Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 490835, 10 pages
http://dx.doi.org/10.1155/2015/490835

Research Article

Response Time Analysis of Distributed Web Systems Using QPNs

Tomasz Rak

Faculty of Electrical and Computer Engineering, Rzeszow University of Technology, Aleja Powstancow Warszawy 12, 35-959 Rzeszow, Poland

Correspondence should be addressed to Tomasz Rak; trak@kia.prz.edu.pl

Received 5 March 2015; Revised 15 May 2015; Accepted 24 May 2015

Academic Editor: GuanJun Liu

A performance model is used for studying distributed Web systems. Performance evaluation is done by obtaining load test measurements. Queueing Petri Nets formalism supports modeling and performance analysis of distributed World Wide Web environments. The proposed distributed Web systems modeling and design methodology have been applied in the evaluation of several system architectures under different external loads. Furthermore, performance analysis is done to determine the system response time.

1. Introduction

Distributed Web Systems (DWS) development assumes that the systems consist of a set of distributed nodes. These systems provide up-to-date data within the set time frames. Groups of nodes (clusters) are organized in layers conducting predefined services. This approach makes it possible to check the response time and easily scale the system. An example of a DWS is a stock trading system [1] (as one class of Internet systems). In this system, it may be a requirement for certain positions to be bought or sold when market events occur. A certain amount of system latency may be acceptable, but the event must still be reacted to within a deterministic period of time. If a lot of system responses exceed the time limit, the system will not be often used by users. The response time specified as an average value is normally dictated by business not by the environment.

Modeling and design of DWS are developed in two ways (Figure 1). On one hand, formal models which can be used to analyze performance parameters are proposed [2–7]. To describe such systems, formal methods like Queuing Nets and Petri Nets are used. Sometimes elements of the control theory are used to manage the movement of packages in web servers [1]. Experiments connected with simulation models are the second way especially when there are many nodes [1, 8, 9]. Applying experiments and models greatly influences the validity of the systems being developed. The convergence of simulation results with the real systems results confirms the correctness of the modeling methods.

Our earlier works [1, 10] are based on Queuing Nets and Timed Coloured Petri Nets. A distributed Internet system model, initially described in compliance with Queuing Net rules, was mapped onto Timed Coloured Petri Net structure by means of queueing system templates. We have used two types of formal models that have been exploited in the industry. We created some separate system models using Queuing Nets and Petri Nets, which allow the performance analysis. The final Timed Coloured Petri Net based model can be executed and used for modeled system performance prediction.

In our solution we propose alternative Queueing Petri Net models [11]. The models have been used as a background for developing a programming tool which is able to map timed behavior of Queueing Nets by means of simulation. We developed [12, 13] our individual method of modeling and analysis of DWS. The well-known software toolkits such as Queueing Petri net Modeling Environment (QPME) [3] can be naturally used for our models simulation and performance analysis. We develop QPN models of DWS that allow the performance evaluation.

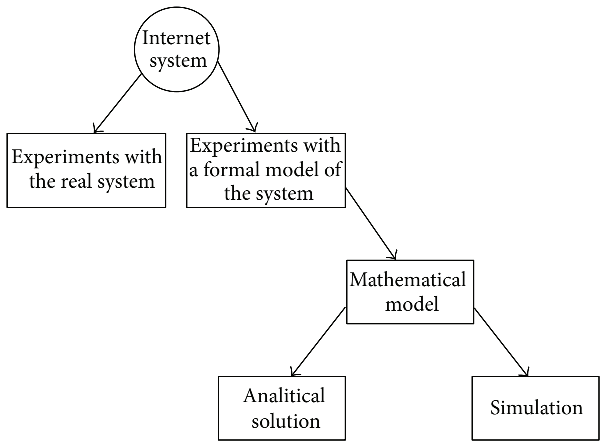

Internet system
Experiments with the real system
Experiments with a formal model of the system
Mathematical model
Analitical solution
Simulation

Figure 1: Performance modeling possibilities.

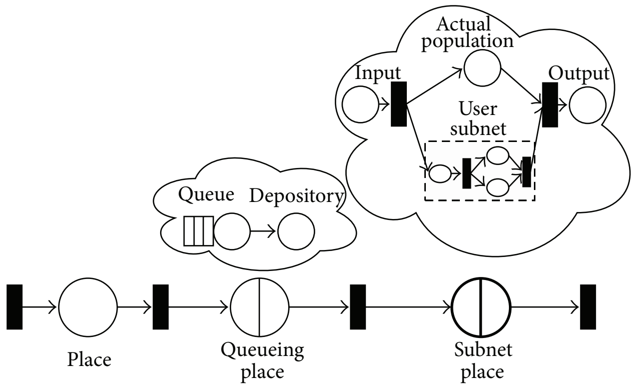


Figure 2: Elements of QPN.

The remaining work is organized as follows. Section 2 describes QPN. Section 3 presents DWS architecture and describes the modeling approach. Section 4 presents performance analysis results. The final section contains concluding remarks.

2. Queueing Petri Nets

In our solution, we propose a popular formal method—Queueing Petri Net [11]. This method is based on Queueing Nets and Petri Nets. Queuing Theory deals with modeling and optimizing different types of service units. Queueing Net usually consists of a set of connected queuing systems. The various queue systems represent computer components. Queueing Nets (QN) are very popular for the quantitative analysis [14]. QNs have a queue, scheduling discipline and are suitable for modeling competition of equipment. To analyze any queue system it is necessary to determine: arrival process, service distribution, service discipline, and waiting room (scheduling strategies). Petri Nets (PN) are used to specify and analyze the concurrence in systems. The system dynamics is described by the rules of tokens flow. The net scheme can be subjected to a formal analysis in order to carry out a qualitative analysis, based on determining its logical validity. PN have tokens representing the tasks and are suitable for modeling software. PN are referred to as the connection between the engineering description and the theoretical approach. Petri Nets are well-known models used to describe and analyze service units. PN cannot be used for a quantitative analysis due to lack of time aspects. The studies focus on incoming load measuring, for example, measure of the response time or presentation of an overall modeling plan. QN—quantitative analysis—has a queue and scheduling discipline and are suitable for modeling competition of equipment. PN—qualitative analysis—have tokens representing the tasks and are suitable for modeling software.

