Simulation Model Driven Performance Evaluation for Enterprise Applications

Ernest Sithole
School of Computing and Information Engineering
University of Ulster
Coleraine - BT52 1SA
Co. Londonderry, UK.
e.sithole@ulster.ac.uk

Sally McClean
School of Computing and Information Engineering
University of Ulster
Coleraine - BT52 1SA
Co. Londonderry, UK.
si.mcclean@ulster.ac.uk

Bryan Scotney
School of Computing and Information Engineering
University of Ulster
Coleraine - BT52 1SA
Co. Londonderry, UK.
bw.scotney@ulster.ac.uk

Gerard Parr
School of Computing and Information Engineering
University of Ulster
Coleraine - BT52 1SA
Co. Londonderry, UK.
gp.parr@ulster.ac.uk

Adrian Moore
School of Computing and Information Engineering
University of Ulster
Coleraine - BT52 1SA
Co. Londonderry, UK.
aa.moore@ulster.ac.uk

Stephen Dawson
SAP Research Belfast
Queens Island, Titanic Quarter
Belfast - BT3 9DT
stephen.dawson@sap.com

ABSTRACT

Performance evaluations for enterprise applications running over IT systems are difficult to carry out given the multiplicity and variability of the operational components that constitute the dispersed IT infrastructures. To overcome this challenge, most of the approaches for performance assessment employ benchmarking strategies. While benchmarking methods provide exact indications on the performance capability of the measured facility, the results so obtained mostly apply to specific physical implementations considered in benchmark runs. The information provided by benchmark data thus restricts the ability to carry out meaningful performance analysis unless wide varieties of physical scenarios are generated for comparative studies. Given the logistical drawbacks associated with benchmarking techniques, we therefore propose a flexible model-based approach to determine quantitative performance for applications in IT systems by producing a range of performance models through the use of generic components that are easily assembled in simulation environments. Our approach initially considers a Tier 2 model framework whose components are derived from the SAP Sell-from-Stock application routine running on a multi-core processor server. The modelled framework is extensible enough to provide the definitions of resource consumptions patterns of different applications as well as the variety of server hardware systems. The simulations of our initial models developed so far generate results that are comparable to measurements obtained for scenarios in the low and moderate loading levels.

SimulationWorks2010, March 15 - 19, Torremolinos, Malaga, Spain

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modelling Techniques; G.3 [**Probability and Statistics**]: Distribution Functions; I.6 [**Simulation and Modelling**]: Model Validation and Analysis

General Terms

Measurement, Performance

Keywords

Benchmarks, Enterprise Applications, Performance Evaluations

1. INTRODUCTION

As highlighted in [19], most computing environments are inherently too complex to carry out performance analyses for running applications. This challenging nature of performance evaluation emanates from the various aspects of the computing systems' makeup and operation such as (i) the multiple functional components such as hardware nodes and software routines making up the infrastructure, which are almost invariably very high in number, (ii) topology changes in the infrastructure due to reconfigurations and enforcement of dynamic resource allocation policies, (iii) the variety of internal behaviours on each of the functional elements that constitute the distributed system, and (iv) the dependencies that may exist between the functional characteristics and some of the relationships not readily discernible.

For other computing systems such as Grid-, Virtualisation- or Service Oriented Architecture (SOA) based systems, where functional capabilities are abstracted and presented as service packages, accurately determining performance patterns is even more challenging. The increased complexity arises

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8706
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8706

from additional characteristics of the middleware based service framework. Given the intractable nature of the tasks associated with identifying the key functional elements of IT systems and accurately describing their main behaviours, many approaches have adopted benchmarking procedures to carry out performance evaluation of applications. Benchmarks are made up of sets of metrics obtained from systematic measurements carried out in order to capture the exact performance capability of either (i) the entire system implementation (which constitutes hardware, OS platforms, application routines and support software as in the case of the SAP Sales and Distribution benchmark [15][16]) or (ii) a subset of the IT infrastructure's functional components e.g. CPU hardware [20] or backend database transactions [21]. The data provided through benchmarking techniques is precise, and depending on the specific parameter(s) that the benchmark is set up to measure, the performance capability which a particular IT server implementation is able to achieve can be obtained in terms of response times, throughput rates or scalability trends. However, benchmarking techniques suffer from a number of weaknesses, one of which is that the performance metrics obtained from benchmark runs are mostly applicable to a specific implementation arrangement and thus the full significance of measurements is not always clear. Thus, in order to derive more meaning from benchmark results, a variety of configurations for computing implementations have to be set up to enable comparisons. The comparisons can then reveal performance patterns from which the respective contributions of various system components to the benchmark results are established.

