International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:8, 2015

Coloured Petri Nets Model for Web Architectures of Web and Database Servers

Nidhi Gaur, Padmaja Joshi, Vijay Jain, Rajeev Srivastava

Abstract**—**Web application architecture is important to achieve the desired performance for the application. Performance analysis studies are conducted to evaluate existing or planned systems. Web applications are used by hundreds of thousands of users simultaneously, which sometimes increases the risk of server failure in real time operations. We use Coloured Petri Net (CPN), a very powerful tool for modelling dynamic behaviour of a web application system. CPNs extend the vocabulary of ordinary Petri nets and add features that make them suitable for modelling large systems. The major focus of this work is on server side of web applications. The presented work focuses on modelling restructuring aspects, with major focus on concurrency and architecture, using CPN. It also focuses on bringing out the appropriate architecture for web and database servers given the number of concurrent users.

Keywords**—**Coloured petri nets, concurrent users, performance modelling, web application architecture.

I. Introduction

WEB application's performance plays a vital role in its usability. However, predicting performance before application's deployment poses a big challenge. A modelling approach is being considered to predict the performance impact on an application before its actual deployment in the production environment. Performance modelling provides future insight for application architecture requirements to achieve desired performance. It helps to infer scalability issues beforehand that may occur post deployment. Performance modelling works as a tool to guide application design, system design and thereby helps in understanding the overall application performance. Performance for client-server architecture based web application depends mainly upon the client and server design, its architecture, and network bandwidth. The focus of this paper is on modelling infrastructure architecture at server end to achieve the desired performance for client-server based web applications.

Web applications are event based and Petri Nets are one of the mathematical tools that are used to model the event based system. Hence, Petri Nets are chosen to model the web application performance in this work. The focus of this work is to model the database and web/application server for three tier client server architecture.

Web server architecture is a function of number of concurrent users accessing the application, total load on application and the peak load. Use of load balancer, total number of required servers and their architecture is dependent upon all these parameters. The model brings out this aspect and suggests architecture to be used with given number of concurrent users. In the case of database, the architecture varies depending upon the choice of database, type of SQL queries and number of concurrent users with similar type of SQL queries.

The focus of our work is to analyze the server deployment strategy with increasing number of concurrent users. The model will also help in predicting the load a server can handle as well as the number of servers or VMs required with increasing number of concurrent accesses. We have considered VM's but not the cloud environment.

The aim of the work is to develop a model which will represent performance aspects of a web application more precisely effect of concurrent users, and will also provide an insight for required restructuring of architecture to achieve the desired performance. The presented work focuses on modelling restructuring aspects using CPN, with major focus on concurrency and architecture. It also brings out the appropriate deployment architecture for web and database server with given number of concurrent users. The presented model should be easy to use even for those having no background of Petri-Nets. Using the suggested model, developers will be able to predict the application performance without even deploying the complete application on production environment.

The paper is organized as follows. Section II covers the work done by researchers in the area of web architecture modelling. It also covers the background of web architecture and petri-nets, and the reasoning of selecting CPNs for modelling. Section III describes the presented Petri net model for web and database server architecture. It also provides the simulations of these models. The presented models are analysed using different Petri net analysis techniques. The Section IV covers the analysis of the performance model. The paper is concluded with Section V on conclusion and future work.

II. Literature Review

A. Web Application Overview

Web application performance is discussed extensively in the literature and different approaches followed to model it. In recent years, researchers have done a lot of work on modelling web application for performance related issues. Web server and database server are crucial components in the performance of overall web application architecture. The performance and

Nidhi Gaur is with the Sardar Patel Institute of Technology, Andheri, Mumbai, India (e-mail: nidhigaur@sfitengg.org)

Padmaja Joshi, Vijay Jain, and Rajeev Srivastava are with the Centre of Development and Computing, Gulmohar Cross Road No. 9, Juhu, Mumbai (e-mail: padmaja@ cdac.in, vijayj@cdac.in, rajeevs@cdac.in).

scalability issues related to these components are also discussed in the literature using different approaches. Some of these are summarized here.