Queueing Petri Net (QPN) formalism is a very popular formal method of functional and performance modeling (performance analysis). These nets provide sufficient power to express modeling and analyzing of complex online systems. The choice of QPN was caused by a possibility of obtaining different character information. The main idea of QPN is to add queueing and timing aspects to the net places. QPN have the advantages of QN (e.g., evaluation of the system performance and the network efficiency) and PN (e.g., logical assessment of the system correctness). QN consists of a collection of service stations and clients. The service stations (queues) represent system resources while the clients represent users or transactions. A service station is composed of one or more servers and a waiting area (Figure 2). Tokens enter the queueing place through the firing of input transitions, as in other PN. When a request arrives at a service station, it is immediately serviced if a free server is available. Otherwise, the request has to wait in the waiting area. Different scheduling strategies can be used to serve the requests waiting in the waiting area. Places (QPN) are of two types: ordinary and queued. A queueing place (resource or state) is composed of a queue (service station) and a depository for tokens that completed their service at a queue (Figure 2). After being served by the service station, (coloured) tokens are placed onto a depository. Input transitions are fired and then tokens are inserted into a queueing place according to the queue's scheduling strategy. Queueing places can have variable scheduling strategies and service distributions (timed queueing places). Tokens in the queue are not available for output transitions while tokens in the depository are available to all output transitions of the queued place. Immediate queueing places impose a scheduling discipline on arriving tokens without a delay [11]. The last place on Figure 2 is called a subnet place. Hierarchical QPN (HQPN) has a dedicated input and output place, which are ordinary places of a PN. Tokens being inserted into a subnet place after a transition firing are added to the input place of the corresponding HQPN subnet. The semantics of the output place of a subnet place is similar to the semantics of the depository of a queueing place. Every subnet contains actual population place used to keep track of the total number of tokens fired into the subnet place.

QPN is a tuple (2), where CPN is Coloured Petri Net (1) [11, 15]:

$$\mathrm{CPN} = (P, T, C, I, M), \tag{1}$$

where

(i) $P = \{p_1, p_2, \ldots, p_i\}$ is a finite and nonempty set of places,

(ii) $T = \{t_1, t_2, \ldots, t_j\}$ is a finite and nonempty set of transitions,

(iii) $P \cap T = \varnothing$,

(iv) C is a colour function defined from $P \cup T$ into finite and nonempty sets (specify the types of tokens that can reside in the place and allow transitions to fire in different modes),

(v) $I(p,t)$ are the backward and forward incidence functions defined on $P \times T$, such that $I(p,t) \in [C(t) \rightarrow C(p)]$, $\forall (p,t) \in P \times T$ (specify the interconnections between places and transitions),

(vi) $M(p)$ is a initial marking defined on P such that $M(p) \in C(p)$, $\forall p \in P$ (specify how many tokens are contained in each place):

$$\text{QPN} = (\text{CPN}, Q, W), \tag{2}$$

where

(i) $Q = (Q_1, Q_2, (q_1, \ldots, q_{|P|}))$, where

  (a) $Q_1 \subseteq P$ is a set of timed queueing places,

  (b) $Q_2 \subseteq P$ is a set of immediate queueing places,

  (c) $Q_1 \cap Q_2 = \varnothing$,

  (d) $(q_1, \ldots, q_{|P|})$ is an array with description of places (if p_i is a queueing place, q_i denotes the description of a queue with all colors of $C(p_i)$ into consideration or if p_i is the ordinary place (p_i) equals *null*).

(ii) $W = (W_1, W_2, (w_1, \ldots, w_{|T|}))$, where

  (a) $W_1 \subseteq T$ is a set of timed transitions,

  (b) $W_2 \subseteq T$ is a set of immediate transitions,

  (c) $W_1 \cap W_2 = \varnothing$, $W_1 \cup W_2 = T$,

  (d) $(w_1, \ldots, w_{|T|})$ is an array (entry $w_j \in [C(t_j) \mapsto \mathbb{R}^+]$ such that $\forall c \in C(t_j) : w_j(c) \in \mathbb{R}^+$) of

    (1) rate of a negative exponential distribution specifying the firing delay due to colour, if $t_j \in W_1$,

    (2) firing weight specifying the relative firing frequency due to colour, if $t_j \in W_2$.

QPN have been recently applied in the performance evaluation of DWS, databases [16] and grid environments [17] because they are more expressive to represent simultaneous resource possession and blocking. Here, QPN models are used to predict DWS performance.

3. Distributed Web System

Among many Internet systems, we can indicate DWS.

3.1. Distributed Web System Architecture. Distributed Internet system architecture is made up of several layers:

(i) Layer 1 (Web servers—Tier 1) presents information (system offer) for clients in the web pages form and contains clusters.

(ii) Layer 2 (Application servers—Tier 2) manages transactions (clients requests) and provides the clustering functionality that allows load balancing.

(iii) Layer 3 (Database servers—Tier 3) controls the transactions, as a single element of this layer or multiple servers with database replication.

(iv) Layer 4 (Data nodes—Tier 4) is the data storage system.

In our approach the presented architecture has been simplified to two layers:

(i) Front-End (FE) layer is based on the presentation and processing mechanisms (Tiers 1 and 2). These two functions are realized by software.

(ii) Back-End (BE) layer contains one or more—in case of replication [1]—several databases. It consists of two presented Internet system layers (Tiers 3 and 4). This layer keeps the system data.

An architecture composed of these layers is used for e-busines systems. The presented double-layer system architecture realizes Internet system functions. These simplifications have no influence on the modeling process, which has been shown repeatedly for example, [3]. Proposed in the paper approach may be treated as an extension and continuation of solutions presented in [1]. Clustering mechanism was used in both layers. We used 1, 3, 6 and 9 nodes.

3.2. Client of Distributed Web System. An access to the system is realized through transactions. Activities related to the requests processing are as follows:

(i) Searching the Internet and sending a request by the Internet client.

(ii) Sending information to the Internet client or communicating with the application server by the presentation server.

(iii) Sending a request to the database by the application server.

(iv) Carrying out a transaction on the stored data, and returning results by the database server.

(v) Including results in appropriate places on a web page by the application server after receiving data.

(vi) Browsing web pages with results by the client using a web browser.

The characteristic feature of many DWS is a large number of clients using the Internet services (e.g., stock trading system) at the same time. DWS clients have different response time requirements. In case of the described systems class, clients

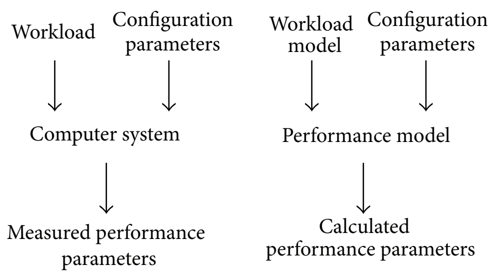


Figure 3: Difference between computer system and performance model.

are often focused on one event related to the same system offer. Based on these futures we used a stock trading system as a benchmark with two-layered architecture. In this paper, we consider one class of Internet systems and one class of Internet clients.