The logistical challenges involved in generating multiple scenarios for benchmarking purposes are (i) the time taken and technical skills required in setting up benchmarked infrastructures, (ii) the cost of hardware, software and other support facilities and (iii) possible disruptions to, or interference from, other operations that execute on the live IT infrastructure. In view of the drawbacks associated with benchmarking techniques, we propose a performance modelling approach that is based on the use of resource models, which can be readily developed and assembled in a simulation environment. Another important feature that is built into the models is the characterisation of user behaviours so that patterns of workload levels are defined. The response of the physical resource to user requests is defined in terms of generic runtime properties of applications and the definitions are described so that the resource consumption patterns can be extended to incorporate overheads associated with IT middleware support strategies like SOA, database and virtualisation systems. We derive the performance of distributed applications from combining two sets of definitions which are made up of user activities and resource attributes. From the performance delivery trends obtained in the initial baseline models, we intend to develop further scenarios, which will enable the carrying out of studies on such aspects as the scalability trends for distributed applications and resource utilisation levels whenever workloads and IT resource configurations change. Thus, while our study in this paper is limited to the characteristics a single application implementation, the modelling technique presented is a case study of a general strategy through which various IT system configurations can be defined. We consider our proposed technique to be a useful approach that has the potential to be a complementary and alternative strategy to existing performance evaluation schemes that rely upon direct physical measurements in order to establish the capabilities of IT implementations.

2. MODELLED FRAMEWORK FOR APPLICATION PERFORMANCE

In the simulation approach that we consider, use is made of OPNET-based components in order to determine application performance metrics of the modelled application running over an IT infrastructure. The extensive definitions in OPNET cover such aspects as user behaviour (in terms number of customers, intervals and communication overheads associated with customer requests); application resource needs at server runtime and, the functional and quantitative capabilities (associated with communication links, CPU and Storage server resources). It is important to highlight that the principal model definitions of application resource consumptions and server capabilities are OPNET-independent and so could therefore be deployed in alternative simulation environments. Reference can be made to Figure 1 for the main components that make up the model and physical implementation that were used in our simulation and measurement experiments.

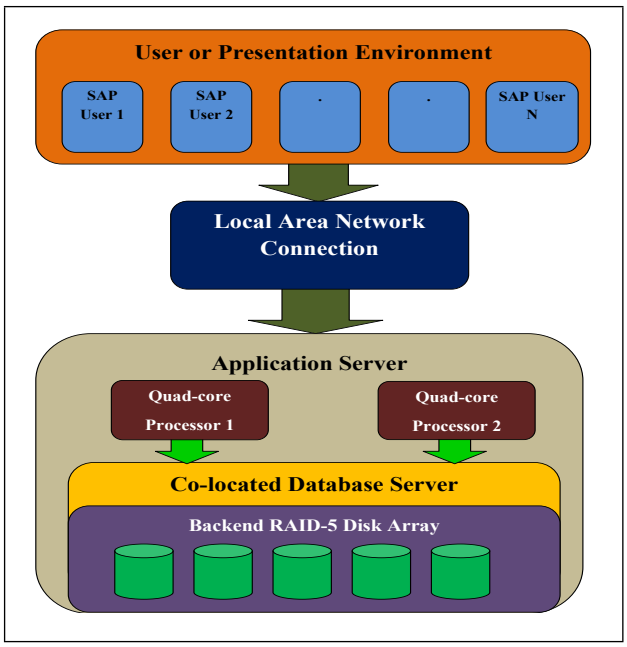

Figure 1: Tier 2 Server Configuration employed in the Measurements and Simulation Experiments.

We base the composition of the enterprise application routine in our experiments on a Sell-from-Stock scenario, which in terms of its component steps is structurally similar to the SAP Sales and Distribution (SD) Benchmark execution sequence [7][16]. The Sell-from-Stock is a component module that forms part of the application suite provided by the SAP Enterprise Resource Planning (ERP) Software package [17]. While the modelled application routine bears compositional similarity to the SD Benchmark, the actual resource consumption patterns that we consider in our models and physical measurements are only applicable to the experi-

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8706
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8706

ments featured in this paper. The various elements of our model and their respective functions are briefly described as follows:

- Choice of the Tier 2 configuration, which is made up of the two main functional stages of (i) User or Presentation Environment and (ii) the Server Entity.
- The User Environment from which requests are launched to the server through a Local Area Network (LAN-based) 100 Mbit/s communication link.
- The User Activities which generate application requests to the server. The user-initiated events make up the 15 operational steps described in [16]. For sequence and composition of the user steps of the entire routine of the enterprise application, reference can be made to information in Table 1.
- The Server Entity which handles application requests through CPU computational and database read/write operations.
- The Runtime Routine which describes the execution of the application. The application is treated as a composite business process [7]. We use the information provided in [16] for the definition of the Sell-from-Stock scenario in terms of fifteen steps, which we configured as execution stages of a user request cycle. The runtime properties of the received request correspond to resource consumption at CPU, memory and disk storage stages for each work unit associated with the executing process. The execution parameters can be adjusted to describe overheads related to the modelled IT implementation such SOA- or database-driven strategies.
- The Server Responses that communicate with the user following completion of server execution of received requests.

Figure 1 shows the Tier 2 server configuration that was used in the initial experiments on application performance carried out in simulation and on our physical testbed. The Tier 2 implementation involves two functional stages in the execution of distributed applications; the Presentation Environment and the composite Server Entity. According to this server setting, both the application and database service routines are provided on the same physical server entity. The main operational stages that are involved in the execution of the Sell-from-Stock are shown in Table 1. The model descriptions of the internal functionality of the Sell-from-Stock routine are achieved by extending the method presented in [7] i.e. our model definitions treat the execution of Sell-from-Stock routine as a business process that is accomplished through a sequence of 15 user-initiated steps.