Beltran and Ayguade [1] discussed about improving web server performance using main memory compression. They implemented on the Linux OS a main memory compression system that takes advantage of the full power of current multiprocessors architectures, evaluated its performance with a highly-threaded web server and obtained positive results such as a 30% throughput improvement and a 70% reduction in the disk bandwidth usage. Mahnaz Shams and Far [2] used the model based approach for testing the performance of web application. The fine-grained control of workload characteristics are supported by this approach. The approach is beneficial to create controlled workloads and to study impact of varying workload characterstics on system performance. The methodology relies on Extended Finite State Machines (EFSMs). EFSMs can model applications with higher order request dependencies without encountering the state explosion problem. Marwah and Fetzer [3] proposed a web server architecture based on enhanced TCP splicing. The enhancements allow a TCP connection to be spliced at multiple proxies, providing both fault-tolerance and higher scalability. Zhong Xu and Bhuyan [4] described the DNS-based distributed system is a promising solution in terms of performance, scalability, and availability. DNS (Domain Name Server) name caching has significant effects on the load balance of distributed web server systems. Casalicchio and Tucci [5] discussed static and dynamic scheduling algorithms for web server scalability.

B. Performance Modelling

The Researchers used various methods for modelling the performance of web applications. Spitznagel and Garlan [6] show how queueing network modelling can be adapted to support performance analysis of software architectures. The authors also describe a tool for transforming a software architecture in a particular style into a queueing network and analyzing its performance. However, they also state that the performance accuracy depends upon the estimates supplied by the users and may not be reliable. Zhaoyang, Wei, and Zhiqian [7] proposed method of web server performance analysis, which models the web server service threads and the queue using Markov chain and queuing network, calculates coefficient of performance level, integrates different performance metrics, gives a direct quantifiable result, and matches optimal server parameters. Calculation of performance metrics like average delay time and throughput of the composite web service is provided by Zhang, Chang, Kim, and Chung [8].

Many of the researchers used Petri nets for modelling web application performance. Kounex and Buchhman [9] discussed how Queuing Petri-Net (QPN) models can be exploited for performance analysis of distributed e-business systems. Hai-Yan and Yan [10] have computed total average executive time of a work-flow represented using Petri nets using Stochastic Petri Nets and probability theory. Wang [11] used Petri nets for modelling of dynamic event driven systems. Samolej and Rak [12] proposed alternative Queueing Systems models expressed into Timed Coloured Petri Nets (TCPNs). The models have been used as a background for developing a programming tool which is able to map timed behaviour of queueing nets by means of simulation. Wells, Christensen and Mortensen [13] modelled distributed computing environments for performance analysis by means of Timed Hierarchical Coloured Petri Nets.

Unlike the approaches discussed, Coloured Petri Nets (CPNs) are used to model server sizing for concurrent users. Sizing of the database server and web server is to be done depending on the number of read and write requests. The states involved in this activity are load balancer, web server and database servers. These are defined by places. Since the impact of concurrent users on servers is to be brought out, one of the parameters that should be present in the model is concurrent users. At the server side type of database requests, number of web servers and database servers should be part of the proposed model. These are represented with different coloured tokens.

C. Petri Nets: Theory, Types of Petri-Nets along with Their Properties

Petri nets, as graphical and mathematical tool, provide a uniform environment for modelling, formal analysis, and design of discrete event systems. It can also be used to model asyn- chronous events, concurrent operations, process synchronization and conflicts or re- source sharing.

Historically Petri nets find its origin in 1962 in Petri's dissertation [14] Since then, Petri nets are explored and used in variety of industrial applications. Muratha [15] gives brief review of history of Petri nets, its behavioral and structural properties, various analysis methods and various applications. Properties of Petri nets are discussed by Zurawaski and Zhou [16] in the context of industrial applications. An example of a simple robotic assembly system is used for performance analysis, using Petri nets. The techniques are explained by examples of simple production systems. The paper introduces various types of Petri nets like high-level Petri nets, Fuzzy Petri nets, and Temporal Petri nets. Various methods and algorithms for analysis of Petri nets is discussed by Heiner and Donaldson [17].