3.3. Modeling of Distributed Web System. We have many modeling methodologies. We can indicate some approaches to design like

(i) educated guess,

(ii) load testing,

(iii) performance modeling [3].

From many performance engineering models (workload, performance, availability, reliability, and cost) we have chosen and use the performance model (Figure 3). Performance models provide some recommendations to realize the required performance level.

At this stage of our research it has been decided that simulation will be the main mechanism used to do the analysis of the constructed models because our architecture is too large for analytical solutions (Figure 1). In our simulations we applied the performance analysis.

Typically, DWS are composed of layers where each layer consists of a set of servers—a server cluster. The layers are dedicated to adequate tasks and exchange requests between each other. To explain our approach to DWS modeling a typical structure will be modeled and simulated. The first layer (FE) is responsible for presentation and processing of client requests. The nodes of this layer are modeled by Processor Sharing (PS) (PS—requests are assumed to be served simultaneously with the server speed being equally divided among them (with infinitesimally small time slices)—it is used for modeling CPUs.) queues. The next layer (BE) implements system data handling. The nodes of this layer are modeled by using the First In First Out (FIFO) queue. Requests are sent to the system and then can be processed in both layers. The successfully processed requests are send back to the client. Client is modeled by Infinite Server (IS) queue.

Consequently, an executable (in a simulation sense) QPN model is obtained. Tokens generated by the arrival process are transferred in sequence by models of FE layer and by BE layer. QPN extend coloured Stochastic Petri Nets by incorporating queues and scheduling strategies into places forming queueing places. This very powerful modeling formalism has the synchronization capabilities of Petri Nets while also being capable of modeling queueing behaviors. The queue mean service time, the service time probability distribution function, and the number of servicing units defined for each queueing system in the model are the main parameters of the modeled system. In the demonstrated model it has been assumed that queues belonging to the same layer (FE and BE) have identical parameters. QPN consists of a set of connected queueing places. Each queueing place is described by arrival process, waiting room, service process, and additionally depository. We apply several queueing systems most frequently used to represent properties of system components.

In the Queueing Petri net Modeling Environment software tool, it is possible to construct QPN with queueing systems having PS and FIFO disciplines. As it was mentioned above, the main application of the software tool presented in the paper is modeling and evaluation of DWS.

4. Response Time Analysis

We have many Quality of Service parameters:

(i) performance (utilization, throughput, and response time),

(ii) availability,

(iii) reliability.

Monitoring of the above mentioned parameters helps to determine the system behavior. It allows collecting selected elements of the net state at the moment of an occurrence of certain events during the simulation. It has been mentioned above that in each of the model layers, response time will be monitored.

In these studies the performance is measured in terms of mean response time of business transactions. Parameters that determine the response time are

(i) workload intensity and hardware and software parameters,

(ii) residence time and service demand.

Response time (3) is equal a sum of residence times, where i is the number of places:

$$R = \sum_{i}^{k=1} R'_k. \tag{3}$$

Residence time (4) is equal to a sum of queueing time and service demand:

$$R'_k = Q_k + D_k, \tag{4}$$

where queuing time is $Q_k = \sum_i^{k=1} q_k$ and service demand is $D_k = \sum_i^{k=1} d_k$. Average service time in a particular resource does not contain the time of waiting for the resource. Service demand also does not depend on the load.

Table 1: Example of mean response time [ms] for 15 requests per second workload.

	Buy Quote	Sell Quote	Update Profile	Show Quotes	Get Home	Get Portfolio	Show Account
100 clients	158.329	168.806	62.192	62.595	43.929	95.894	38.257
200 clients	197.483	208.062	91.061	91.363	83.432	117.602	78.634
300 clients	197.142	209.295	91.639	91.682	83.165	114.57	78.446

4.1. Measured Parameters. Simulation parameters are based on a benchmark with realistic workload. We present the results of our earlier experimental analysis in [12]. The goal was to check—among others—the service demand parameter for FE and BE nodes. The application servers, considered as a FE layer, are responsible for the method execution. All sensitive data is stored in a database system (BE layer). When a client has to retrieve or update data, the application server makes the corresponding calls to the database system. DWS are usually built on middleware platforms such as J2EE. We used the DayTrader [18] performance benchmark which is available as an open source application. Overall, the DayTrader application is primarily used for performance research on a wide range of software components and platforms. DayTrader is a suite of workloads that allows performance analysis of J2EE application server. DayTrader is a benchmark application built around the paradigm of an online stock trading system. It drives a trade scenario that allows to monitor the stock portfolio, inquire about stock quotes, buy or sell stock. By client business transactions we mean the stock-broker operations: Buy Quote, Sell Quote, Update Profile, Show Quote, Get Home, Get Portfolio, Show Account, and Login/Logout. Each business transaction emulates a specific class of clients.

One of the most important requests [12], Buy Quote (Requests class (type), which has the bigger impact on the behavior of the system (Table 1)) is used in simulations. Experiments [12] have shown that mean number of requests per second for a FE layer is about 1400. Respectively, the mean measured number of requests per second for BE layer is about 7500 requests per second. We can also see that the delay in the requests processing is mainly caused by the waiting time for service in one BE node, but the main problem is the performance of the system response time. Our approach presented in [12] predicts response time for DWS and the relative error is lower than 15[%].

4.2. Queueing Petri Net Models. QPN models (Figure 4) are used to predict the system response time. We use the Queueing Petri net Modeling Environment (QPME) [15] tool. QPME is an open-source tool for stochastic modeling and analysis based on QPN modeling formalism used in many works [3, 5, 6, 19]. Scheduling strategies, service time distributions, and number of servers for queues are shown in Table 2. Queues are described by Kendall notation ($A/B/m/K/L/N$), where A denotes the probability distribution function specifying the interarrival time of tokens (− means different requests interarrival time), B is the probability distribution function of a service times (M means exponential (Markovian) distribution of requests service time), m is the number

Table 2: Queueing systems definitions.

Queue place	Queueing system	Description
Clients	$-/M/1/IS$[c]	Clients
FE_CPUm[a]	$-/M/1/PS$[d]	FE nodes
BE_I/On[b]	$-/M/1/FIFO$[e]	BE nodes

[a]m: number of FE nodes.
[b]n: number of BE nodes.
[c]IS service discipline.
[d]PS service discipline.
[e]FIFO service discipline.