The characterisations of server resources are based on the server hardware attributes from which the physical measurements were taken. The Sell-from-Stock application includes the creation of a customer order with five line items and the corresponding delivery with subsequent goods movement and invoicing. The execution of the benchmark routine consists of the following main transactions which are responded to by the server in a sequential manner:

- **VA01** - Create an order with five line items.

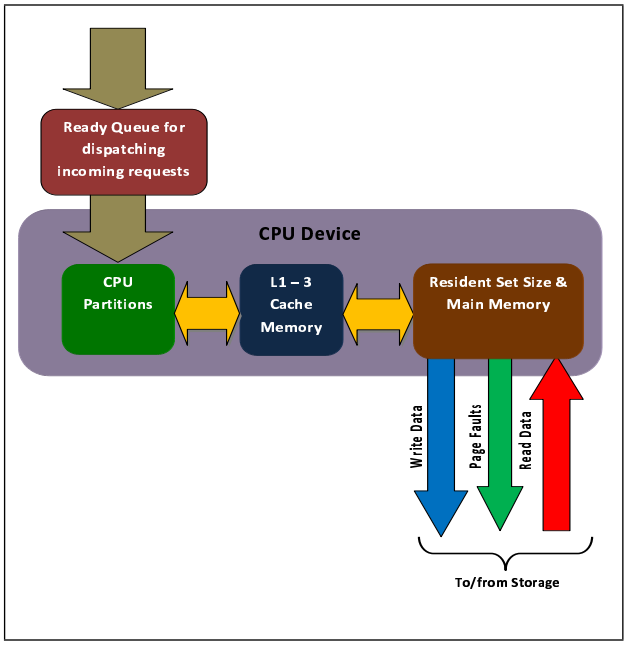

Figure 2: Modelled Components of the Server Processor Hardware.

- **VL01N** - Create a delivery for this order.
- **VA03** - Display the customer order.
- **VL02N** - Change the delivery and post goods issue.
- **VA05** - List of orders for one sold-to party.
- **VF01** - Create an invoice.

As shown in Table 1, there are 15 dialogue steps in this process (steps 2 to 16); these map into six transactions (VA01- VF01). These are repeated n times where each 15 step run takes at least 150 seconds due to a 10 second think time between steps. One run (steps 2 to 16) corresponds to the selling of 5 line items.

3. DEFINING USER REQUESTS AND APPLICATIONS CHARACTERISTICS

Given that the modelling approach we employ treats the Sell-from-Stock routine as a sequence of execution steps, we defined the individual job routines associated with server operation as a response to each of the 15 dialog steps contained in one complete user request-cycle. The discussion in [13][14] provides detailed explanation on the four levels of definition that are specified for the OPNET-based definitions of user activities, which are at Profile, Application, Task and Phase levels. The Profile definitions provide the description of the individual users who generate requests to the server. For each profiled user there is at least one Application routine that is launched for execution. The Application consists of Tasks, which in turn are broken down to Phases. It is at phase level that the actual event modelling (for both the user packet requests and server responses) is provided for our simulations. An important requirement in the modelling of the characteristics of the Sell-from-Stock execution

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8706
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8706

Table 1: Steps of the Sell-from-Stock Routine

Dialog Step	Operation	Associated Transaction
0	*Log On*	-
1	*Main Screen*	-
2	Create Customer Order	VA01
3	Enter Order Info	VA01
4	Enter Details for Items	VA01
5	Save	VA01
6	Create Delivery	VL01N
7	Enter Delivery Info	VL01N
8	Save	VL01N
9	Display Customer Order	VA03
10	Choose Customer Order	VA03
11	Change Delivery Info	VL02N
12	Post Goods (schedule delivery)	VL02N
13	List Orders	VA05
14	Choose Set of Orders	VA05
15	Create Invoice	VAF01
16	Save	VAF01
17	*End*	-
18	*Confirming Log Off*	-

is specifying the actual order in which the request events occur. Information from Table 1 was used to construct the exact sequence which describes the series of application requests that accomplish the entire Sell-from-Stock routine. The sequence determines the progression of the events for each application cycle of the 15 requests, which are executed as follows:

- User think time that precedes the invocation of service requests where the values for the user think time are assumed to be exponentially distributed with a mean of 10 second.
- A user request stage (made up of key strokes and/or click streams). The individual requests, which are separated by user think time, are made up of a series of data packets and in turn, packets are separated by inter-packet times (the inter-packet times are assumed to be exponentially distributed with a mean of 0.002 seconds and packet sizes vary following a uniform distribution ranging between 54 and 62 bytes for user requests.
- Server responses that are reflected through changes in user screen-graphics with an exponentially distributed pattern with a mean of 1,514 bytes characterizing the variability in the size of response packets Table 3 presents the processor runtime characterisations associated with each of the 15 dialog steps in the Sell-from-Stock request cycle. The descriptions of dialog step operations presented in Table 1 were used as a guide in defining the order of the CPU execution parameters.