D. Why Coloured Petri Nets?

As mentioned earlier, in order to model the web and database sizing architecture, one needs to consider parameters such as number of concurrent users, load balancer configuration, database request type-read or write, status of the active server etc. Normal Petri-Net model will become very complicated with these details as it allows only a single type of token. In Generic Petri-Nets all tokens are identical and there is no way to differentiate between the tokens.

However, in CPNs each of these parameters can be represented by a separate coloured token. CPNs allow to attach data value to a token in terms of token colour. The data value can be a simple number, a string, a structure consisting

of some fields etc. Associating additional data is also possible. This is mainly achieved by transition from identical tokens to typed tokens that can hold any kind of data. Hence, CPNs allow to construct a more compact model. Use of coloured tokens and simulation rules makes CPN a very powerful technique for modelling dynamic behaviour of a system as per Jensen [18]. We have used *PIPE-2* [19] tutorial to do the modelling using CPNs. PIPE-2 is chosen due to simplicity of the user interface and ease of work.

Though there is a lot of work done in the area of performance modelling, the focus of this paper is on web application architecture on server side. The model does not focus on network but rather on various possibilities of architectures of web and database servers that can help achieve the desired performance in terms of concurrent users. The model is based on experimentation and on experience of various web based applications. A qualitative analysis of the model is done using Petri-Net's analysis methods which help verify the correctness of the proposed model. The model identifies how the increase in number of concurrent users and type of query affects the sizing and architecture of the web and database servers. It also focuses on bringing out the appropriate architecture for web and database servers given the number of concurrent users and type of queries.

III. Model for Web Server Architecture

In a client-server architecture, web and database server capabilities determine the maximum number of concurrent users accessing the application. In this section, a model is proposed for web and database server sizing, and different architectures for them. The model also depicts the change required in the architecture with varying number of concurrent users. It focuses on the need to restructure the architecture on server side to improve the performance.

In the first phase, a Web Server Model (WSM) explaining the impact of concurrent users is developed. In the second phase, Database Server Sizing Model (DSSM) considering types of SQL queries is developed, and in the last phase a CPN model for Web Application Architecture is developed by integrating WS model and DSS model. The model is described in the same order ahead. Models are designed using CPN and a tool PIPE-2 of Charalambous [20] and Bonet [21] respectively. The selection of CPN as a modelling tool and the details of all these models are discussed in this section.

A. Web Server Model (WSM)

A Web Server Model design should capture architecture for 24X7 availability, scalability, and varying server capabilities. To achieve 24 x 7 availability, a fail-over server in active-passive mode is considered, whereas to achieve the scalability for supporting large number of concurrent users active-active clustered mode configuration gets the priority. Active-active configuration is used when concurrent access load need to be distributed among multiple clustered servers.

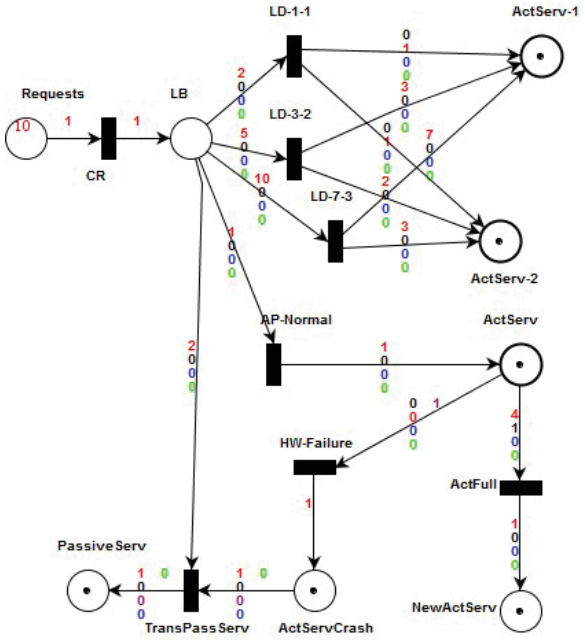

Fig. 1 Web Server Model (WSM) using CPN

1. Places and Transitions

Places or states which are used in WSM are (1) active and passive servers (each server to be shown by a separate state), (2) fail-over server, and (3) load balancer.