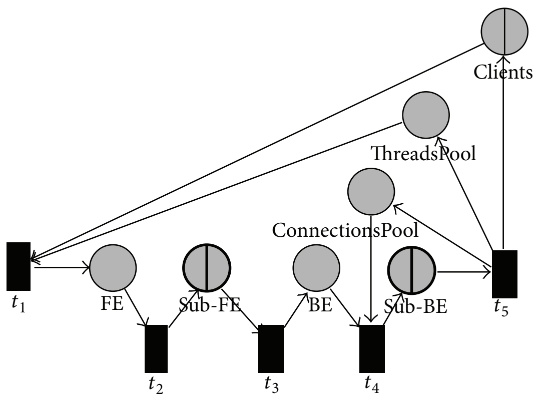


Figure 4: Main model of DWS.

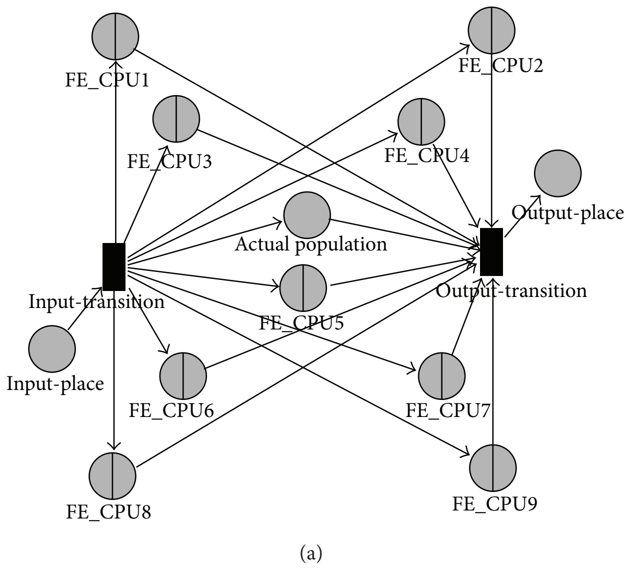
of servers (1 means one server), K limits the number of requests a queue can hold (if not specified, the default is ∞), L determines the maximum number of requests that can arrive in a queue, the size of calling source (if not set: $L = \infty$), and N is the scheduling strategy (if not specified, the default is FIFO).

Model elements are presented in Figure 4. Client think time is modeled by IS scheduling strategy (*Clients* place). Servers of FE layer are modeled using the PS queuing systems (*FE_CPU* places), in subnet place (*Sub-FE* in Figure 5(a)). Service in all queueing places is modeled by an exponential distribution. Service demands in layers are based on experimental results [12]:

(i) $d_{FE\_CPU} = 0.714$ (ms),

(ii) $d_{BE\_I/O} = 0.133$ (ms).

BE servers are modeled by FIFO queue (*BE_I/O* place), in subnet place (*Sub-BE* in Figure 5(b)). Places (*FE* and *BE*) are used to stop incoming requests when they await application server threads and database server connections, respectively. Application server threads and database server connections are modeled, respectively, by *ThreadsPool* and



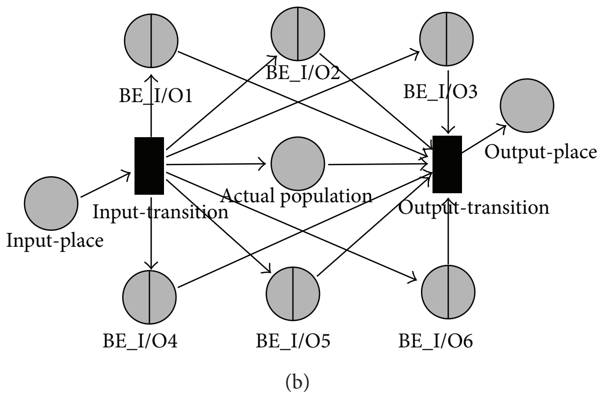


Figure 5: A subnet place example: (a) FE subnet with 9 nodes and (b) BE subnet with 6 nodes.

ConnectionsPool places. The process of requests arrival to the system is modeled by exponential distribution with the λ parameter (client think time) corresponding to the number of clients requests per second. The initial marking for places are

(i) number of clients (number of tokens in *Clients* place),

(ii) application server threads pool (number of tokens in *ThreadsPool* place),

(iii) database server connections pool (number of tokens in *ConnectionsPool* place).

In these models, we have three types (A colour specifies a type of tokens that can be resided in the place) of tokens:

(i) Requests.

(ii) Application server threads.

(iii) Connections to the database.

Based on definition (2), we define the following model (5) of DWS:

$$\mathrm{QPN} = (P, T, C, I, M, Q, W), \qquad (5)$$

where

(i) $P = \{FE, BE, ThreadsPool, ConnectionsPool\}$,

(ii) $T = \{t_1, t_2, t_3, t_4, t_5\}$,

(iii) C (Table 3),

(iv) $I(p, t)$,

(v) $M(p)$ (Table 4),

(vi) $Q = (Q_1, Q_2, (-/M/\infty/IS_{Clients}, null, -/M/1/PS_{Sub\text{-}FE}, null, -/M/1/FIFO_{Sub\text{-}BE}, null, null))$, where

(a) $Q_1 = \{Clients, FE\_CPU_m, BE\_I/O_n\}$,

(b) $Q_2 = \varnothing$,

Table 3: Type (color) to be attached to a token.

Place/transition	Color	Description
Clients, *FE_CPUm*, *BE_I/On*, *FE*, *BE*	*req*	Token represents a client's requests
ThreadsPool	*thr*	Token represents a thread
ConnectionsPool	*con*	Token represents a connection
t_j	*req*	Token represents a client's requests

Table 4: Initial marking.

Place	Value	Description
Clients	100, 200, 300, 400, 500	Number of clients
FE_CPUm	—	FE nodes
FE_I/On	—	BE nodes
ThreadsPool	30, 90, 180, 270	Number of threads
ConnectionsPool	40, 120, 240, 360	Number of connections
FE	—	Stop incoming requests
BE	—	Stop incoming requests

(vii) $W = (W_1, W_2)$, where

(a) $W_1 = \varnothing$,

(b) $W_2 = T$,

(c) $\forall c \in C(t_j) : w_j(c) := 1$ (all transition firings are equally likely).