An important set of model parameterisations involved the estimation of the length of time spent performing computational operations based on the data from our physical measurements. We postulated the phase-type Erlang and Gaussian distributions for characterising the variability of CPU Service Time. The use of the coefficient-of-variation analysis [4][8] indicated that the simple phase-type Erlang distributions with small orders are not appropriate for defining the length of time spent by work items executing at the server CPU stage. A sample size, n, of 25 measurements was provided for each of the steps run in the entire Sell-from-Stock execution cycle and the coefficient of variation, CV, was calculated from the expression,

$$CV = \frac{\sigma}{\mu}, \tag{1}$$

where σ and μ are the respective standard deviation and mean for each of the analysed datasets.

The Gaussian distribution pattern was considered via the sample skewness analysis [24] which presented encouraging results i.e. near zero skewness coefficients. The calculations of sample skewness coefficients of the data were carried out on datasets of the CPU Service Time measurements for Dialog and Update steps, which were aggregated into the six main transactions that make up the Sell-from-Stock Application execution cycle. According to this approach we considered in the analysis of input measurements data, the calculation of the skewness property, $g1$, was based on the standardised third moment and the standard deviation, so that the magnitude of the asymmetry in the respective samples of measurements data can be established through the formula,

$$g1 = \frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} (\frac{Xi - \mu}{\sigma})^3. \tag{2}$$

Verifications of the skewness analyses were conducted through the one-sample Kolmogorov-Smirnov test[4][5], which showed that five of the six transactions had asymptotic significance values that were above the minimum of 0.05 for normality as shown in Table 2. It can also be seen from Table 2 that both the asymmetry calculations and Kolmogorov-Smirnov test for the transaction VL01N do not produce skewness values that approximate well to Gaussian distributions.

Table 2: Results for Sample Normality Tests of CPU Service Times

Transaction Name	Sample Skewness Coefficient	KS Asymptotic Sig. (2-tailed)
VA01	0.95449	0.475
VL01N	3.20506	0.036
VA03	0.16459	0.621
VL02N	-0.16455	0.669
VA05	-0.01843	0.534
VF01	0.34983	0.136

From conducting further Kolmogorov-Smirnov tests to determine the degree of normality that is exhibited by the individual steps, which make up the VL01N transaction entity, it was established that one Dialog Step, D4001 and an Update Step, U3000, showed weak normality. However, given that the steps, D4001 and U3000, which did not fully satisfy the normality test make up only two of the total of 20 (Dialog and Update) runtime stages, which are invoked in the execution sequence of the entire Sell-from-Stock scenario, we assume that the overall runtime variability of CPU Service Time is Gaussian in nature. A further justification for assuming the normal distribution for the measurement

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8706
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8706

data is the fact that the values for the CPU consumption from the test requests that were generated can be considered as convolutions of independently and identically distributed random variables and hence, for sufficiently large samples, according to the Central Limit Theorem the probability density function of the samples approaches the Gaussian pattern[4].

Table 3: Input Parameters for CPU Service Times

Application Step	Mean CPU Service Time (s)	CPU Execution Time Variance
1	0.02172	0.00002
2	0.08469	0.00009
3	0.44393	0.00106
4	0.01293	0.00001
5	0.01828	0.00001
6	0.12276	0.00011
7	0.15038	0.00133
8	0.01799	0.00001
9	0.18059	0.00008
10	0.01914	0.00001
11	0.40924	0.00106
12	0.01017	0.00005
13	0.27586	0.00021
14	0.01762	0.00001
15	0.31614	0.00026

From the data that was captured for a single user, the patterns of processor execution times were therefore considered to follow a Gaussian distribution, the mean values and variances of which are shown in Table 3. Thus, the input parameter characteristics for the CPU service times as fed into our model correspond to the processor sharing implementation with the variability of the user requests and CPU service times following the exponential and Gaussian distributions, respectively.

Table 4: Input Parameters for Storage I/O Operations

Application Step	Input Read Requests	Output Write Requests
1	1	0
2	0	0
3	47	19
4	5	0
5	0	0
6	58	0
7	30	26
8	0	0
9	58	0
10	0	0
11	110	86
12	0	0
13	168	0
14	0	0
15	80	33

In addition to the processor-related tasks that execute in response to user requests, disk operations that are initiated by the CPU to read and write data in the external storage are also defined in our model. Both sets of runtime attributes (for CPU and I/O operations) shown in the two tables can be adjusted to accommodate the functional overheads that are related to SOA and database functionality. For the modelled database, the operations associated with the Create, Update and Delete functions are grouped under CPU-initiated write requests with the database read operations being attributed to CPU-read requests. Table 4 shows the list of the model parameters for the Sell-from-Stock routine and the number of CPU-generated read/write operations associated with each dialog step. The average block sizes for the input and output data were set to a fixed value of 1 Kbytes for every read or write operation. The input parameters for disk storage operations are based on the number of read and write requests associated with the invocation of each dialog step as estimated from the measurements data for a single user.

4. DEFINING SERVER HARDWARE CHARACTERISTICS

Based on the makeup of physical hardware of the server on which our measurements were conducted, we derive the CPU, disk and operating system definitions from a server machine which has a four quad-core AMD Opteron 8354 processing unit, 68 GB RAM and 1.2 TByte disk capacities. All the three above-mentioned hardware components are managed by the virtualisation operating system, VMware ESXi Hypervisor, through which the OS, CPU, memory and disk storage allocations for our Sell-from-Stock's physical experiments are managed. The specific definitions in the OPNET server modelling package of the three main components, which make up the derived virtual machine used in the Sell-from-Stock's experiments, are briefly described below.