At least two active servers are required to demonstrate active-active mode, one passive server along with one active server to demonstrate active-passive configuration and a fail-over server should be present for fulfilling the requirement of 24 X 7 availability. The events involved are shown as transitions in the model (Fig. 1). The model represents both active-active and active-passive architectures.

For active- active three load balancer configurations are considered - load distribution in the ratio of (50-50), (60-40) and (70-30) and are shown with transitions $LD{-}1{-}1$, $LD{-}3{-}2$, $LD{-}7{-}3$ in Fig 1.

2. The WSM Description

Fig. 1 shows the proposed WSM. The upper portion of the model depicts active-active architecture whereas the lower portion covers active-passive with possible fail-over mechanism.

The fail-over server is not shown separately in active-active because if one server fails the load balancer distributes the requests among the remaining working servers. Table I gives the description of all the states and transitions that are used in WSM.

In active-active mode, only one of the transitions of $LD{-}1{-}1$, $LD{-}3{-}2$, $LD{-}7{-}3$ will be active at any given point in time. Requests will be distributed to the two active servers as per the chosen ratio by the load balancer.

TABLE I
PLACES AND TRANSITIONS USED IN THE PROPOSED MODEL

Web Component	Petri-net Component	Description
Requests		This represents the portal or client side from which requests are sent.
LB		The LB place represents the load bal- ancer that stores the requests from concurrent users for further distribu- tion among various servers.
CR or T0		The CR transition represents the con- current requests from the users.
ActServ-1		The ActServ-1 is the active server in the active-active configuration of load balancer.
		The two active servers are shown in the active-active mode in the model among which load is distributed by load balancer.
		This is also one of the active servers in the active-active configuration of load balancer.
	Place	This is the transition that represents equal distribution of load among two servers in active-active configuration.
	Place	This is the transition that represents 60-40 distribution of load among two servers in active-active configuration.
ActServ-2		This is the transition that represents 70-30 distribution of load among two servers in active-active configuration.
LD-1-1	Transition	This is the event represented by tran- sition that leads to selection of active- passive configuration.
LD-3-2	Place	The ActServ is the active server of the active-passive configuration that han- dles the concurrent requests from the users forwarded by the load balancer.
LD-7-3		This is the passive server of the active- passive configuration that handles the concurrent requests from the users for- warded by the load balancer in case of active crash. This is designed for fail back scenarios.
AP-Normal	Place Transition Transition Transition Place	This is the event when capacity of active server is full and is unable to handle requests any more.
ActServ	Place	This the new active server added to the cluster in active-passive configuration when active full event is generated.
PassiveServ	Transition Place Place Transition Transition Transition Place Transition	This represents the notification state when active server in active-passive configuration crashes.
		This is the event to notify that the further requests are to be handled by passive server.
		The hardware failure event is mod- elled as transition that notifies the ac- tive server crash.
ActFull NewActServ ActServCrash TransPassiveServ HW-Failure DBConn-1 DBConn-2	Transition Place Transition Transition Place Place	The DBConn-1 represents the database connectivity between active- active server and database server.
		The DBConn-2 represents the database connectivity between active- active server and database server.
DBReq		The concurrent requests that requires the database access are represented by the place database requests.
		The read queries are notified by RdReq event.
		The write queries are notified by Wr- Req event.
RdReq WrReq DBServ-1 RdOnly Write DBServRd DBServWr		The database server that handles the both read and write queries.
		The read only queries that goes to the database server for read requests.
		The queries that goes to the database server for read and write requests.
		The database server that handles only read queries.
		The database server that handles both read and write queries.

The ratio can be chosen based on the active server specifications. Only two servers are being shown in the model, however multiple servers can be considered to cater large number of concurrent requests.

In active-passive mode, which is captured in the lower portion of the model, four places and four transitions are used. These places and transitions capture hardware failure of the active server and active server capacity getting full. Transition AP−Normal selects active-passive over active-active when triggered. The requests are then sent to the ActServ rather than ActServ−1 or ActServ−2. Whenever this server fails, transition H W−f ailure triggers and the requests are forwarded to PassiveServ. NewActServ is enabled when the capacity of the ActServ is FULL.