4.3. Simulation Results. Many simulations were performed for various input parameters (Table 5). Total response time is a sum of all individual response times of queues and

Table 5: Parameters of simulations (one class of requests corresponds with Buy Quote requests).

	Element/parameter	Name/value
QPME	FE queues	*FE_CPUm*[a]
	BE queues	*BE_I/On*[b]
Software[c]	*ThreadsPool* place	30[d]
	ConnectionsPool place	40[e]
Client workload	λ	0.06[f]
	Clients place[g]	100, 200, 300, 400, 500
	Simulation time [s]	300

[a]*m*: number of FE nodes.
[b]*n*: number of BE nodes.
[c]Initial marking per node.
[d]30 threads for one FE node, 90 threads for three FE nodes, 180 threads for six FE nodes, and 270 threads for nine nodes.
[e]40 connections for one BE node, 120 connections for three BE nodes, 240 connections for six BE nodes, and 360 connections for nine nodes.
[f]Client think time equals 16,67 [ms].
[g]Client workload based on 15, 30, 45, 60 requests per second.

depositories in a simulation model without the client queue response time (client think time).

We investigate the bahaviour of the system during the increase in workload intensity. The number of clients was increased in accordance with values (*Clients* place) from 6000 to 30000 requests per second.

We used some scenarios in which we have a single requests class. The results involve the response time of the whole system. Multiple FE (1, 3, 6, and 9) and BE (1, 3, 6, and 9) nodes are the main configuration scenario. QPN model was used to predict the performance of the system for the scenarios (1FE1BE (1 node in FE layer and 1 node in BE layer), 1FE3BE, 1FE6BE, 1FE9BE, 3FE1BE, 3FE3BE, 3FE6BE, 3FE9BE, 6FE1BE, 6FE3BE, 6FE6BE, 6FE9BE, 9FE1BE, 9FE3BE, 9FE6BE, and 9FE9BE) and it was developed using QPME.

As a result, the response time of transactions is improved for cases with a higher number of FE and BE nodes. Increasing number of nodes resulted in simultaneous increase in the number of application server threads and connections to the database.

Figures 6 and 7 show the mean response time for all tests. Figure 6 shows the mean response time from the perspective of FE layer and Figure 7 from the perspective of BE layer. As we can see the overall response time decreased while the number of nodes was increasing.

The response time of one FE node architecture for all cases is the biggest. A difference in response time (Figure 6) between FE1 and FE3 is much bigger than between FE3 and FE9 (with different number of nodes in BE layer). We can also see the difference between system behavior for the increasing number of clients. For example, for 500 clients we can observe big discrepancy between FE1 and FE3 in all cases.

As we can see in Figure 7 an increasing number of nodes does not always reduce the response time. When more nodes are added, the analysis of their impact on other elements of the system should be preluded.

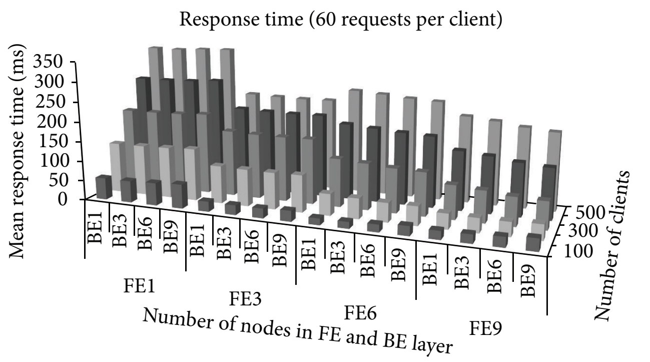


Figure 6: Mean response time simulation results for different number of nodes in FE (1, 3, 6, and 9) layer and in BE (1, 3, 6, and 9) and different numbers of clients.

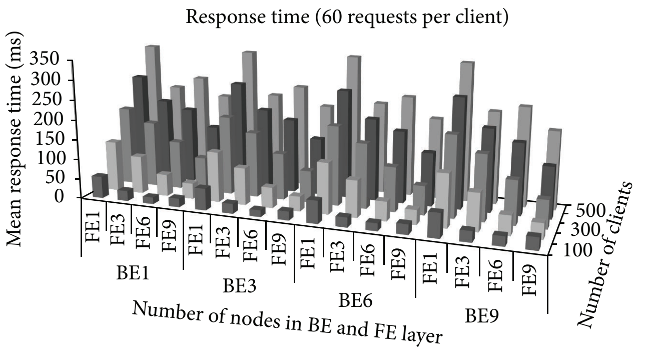
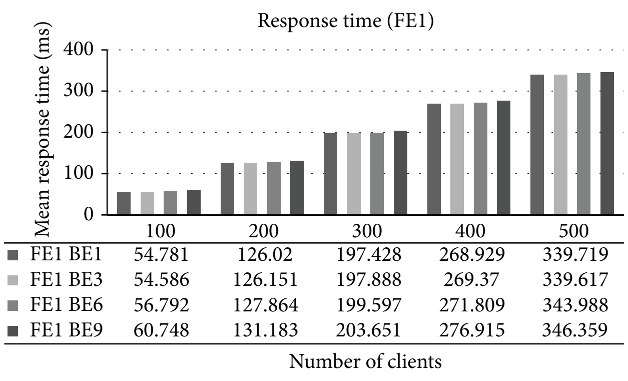


Figure 7: Mean response time simulation results for different number of nodes in BE (1, 3, 6, and 9) layer and in FE (1, 3, 6, and 9) and different numbers of clients.

In the two Figures 8 and 9, we have presented the same results for each part separately. We presented tables, below graphs, showing the results to clearer analysis. The overall system response time increases with the increasing workload. As we can see in Figure 8 mean response time decreases while the number of nodes in BE layer increases. The response time for the same number of nodes in FE layer is almost the same (Figures 8(a) and 8(b)). We can see that for 6 and 9 nodes in FE layer the mean response time decreases for different number of nodes in BE layer (Figures 8(c) and 8(d)).