4.1 Operating System and CPU Features

Received requests are put in the OS ready or dispatch queue, from where they are scheduled to run concurrently along with other pending routines executing on a common CPU partition. In order to model the execution of multiple jobs on a time-sharing basis, use was made of the OPNET predefined parameter for OS time slice duration. Based on the OS features of the virtualised Suse Linux Enterprise Server (SLES 10) operating system, the duration of 100 ms for the time slices is assigned to each of the multiple user requests at the CPU. The task sampling interval of 100 ms, which is associated with the Linux platform [13][14], governs the rate at which concurrent jobs are time-multiplexed for execution in our simulations. The option of pre-emptive scheduling is also chosen in our OS definitions so that our models can characterise the handling of received jobs according to their priority levels.

The processor parameters characterise the CPU capability through arrays of CPU partitions, with each partition providing the individual processor elements that perform the actual computational operations. Additional definitions for the processing functionality are the selection of operating system and page-fault based executions for virtual memory operations. From the hardware attributes of our virtualised server, we define the processing capability that corresponds to a two quad-core AMD Opteron 8354 CPU with 8 GB RAM running at 2.2 GHz shown in Figure 2. As the next

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8706
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8706

paragraph goes on to justify, we consider that even though a single hardware server entity is used in our experiments, the multiple processors provided internally by the CPU architecture have to be treated as individual server elements thereby making the associated CPU queue model an expanded server one in stead being an M/G/1 model, some versions of which are considered in [2][10][22][23].

It has already been stated in Section 3 that the inter-arrival times for user requests received at the server follow an exponential distribution pattern of mean 10 seconds, which is determined by the timer settings of the script that generates the Sell-from-Stock work items for execution in the server processor. With the processor service times having been characterised according to a normal distribution pattern, and with eight processor cores having been allocated on the physical test bed running on a SLES 10 operating system whose scheduling time slice is set to 100 ms, the modelled queue for the processor server is therefore defined as an M/G/8 - PS queue i.e. Poisson arrivals, a general service time distribution and 8 server elements with operating system-controlled service under a processor sharing discipline. Figure 2 shows the makeup of the processor functional components that were defined using the Server Modelling features.

4.2 Disk Drive Parameters

The model definitions for storage cover such aspects as disk spindle speeds, disk transfer rates, latency times, and the average and maximum seek times associated with the server's disks. The functional makeup of the storage server hardware that is modelled here is shown in Figure 3.

Additional definitions for Disk Interface parameters determine the bus transfer rates and activation or disabling SSA mechanisms on the storage interface device. From attributes of the virtualised storage on our server, we defined the storage interface parameters based on the Wide Ultra SCSI specification and the disk device on the IBM Deskstar 120 GXP (120 GB) performance characteristics. The associated data transfer speeds, latencies and drive seek times for the storage and interface devices are predefined in the OPNET server attribute object based on vendor specifications. Five disks are combined into a RAID 5 configuration that improves data transfer speeds by partitioning the input and output datasets into striped components. The striped datasets are then read from or written to the respective disks which they are assigned to. In modelling the RAID 5 setting, our definitions assume that one disk is set aside for error correction, thus effectively leaving out four disks for the transfer of useful data.

Based on our choices of the storage parameters, the following quantitative performance values are built into the Server Modelling features:

- Average seek time (ms) - 8.5
- Maximum seek time (ms) - 17
- Average latency (ms) - 4.17
- Spindle speed (RPM) - 10,000
- Disk transfer rate (MBytes/sec) - 55, 572.

Considering that the read-in and write-out operations initiated by the runtime executions of the step processes generate fixed numbers of data transfer requests as shown in Table 4 and also that the modelled operational response times of the storage interface and drive disk functions shown in Figure 3 are largely constant, the service time distribution for the storage back-end operations can therefore be treated as deterministic. Since the interactions with the back-end storage are initiated by the active processes being serviced at the CPU stage, the characterisation of the storage service requests can therefore assume that the generated requests for disk input and output operations follow the normal distribution pattern associated with processor service times. Based on the above mentioned considerations, the modelled disk queue for the server device is therefore defined as a G/D/1 - FCFS queue i.e. a general arrival pattern, deterministic service times and one server with First Come First Served queue discipline.

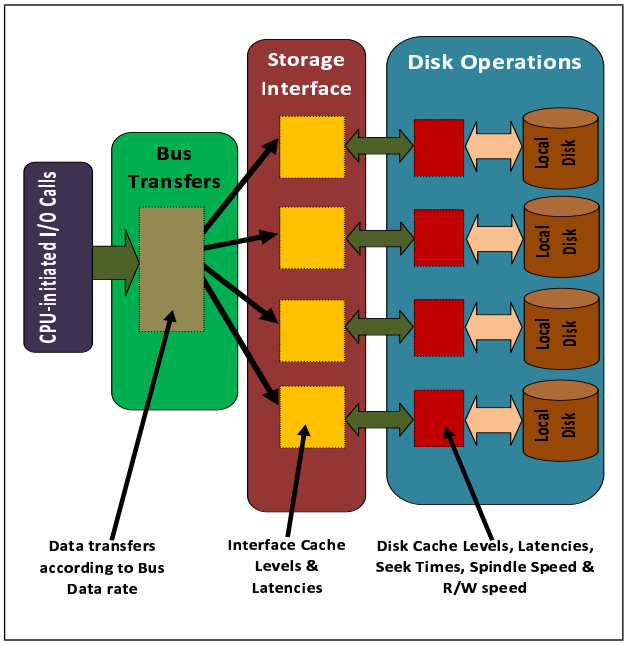

Figure 3: Modelled Components of the Server Storage Hardware.