TABLE II
TOKENS USED

Token type	Token colour	Representation
Concurrent users	Red	one red token represents1000 Requests
Web Servers	Black	one black token represents one web server
Write operation	Blue	one blue token represents 500 writes
Read operation	Green	one green token represents 500 reads
Hardware failure	Purple	

The transitions used are only immediate transitions that can fire immediately when gets enabled. As multiple transitions are possible on the event, it is required to assign priority to some transitions over other. The immediate transition makes it possible to assign priorities. Thus, in the active-active model *LD*−1−1 is given more priority than the other two configurations. Similarly, *AP*−*Normal* is given priority over all *LD*s.

The transitions are triggered by the tokens. Table II shows the tokens and their colours that are used in modelling the web server architecture. For example, concurrent user requests are represented by red colour tokens. One red token represents 1000 concurrent users. The place *Requests* has 10 tokens which means 10,000 concurrent users. These requests are distributed by load balancer either among active-active server or to active-passive server depending upon the enabled transition. When one red token which represents concurrent requests is released, *AP-Normal* transition is enabled which is fired to send the requests to *ActServ* in active-passive configuration. When number of requests increases to 4000 shown by the 4 red tokens on the arc from *ActServ* to *ActFull* transition, transition *ActFull* gets activated and *N ewActServ* is added. When two red tokens are released the transition *LD*−1−1 is enabled. If *LD*−1−1 is chosen the requests are distributed 50-50% i.e. one red token from *LD*−1−1 to *ActServ* −1 and one for *ActServ* − 2 representing 2000 requests that are divided equally as 1000 to each. When five red tokens are released *LD*−3−2 is enabled it sends 3 red tokens to *ActServ*−1 and 2 red tokens to *ActServ* − 2 representing 3000-2000 distribution of 5000 requests and *LD* −7−3 is enabled when 10 red tokens are released, it sends 7 and 3 tokens to *ActServ*−1 and *ActServ*−2 respectively representing 7000-3000 distribution of 10,000 requests.

3. Simulation

Simulation is an experimentation with a model of a system. A CPN model is an executable representation of a system comprising of the states and the events or transitions that

causes the system to change its state and move to another state based on the type of event. It is possible to analyze and explore various scenarios and behaviour of a system using simulations provided by CPN Model.

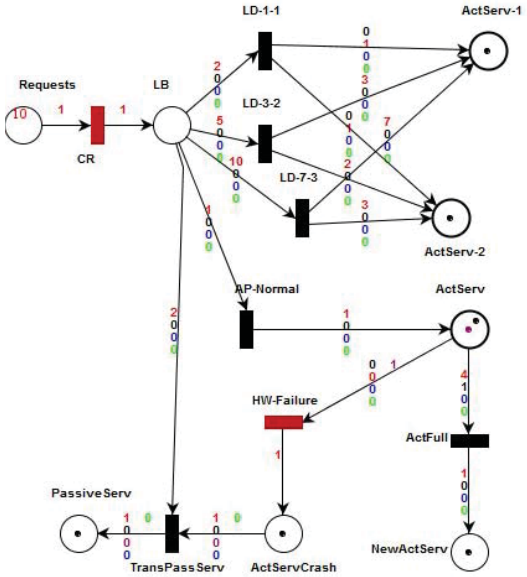

Fig. 2 Hardware Failure in Active-Passive mode

One such scenario of WSM is shown in Fig. 2. When *ActServ* is full, a new node called *NewActServ* is added to the cluster. The *PassiveServ* is defined by place that is continuously listening to the *ActServ*. In case *ActServ* fails, the *PassiveServ* takes over. Fig. 2 shows this simulated scenario. The event hardware failure is also defined in the model by a transition to show this scenario. The presence of purple dot in place *ActServ* activates the hardware failure event showing the condition of active-crash. Henceforth, request to transition *TransPassServ* is activated and the requests get forwarded to *PassiveServ*. The analysis of this model is covered in Section IV.