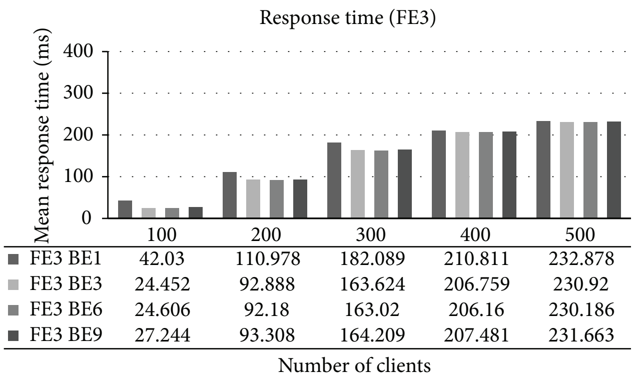
In the second scenario (Figure 9), the changes of the number of nodes in FE layer have a bigger impact on system response time. In all cases with the number of clients equal to 200, 300, and 400, we can observe linearly decreasing response time. The response time difference in cases of 100 clients may be due to a big number of nodes in BE layer compared to a small number of requests. Response time difference (shortest response time) in cases of 500 clients signalizes that 3 FE nodes (in this exampled load) is the best solution. More number of nodes in FE layer is also good but the benefit is not so obvious.

The basic conclusion is that it is difficult to determine what would be the behavior of the system after adding more nodes in layers; therefore, research and performance analysis is necessary.



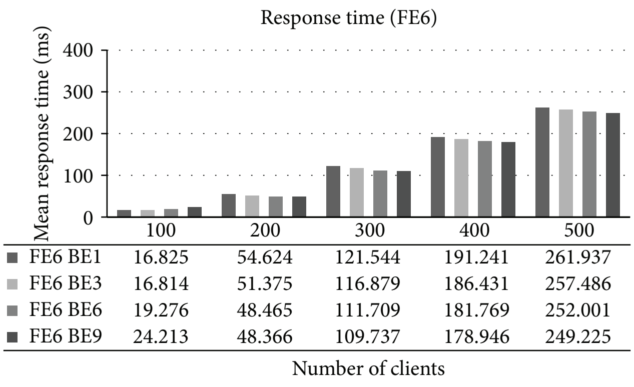
	100	200	300	400	500
■ FE1 BE1	54.781	126.02	197.428	268.929	339.719
■ FE1 BE3	54.586	126.151	197.888	269.37	339.617
■ FE1 BE6	56.792	127.864	199.597	271.809	343.988
■ FE1 BE9	60.748	131.183	203.651	276.915	346.359

(a)



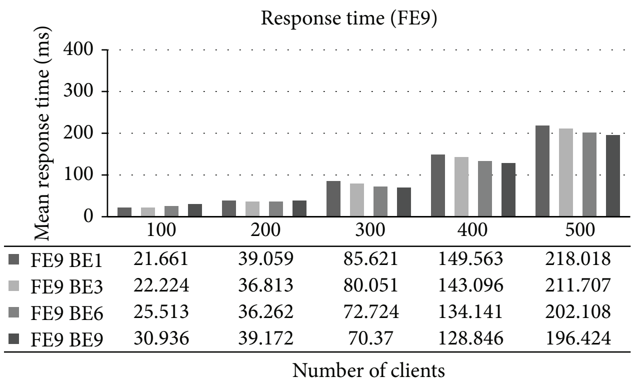
	100	200	300	400	500
■ FE3 BE1	42.03	110.978	182.089	210.811	232.878
■ FE3 BE3	24.452	92.888	163.624	206.759	230.92
■ FE3 BE6	24.606	92.18	163.02	206.16	230.186
■ FE3 BE9	27.244	93.308	164.209	207.481	231.663

(b)



	100	200	300	400	500
■ FE6 BE1	16.825	54.624	121.544	191.241	261.937
■ FE6 BE3	16.814	51.375	116.879	186.431	257.486
■ FE6 BE6	19.276	48.465	111.709	181.769	252.001
■ FE6 BE9	24.213	48.366	109.737	178.946	249.225

(c)



	100	200	300	400	500
■ FE9 BE1	21.661	39.059	85.621	149.563	218.018
■ FE9 BE3	22.224	36.813	80.051	143.096	211.707
■ FE9 BE6	25.513	36.262	72.724	134.141	202.108
■ FE9 BE9	30.936	39.172	70.37	128.846	196.424

(d)

Figure 8: Mean response time simulation results for different number of nodes in FE (1, 3, 6, and 9) layer and in BE (1, 3, 6, and 9) and different numbers of clients: (a) 1 node in FE layer and 1, 3, 6, and 9 in BE layer, (b) 3 nodes in FE layer and 1, 3, 6, and 9 in BE layer, (c) 6 nodes in FE layer and 1, 3, 6, and 9 in BE layer, and (d) 9 nodes in FE layer and 1, 3, 6, and 9 in BE layer.

5. Conclusions

We cannot always add new devices to improve performance, because the initial cost and maintenance will become too large. Because the overall system capacity is unknown, we propose a combination of benchmarking and modeling solution.

It is still an open issue how to obtain an appropriate DWS. Our earlier works propose Performance Engineering frameworks [10] to evaluate performance during the different phases of their life cycle. The demonstrated research results are an attempt to apply QPN formalism to the development of a software tool that can support DWS design. The result of the analysis is the discovery of the DWS structure useful in performance modeling. The idea of using QPN was proposed previously by other authors. In the presented approach, an alternative implementation of QPN has been proposed.

Earlier [12, 13] we set the parameters for the system experimentally. We verified [12] the influence of the individual layers on the system performance. The paper [13] focuses on the expansion of the model by increasing the number of elements in layers. The current model (5) has only a few parameters, but it is fully functional and can be scaled to larger systems. The modeling approach presented in this paper differs from previous works because of the following:

(i) Realistic workload was not used earlier.

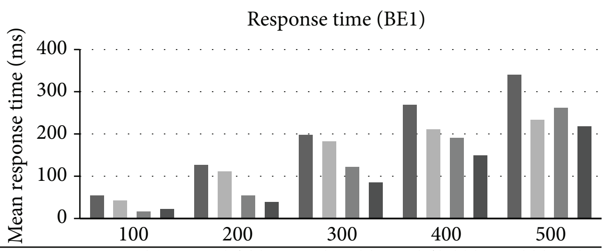
(ii) We showed the model of DWS with a greater number of nodes and different values on arcs.

(iii) We applied 50 number of runs for every simulation.

(iv) We took into consideration response times of all elements (queues and depositories of QPN), especially clients depository.

(v) Simplifying the model to a single element in a single layer.

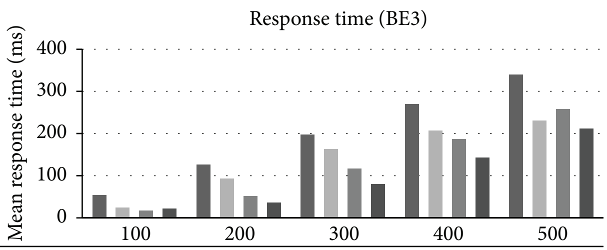
We develop a framework that helps to identify performance requirements (response time parameter). The study demonstrates the modeling power and shows how discussed models can be used to represent the system bahaviour, also in particular layers (Table 6). Next we shall consider analyzing the response time characteristics for more classes of clients (a separate token colour) to effectively model the Internet requests from the clients.