5. EXPERIMENTAL SCENARIOS AND RESULTS

The experimental scenarios for simulations runs were defined as follows:

Our baseline scenario of the Sell-from-Stock model with users sending requests to the virtual machine is made up of two quad-core AMD Opteron 8354 CPUs with 8 GB RAM running at 2.2 GHz. To generate performance scenarios for the baseline graph, the application workload on the server was varied between 25 and 100 by changing the number of simultaneous user in uniform steps.

The duration of the simulated time was set to 40 minutes and the output results were computed based on the running averages obtained for the entire time simulation epoch, during which duration the volume of user requests can be considered to reach steady state.

In order to carry out comparative studies, the results from

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8706
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8706

the simulations were considered in conjunction with data captured from the server-based physical measurements of the Sell-from-Stock runs.

5.1 Response Times from Testbed and Simulations Runs

Our testbed experiments and simulations are based on scenarios of low and moderate user loads, with the following input loads being generated for comparison between the physical measurements and the simulations of models: 25, 50, 75 and 100 users.

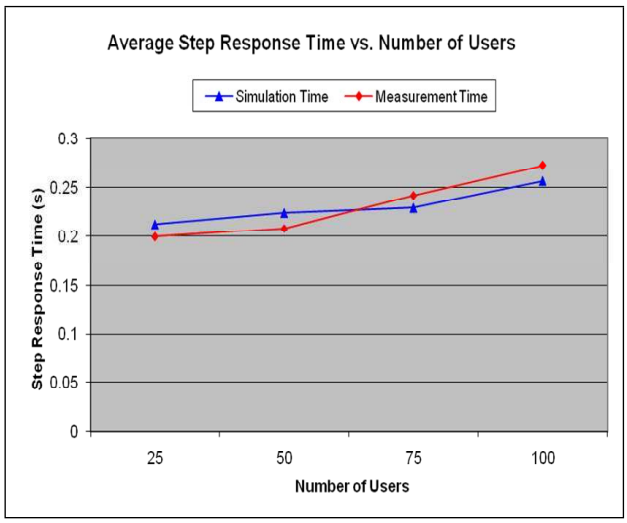

Figure 4: Response Times: Comparison of Simulation and Measurement Results.

Our simulations of storage hardware operation assume that the disks have internal Input/Output delay characteristics that correspond to both random and sequential disk access schemes. Figure 4 shows the variation of the average server response time for the Sell-from-Stock's 15 dialog steps as the load on the application server increases. There is a steady rise in server response times for both the simulations and measurements from an around 0.2 when 25 users are on the system up to approximately 0.25 seconds when the server load is at 100 users. The comparison of simulation and measurement results shows a variation which ranges between 5.4 and 7.1 % for the server response times in the four scenarios considered in our studies. These server response times can be broken down into CPU and storage times, which are in turn made up of service and wait times at the CPU and storage operational stages. Considering the fact that our featured experimental scenarios relate to the cases of low and moderate loads arriving at a server device with a handling capacity which is characterized by ready availability of memory and the CPU power of eight processing cores (that are carved out from the underlying server resources through the virtualisation technique), no significant increase in the response times occurs with the increased change in the number of user requests as both graphs indicate. Within the loading range of 25-100 users, the measurements data show a 38 % increase in response times, while the simulations results show an increase of 25 %. We attribute this deviation to the absence in our simulation experiments of operational overheads that are associated with the background processes involved in the execution of the Sell-from-Stock routine in practice.

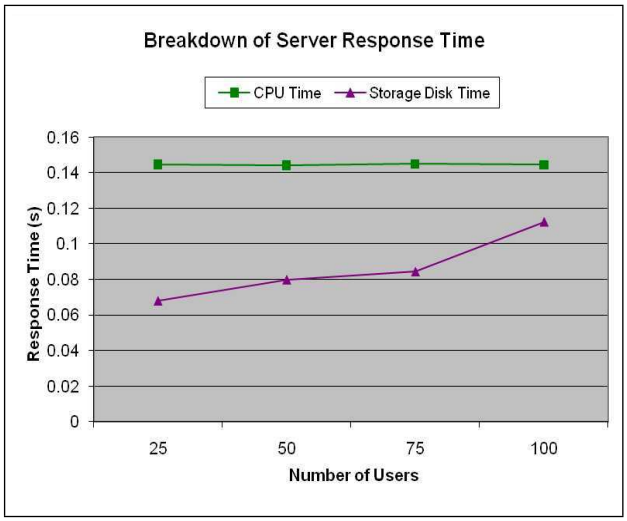

Figure 5: Server CPU and Disk Response Times.