B. Database Server Sizing Model

Similar to architectures of web servers, different architectures are followed for database servers as well. Database server sizing model (DSSM) captures these architectures.

To provide 24x7 availability and zero data loss, database servers are either designed in cluster with one or more active and/or passive servers, or in active-active configuration similar to that of web servers. Active-active configuration is more flexible and cost effective. In databases like Oracle both these configurations are supported. However, in PostgreSQL, which is the focus of this paper, only active-passive configuration is supported. This becomes a limiting factor when a large number of concurrent users need to be catered.

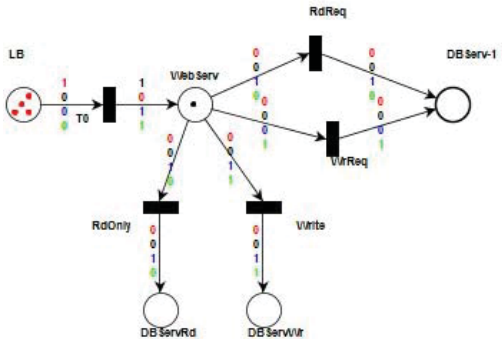

Fig. 3 DSS Model

For considering sizing of database server requires the type of SQL queries being executed on the database. To start with we experimented using the basic read and write type of SQL queries for suggesting the basic model for database server sizing. The proposed model is referred as Database Server Sizing Model (DSSM) and is shown in Fig. 3.

1. Places, Transitions, and Tokens

The states involved in this activity are load balancer *LB*, web server *WebServ* and database servers *DBServ*−1, *DBServRd*, *DBServWr*. These are defined by places in DSS. To understand the impact of concurrent users on servers, one of the parameters that should be present in the model is number of concurrent users. Concurrent users are taken as red colored tokens shown in *LB*. In database queries contains two types of operations, read, and write, on which the performance of the database server is decided. Blue and green coloured tokens used to represent the type of SQL queries read or write) being executed on the database.

Queries like “SELECT” are considered as read query which does not modify the content of the database whereas “UPDATE”, “INSERT” or “DELETE” queries are written queries where data may change with execution of these queries. Simple queries of both the types are considered while proposing the model.

2. DSSM Description

One red token represents 1000 concurrent users. Model in Fig. 3 thus shows that the chosen web server can handle 1000 concurrent users. For large number of concurrent users’ multiple servers should be clustered through a load balancer. The application chosen has read and write requests for databases which are considered equiprobable currently. However, this can be modified as per the application. When 1000 concurrent users event *T* 0 is fired, it gets converted to one black token (additional web server), one green (500 read) and one blue (500 write) tokens. Events *Rdreq* and *WrReq* consume the concurrent read and write requests. Current postgreSQL server was supporting 2000 concurrent connections according to the experimentation thus 1000 reads and 1000 writes. Hence, the architecture shown in the Fig. 3 was sufficient.