	100	200	300	400	500
■ BE1 FE1	54.781	126.02	197.428	268.929	339.719
■ BE1 FE3	42.03	110.978	182.089	210.811	232.878
■ BE1 FE6	16.825	54.624	121.544	191.241	261.937
■ BE1 FE9	21.661	39.059	85.621	149.563	218.018

Number of clients

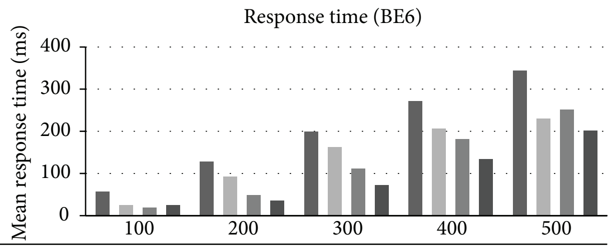
(a)



	100	200	300	400	500
■ BE3 FE1	54.586	126.151	197.888	269.37	339.617
■ BE3 FE3	24.452	92.888	163.624	206.759	230.92
■ BE3 FE6	16.814	51.375	116.879	186.431	257.486
■ BE3 FE9	22.224	36.813	80.051	143.096	211.707

Number of clients

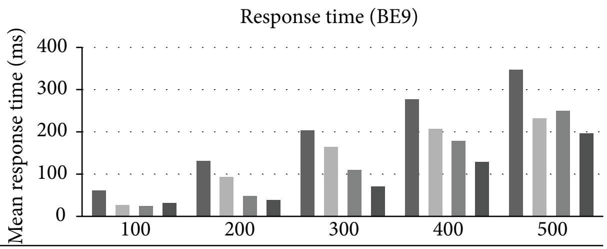
(b)



	100	200	300	400	500
■ BE6 FE1	56.792	127.864	199.597	271.809	343.988
■ BE6 FE3	24.606	92.18	163.02	206.16	230.186
■ BE6 FE6	19.276	48.465	111.709	181.769	252.001
■ BE6 FE9	25.513	36.262	72.724	134.141	202.108

Number of clients

(c)



	100	200	300	400	500
■ BE9 FE1	60.748	131.183	203.651	276.915	346.359
■ BE9 FE3	27.244	93.308	164.209	207.481	231.663
■ BE9 FE6	24.213	48.366	109.737	178.946	249.225
■ BE9 FE9	30.936	39.172	70.37	128.846	196.424

Number of clients

(d)

Figure 9: Mean response time simulation results for different number of nodes in BE (1, 3, 6, and 9) layer and in FE (1, 3, 6, and 9) and different numbers of clients: (a) 1 node in BE layer and 1, 3, 6, and 9 in FE layer, (b) 3 nodes in BE layer and 1, 3, 6, and 9 in FE layer, (c) 6 nodes in BE layer and 1, 3, 6, and 9 in FE layer, and (d) 9 nodes in BE layer and 1, 3, 6, and 9 in FE layer.

Table 6: Example of mean response time [ms] in system layers (FE6BE9).

	100	200	300	400	500
FE	13.178	37.654	99.082	168.294	238.58
FE + BE	23.534	48.02	109.423	178.639	248.921
System	24.213	48.366	109.737	178.946	249.225

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

References

[1] T. Rak and J. Werewka, "Performance analysis of interactive Internet systems for a class of systems with dynamically changing offers," in *Advances in Software Engineering Techniques*, vol. 7054 of *Lecture Notes in Computer Science*, pp. 109–123, Springer, Berlin, Germany, 2012.

[2] X. Chen, C. P. Ho, R. Osman, P. G. Harrison, and W. J. Knottenbelt, "Understanding, modelling, and improving the performance of web applications in multicore virtualised environments," in *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering (ICPE '14)*, pp. 197–207, March 2014.

[3] S. Kounev, C. Rathfelder, and B. Klatt, "Modeling of event-based communication in component-based architectures: state-of-the-art and future directions," in *Proceedings the 9th International Workshop on Formal Engineering Approaches to Software Components and Architectures (FESCA '13)*, vol. 295 of *Electronic Notes in Theoretical Computer Science*, pp. 3–9, Elsevier, May 2013.

[4] K. Jamróz, D. Pitulej, and J. Werewka, "Adapting enterprise architecture at a software development company and the resultant benefits," in *Software Architecture*, vol. 8627 of *Lecture Notes in Computer Science*, pp. 170–185, Springer, 2014.

[5] H. Koziolek, "Performance evaluation of component-based software systems: a survey," *Performance Evaluation*, vol. 67, no. 8, pp. 634–658, 2010.

[6] P. Meier, S. Kounev, and H. Koziolek, "Automated transformation of component-based software architecture models to queueing petri nets," in *Proceedings of the 19th Annual IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS' 11)*, pp. 339–348, Singapore, July 2011.

[7] C. Rathfelder, D. Evans, and S. Kounev, "Predictive modelling of peer-to-peer event-driven communication in component-based systems," in *Computer Performance Engineering*, vol. 6342 of *Lecture Notes in Computer Science*, pp. 219–235, Springer, Berlin, Germany, 2010.

[8] S. Samolej and T. Szmuc, "HTCPNs-based modelling and evaluation of dynamic computer cluster reconfiguration," in

Advances in Software Engineering Techniques, vol. 7054 of *Lecture Notes in Computer Science*, pp. 97–108, Springer, Berlin, Germany, 2012.

[9] K. Zatwarnicki, "Operation of cluster-based web system guaranteeing web page response time," in *Computational Collective Intelligence. Technologies and Applications*, vol. 8083 of *Lecture Notes in Computer Science*, pp. 477–486, Springer, Berlin, Germany, 2013.

[10] T. Rak and S. Samolej, "Distributed internet systems modeling using TCPNs," in *Proceedings of the International Multiconference on Computer Science and Information Technology (IMCSIT '08)*, pp. 559–566, October 2008.

[11] F. Bause, "Queueing petri vets—a formalism for the combined qualitative and quantitative analysis of systems," in *Proceedings of the 5th International Workshop on Petri Nets and Performance Models*, pp. 14–23, IEEE, Toulouse, France, October 1993.