For each of the modelled loading levels, the graphs in Figure 5 provide the breakdown of server response time into the constituent response times associated with the processing and disk storage stages. As can be seen from the trends of the individual durations, there is negligible change in the amount of time spent at processor. This pattern (of minor changes to the response times at the CPU with the rise in the number of users) further affirms the fact that the allocation of 8 processor cores for server response to the Sell-from-Stock routine is sufficient and it is only at much higher rates of requests that the load stress on the processing capability becomes noticeable.

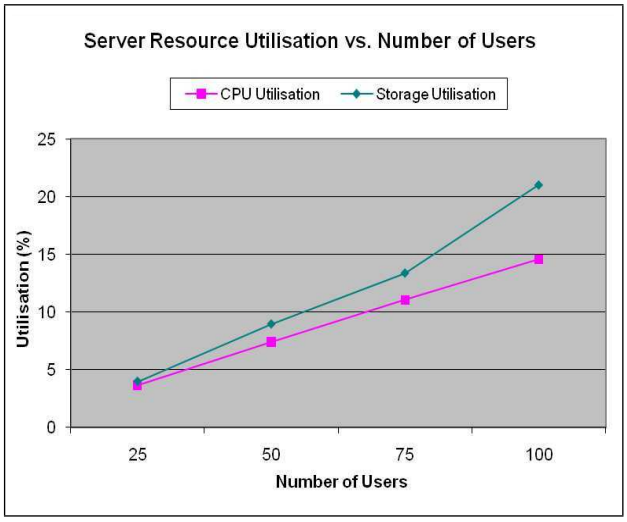

Figure 6: CPU and Disk Utilisation Rates.

The study of the storage response times however presents a different trend with an increase of 65 % over the featured

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8706
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8706

range of user loading. Since the actual operational times (associated with disk latencies, average and maximum seek times as well as disk transfer rates) are fixed for the modelled storage hardware, the average disk service times is therefore almost constant regardless of the number of read/write requests. Thus, the observed trend (of increase in disk response times with user load) can be attributed to the rise in waiting times due to contention at storage.

For more on the sensitivity of internal server hardware components to the number of users, reference can be made to Figure 6 to establish the variation in the levels of processor and disk utilisations with respect to changing load. The observations presented in Figures 5 and 6 suggest that for the type of disk hardware that is modelled here, significant rises in storage response times could result from either the introduction I/O bound requests [3] or from simply increasing the number of user requests with similar storage consumption patterns to the dialog steps associated with the Sell-from-Stock routine.

6. CONCLUSION AND FURTHER WORK

The modelled framework considered in this paper focused on the study of a single enterprise application, the Sell-from-Stock routine, which was run on a server with the operational capability of an 8-core CPU and RAID 5 disk architecture, the hardware resources of which are coordinated by the Suse Linux Enterprise Server (SLES) operating system. Our models and simulations were achieved in the OPNET environment using the Server models and OS characteristics [13][14], with the definitions of application resource consumptions being acquired from calibrating the work units according to the data obtained from physical test bed experiments. Although only a specific case of an enterprise application implementation was considered in this paper, the initial models and simulations that we developed can be broadened to describe other IT system configurations and the models can be also defined in other simulation environments. The generalisability and portability of our approach is based on providing two principal sets of definitions; the Application resource consumption patterns of the individual work items and the Server performance capabilities arising from operations at the CPU and Disk stages.

We then explored our proposed approach by modelling a Sell-from-Stock application routine running on a virtualised server platform. The simulation and measurement results based on the four loading scenarios that were considered in our studies showed comparable server response times for the enterprise application routine that we ran in the experiments. While these results are encouraging, we consider it important to have comparative studies for scenarios where greater loads are introduced on the server. Thus one natural progression of this work will involve running more experiments and making comparisons at higher loads where violations of QoS thresholds are approached.

Another dimension that we intend to explore in further research is the migration of our models to other infrastructure configurations such Tier-3 and parallel server settings. Given the ability of our current models to be extended to feature other scenarios based on varying user load, we consider our approach to be a potentially valuable aid in the analysis of other performance related parameters such as scalability trends, resource utilisation rates and levels of contention at server resources.

We also take cognisance of the fact that the sets of simulation experiments considered in this paper involved the use of server models that were based on the M/G/N - PS and G/D/1 - FCFS queues for the CPU and Disk components respectively. Thus, the Gaussian distribution used to model the CPU Service Time variability encapsulates the operational characteristics of the processor elements as well as data access times for the caching and main memory stages. We consider that further practical measurements may be necessary to determine the trends in CPU Service Time that may emerge at higher loads especially for memory-intensive applications, which generally result in higher cache misses ultimately leading to longer service times at the CPU. These additional measurements will be conducted with a view to exploring the use of phase-type distributions in M/PH/N-PS models [11][12], to quantify the impact of such data access strategies. Also important to consider in connection with phase type models will be the dependencies between number of users and the introduced memory consumption levels affecting cache performance rates for CPU architectures having multi-level and shared cache configurations [18]. Establishing these patterns in resource consumption would enable resource allocation and optimisation strategy [1][6][9] based performance trends obtained from various scenarios of application resource consumption and server configurations to be determined.

