings of the Conference on Software Maintenance and Reengineering*, pages 173–181, 2000.

[13] A. De Lucia, R. Francese, G. Scanniello, and G. Tortora. Developing legacy system migration methods and tools for technology transfer. *Software: Practice and Experience*, 38(13):1333–1364, 2008.

[14] P. Didelon and B. Morin. Simple query comparison between a relational database and an object-oriented database. In *Proceedings of the SPIE.*

Survey and Other Telescope Technologies and Discoveries, pages 377–386, 2002.

[15] T. Horvath. University of Virginia, 2008.

[16] R. K. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modeling.* John-Wiley, 1991.

[17] JBoss Community. JBoss application server website, 2004.

[18] A. Kalbasi, D. Krishnamurthy, J. Rolia, and S. Dawson. DEC: Service demand estimation with confidence. *IEEE Transactions on Software Enginieering*, 38(3):561–578, 2012.

[19] A. Kopytov. SysBench website, 2004.

[20] J. McKendrick. Study: US government spends $36 billion a year maintaining legacy systems., Jan. 2011.

[21] D. A. Menascé, V. A. F. Almeida, and L. W. Dowdy. *Performance by Design: Computer Capacity Planning By Example.* Prentice-Hall, Upper Saddle River, NJ, USA, 2004.

[22] N. Mi, G. Casale, L. Cherkasova, and E. Smirni. Burstiness in multi-tier applications: Symptoms, causes, and new models. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, pages 265–286, 2008.

[23] N. Mi, L. Cherkasova, K. Ozonat, J. Symons, and E. Smirni. Analysis of application performance and its change via representative application signatures. In *Proceedings of the Network Operations and Management Symposium (NOMS'08)*, 2008.

[24] MySQL AB. MySQL community website, 2012.

[25] M. Reiser and S. S. Lavenberg. Mean-value analysis of closed multichain queuing networks. *Journal of the ACM*, 27(2):313–322, 1980.

[26] H. Sahraoui. Coping with legacy system migration complexity. In *Proceedings of the International Conference on Engineering of Complex Computer Systems*, pages 600–609, 2005.

[27] W. Teppe. The ARNO project: Challenges and experiences in a large-scale industrial software migration project. In *Proceedings of the Conference on Software Maintenance and Reengineering*, pages 149–158, 2009.

[28] The Apache Software Foundation. Apache http server project website, 2013.

[29] TPC Corporation. TPC-W official website, 2012.

[30] E. J. Weyuker and F. I. Vokolos. Experience with performance testing of software systems: Issues, an approach, and case study. *IEEE Transactions on Software Engineering*, 26(12):1147–1156, 2000.

[31] M. Woodside, G. Franks, and D. C. Petriu. The future of software performance engineering. In *Proceedings of the Future of Software Engineering Conference*, pages 171–187, 2007.

[32] Q. Zhang, L. Cherkasova, G. Matthews, W. Greene, and E. Smirni. R-Capriccio: A capacity planning and anomaly detection tool for enterprise services with live workloads. In *Proceedings of Middleware, Newport Beach, CA*, pages 244–265, 2007.

[33] Y. Zou. Quality driven software migration of procedural code to object-oriented design. In *Proceedings of the International Conference on Software Maintenance*, pages 709–713, 2005.