[12] T. Rak, "Performance analysis of distributed Internet system models using QPN simulation," in *Proceedings of the Federated Conference on Computer Science and Information Systems*, pp. 769–774, September 2014.

[13] T. Rak, "Performance analysis of cluster-based web system using the QPN models," in *Information Sciences and Systems 2014*, pp. 239–247, Springer, Cham, Switzerland, 2014.

[14] S. Kounev, *Performance Engineering of Distributed Component-Base Systems, Benchmarking, Modeling and Performance Prediction*, Shaker, 2006.

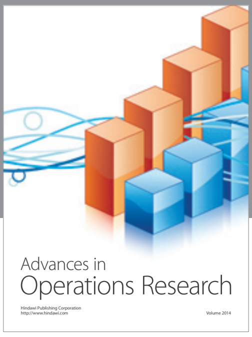
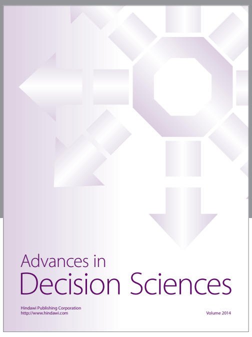
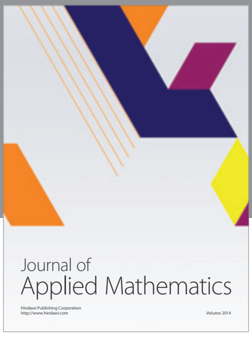
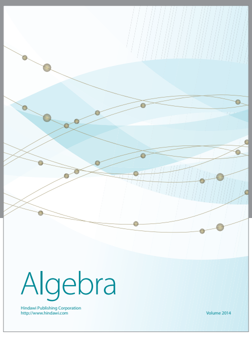
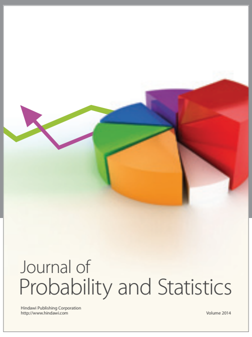
[15] S. Kounev, S. Spinner, and P. Meier, "QPME 2.0—a tool for stochastic modeling and analysis using Queueing Petri nets," in *From Active Data Management to Event-Based Systems and More*, vol. 6462 of *Lecture Notes in Computer Science*, pp. 293–311, Springer, Berlin, Germany, 2010.

[16] R. Osman, D. Coulden, and W. J. Knottenbelt, "Performance modelling of concurrency control schemes for relational databases," in *Analytical and Stochastic Modeling Techniques and Applications*, vol. 7984 of *Lecture Notes in Computer Science*, pp. 337–351, Springer, Berlin, Germany, 2013.

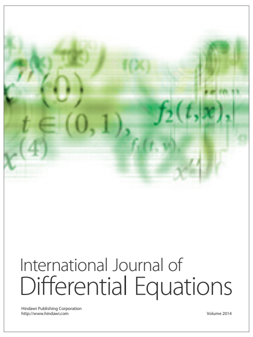
[17] R. Nou, S. Kounev, F. Julià, and J. Torres, "Autonomic QoS control in enterprise Grid environments using online simulation," *Journal of Systems and Software*, vol. 82, no. 3, pp. 486–502, 2009.

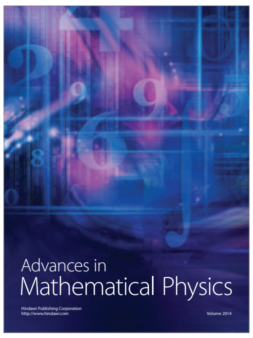
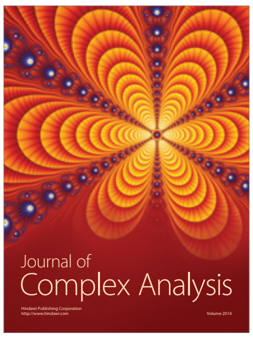
[18] https://geronimo.apache.org/GMOxDOC22/daytrader-a-more-complex-application.html.

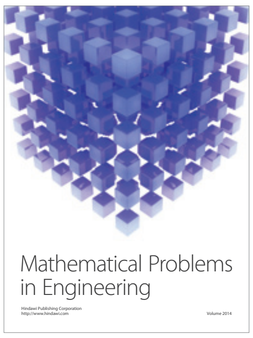
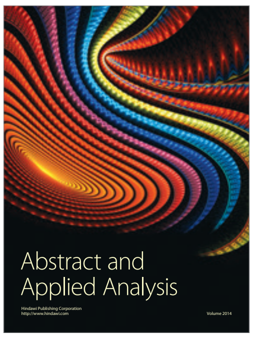
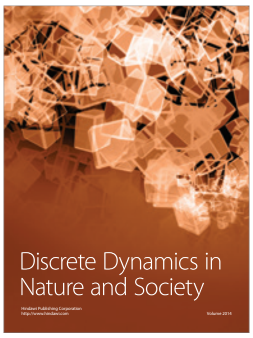
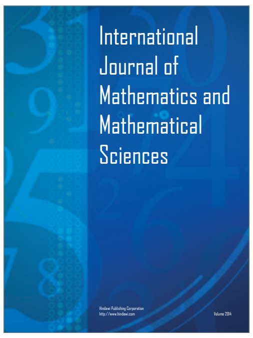
[19] D. Coulden, R. Osman, and W. J. Knottenbelt, "Performance modelling of database contention using queueing Petri nets," in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering (ICPE '13)*, pp. 331–334, ACM, April 2013.


Advances in
Operations Research

Advances in
Decision Sciences

Journal of
Applied Mathematics

Algebra

Journal of
Probability and Statistics

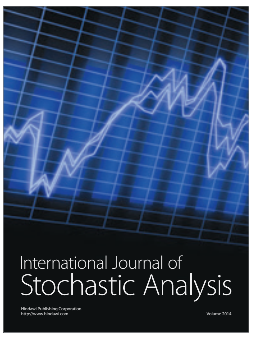
The Scientific
World Journal


International Journal of
Differential Equations

International Journal of
Combinatorics

Advances in
Mathematical Physics

Journal of
Complex Analysis

Journal of
Mathematics

Mathematical Problems
in Engineering

Abstract and
Applied Analysis

Discrete Dynamics in
Nature and Society

International
Journal of
Mathematics and
Mathematical
Sciences

Journal of
Discrete Mathematics

Journal of
Function Spaces

International Journal of
Stochastic Analysis

Journal of
Optimization