7. ACKNOWLEDGMENTS

We would like to express appreciation for contributions by researchers at SAP Belfast Laboratory with whom we had insightful discussions on the various topics presented in this paper. This research was jointly supported by INVEST Northern Ireland and SAP.

8. ADDITIONAL AUTHORS

Additional authors: Dave Bustard (University of Ulster - Coleraine, BT52 1SA. email: `dw.bustard@ulster.ac.uk`) and Giuliano Casale (SAP Research Belfast, BT3 9DT. email: `giuliano.casale@sap.com`).

9. REFERENCES

[1] P. Afeche. Delay performance in stochastic processing networks with priority service. *Operations. Research. Letters.*, 31(5):390–400, September 2003.

[2] N. Bansal. Analysis of the m/g/1 processor-sharing queue with bulk arrivals. *Computers. and Operations. Research.*, 31(5):401–405, September 2003.

[3] S. S. Conn. Oltp and olap data integration: A review of feasible implementation methods and architectures for real time data analysis. *IEEE SoutheastCon*, pages 515–520, April 2005.

[4] M. H. DeGroot and M. J. Schervish. *Probability and Statistics - International Edition.* Addison-Wesley. Publishing. Company., Reading, Massachusetts, 2001.

[5] Z. Drezner, O. Turel, and D. Zerom. *A Modified Kolmogorov-Smirnov Test for Normality.* University Library of Munich, Germany, http://econpapers.repec.org/RePEc:pra:mprapa:14385,, 2009.

[6] H. Feng, V. Misra, and D. Rubenstein. Optimal state-free, size-aware dispatching for heterogeneous

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8706
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8706

m/g/-type systems. *Performance. Evaluation.*, 62(1/4):475–492, October 2005.

[7] S. Finkelstein, R. Brendle, D. Jacobs, M. Hirsch, and U. Marquard. The sap transaction model: Know your applications. *ACM. SIGMOD. Conference.*, June 2008.

[8] L. Guan, M. E. Woodward, and I. Awana. Performance modelling and evaluation of computer systems. *Computer. and System. Sciences.*, 72(7):1119–1120, 2006.

[9] M. Harchol-Balter, K. Sigman, and A. Wierman. Asymptotic convergence of scheduling policies with respect to slowdown. *Performance. Evaluation.*, 49(1/4):241–256, September 2002.

[10] X. Jin and G. Min. Queue length distribution and loss probability of generalized processor sharing systems under multi-class self-similar traffic. *15th International. Symposium. on Modeling., Analysis., and Simulation. of Computer. and Telecommunication. Systems.*, pages 179–185, October 2007.

[11] J. Kim and B. Kim. The processor-sharing queue with bulk arrivals and phase-type services. *Performance. Evaluation.*, 64(4):277–297, May 2007.

[12] M. Nalecz. Graph-theoretic computation of characteristic function based on representation of phase-type distribution. *Performance. Evaluation.*, 64(6):591–611, July 2007.

[13] OPNET. Standard models user guide - applications model user guide, http://opnet.com. *OPNET. Technlogies. Inc.*, January 2004.

[14] OPNET. Standard models user guide - modelling concepts reference manual, http://opnet.com. *OPNET. Technlogies. Inc.*, January 2004.

[15] SAP. Published benchmark results. http://www.sap.com/solutions/benchmark/sd1results.htm.

[16] SAP. Published benchmark results. http://www.sap.com/solutions/benchmark/sd2tier.epx.

[17] T. Schneider. Sap performance optimisation guide. *SAP. Technical. Report. SAP. Press.*, 2003.

[18] R. Sharma and G. C. Sharma. A queueing model with the bulk arrival rate depending upon the nature of service. *Applied. Modelling.*, 20(6):470–475, June 1996.

[19] E. Sithole, S. I. McClean, B. W. Scotney, G. P. Parr, A. A. Moore, D. W. Bustard, and S. Dawson. A taxonomy-driven approach for performance measurement and modelling in service oriented architectures. *Non Functional Properties and Service Level Agreements in Service Oriented Computing (NFLSLASOC 08).*, November 2008.

[20] SPEC. Spec cpu 2006 results. http://www.spec.org/cpu2006/results/. *Standard. Performance Evaluation Corporation.*

[21] TPC. Tpc-c:online-transaction processing benchmark v5. online available, 2009. *Transaction. Processing. Performance. Council. http://www.tpc.org/tpcc/.*

[22] A. Wierman, N. Bansal, and M. Harchol-Balter. A note on comparing response times in the m/gi/1/fb and m/gi/1/ps queues. *Operations. and Research. Letters.*, 32(1):73–76, January 2004.

[23] W. Xiong, D. L. Jagerman, and T. Altiok. M/g/1 queue with deterministic reneging times. *Performance. Evaluation.*, 65(3-4):308–316, March 2008.

[24] L. Yua, S. Wanga, and K. K. Laic. Neural network-based mean variance skewness model for portfolio selection. *Computers. and Operations. Research.*, 35(1):34–46, January 2008.

Digital Object Identifier: 10.4108/ICST.SIMUTOOLS2010.8706
http://dx.doi.org/10.4108/ICST.SIMUTOOLS2010.8706