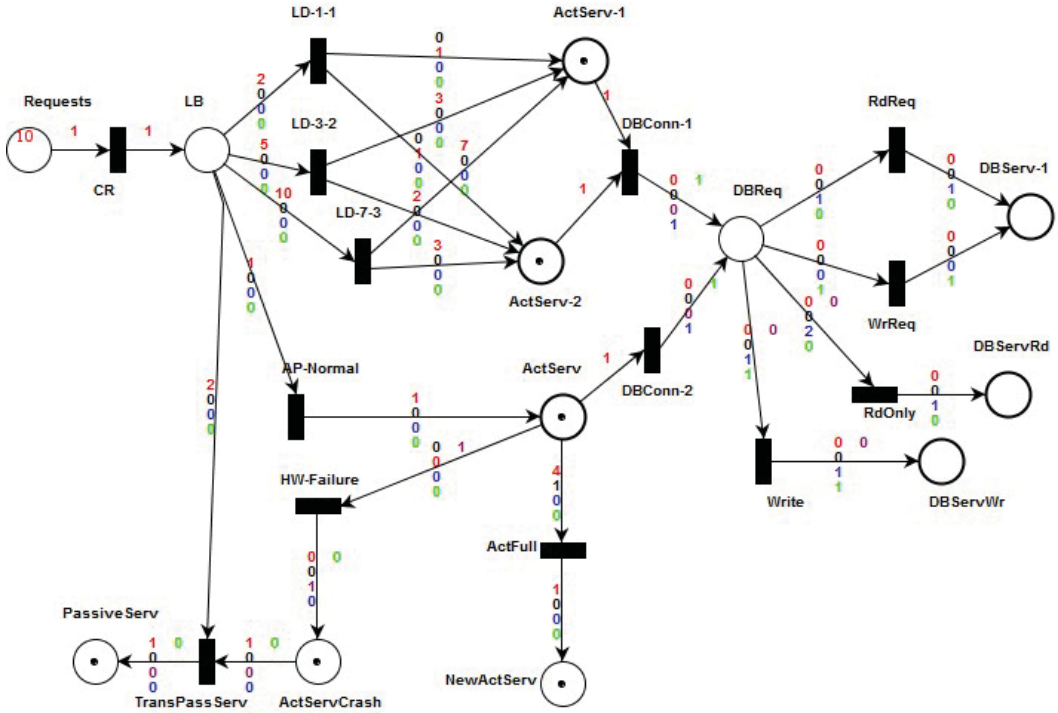

Fig. 4 CPN Model for Web Application Architecture

The number of concurrent users for a given application keeps on increasing and so as the number of read and write queries. After serving a certain amount of connections a thresh- old point reaches for the database server which is determined through experimentation discussed in Section IV. To avoid the risk of failure, one of the solution is to add two database servers with one database server for read requests and another database server for handling write requests as shown in Fig. 3. Henceforth all the read requests can be handled by newly added DB ServRd and all the write requests by the DB ServWr. For the architecture depicted by the model in Fig. 3, the code needs to be modified at the web server end so that the read requests will be directed to separate server and write requests to another database server. In addition to this, data on both the servers should remain in sync and hence, appropriate synchronization technique needs to be adopted. In the experimentation, we implemented data synchronization through synchronous replication feature provided by PostgreSQL 9.3, hence current proposed model will be for PostgreSQL.

3. Experimentation

Experiments were carried out on PostgreSQL Database Server to find out the threshold in terms of maximum number of connections a server can handle. JMeter, a load simulation tool was being used to generate the required load. The number of connections per second were increased gradually to verify the scalability of the DB server i.e. maximum number of connections it can handle without giving any error. Transactions started at database server shall have no error, i.e. success rate for all transactions must be 100%.

The experimental set-up had two Virtual Machines with 2 QEMU virtual CPU version (cpu64) 2.67 GHz and 4MB Cache each, 4GB RAM and 4GB of Swap Memory. Both the servers were having CentOS version 6.4 (64 bit), Kernel: 2.6.32-358.el6.x86 64 GNU Linux, Java Version 1.7 and JMeter version 2.8 installed and configured. Minimal modifications in PostgreSQL configuration file were done to ensure optimal use of hardware resources: (1) shared buffers (Identifies the need of cache memory for PostgreSQL) = 4096 MB (default value 128 MB); (2) max_connections = 2000 (default value 100). As discussed earlier two types of database queries, SELECT (Read Operation) and UPDATE (Write Operation) were used for basic experiment. The database stores registration records of lacs of students for a national level examination. We have created a table with name Applicant which stores registration information for the applicant. The table only stores numeric and alphanumeric data and has 43 fields. The size of Applicant table with data is around 3 GB and contains 10 million application's data, and average row size is 300 bytes. There is a Primary Key Index on ApplicationID column. Queries used are SELECT * FROM applicant WHERE applicationid=? and UPDATE applicant SET score = score * 1.10 WHERE applicationid=?

The objective of the experimentation was to determine threshold in terms of a maximum number of concurrent connections a server can handle. The threshold for the server chosen for experiment reached at 3000. This experiment result is used to design the database server sizing model(DSSM). Analysis of the model is given in Section IV.

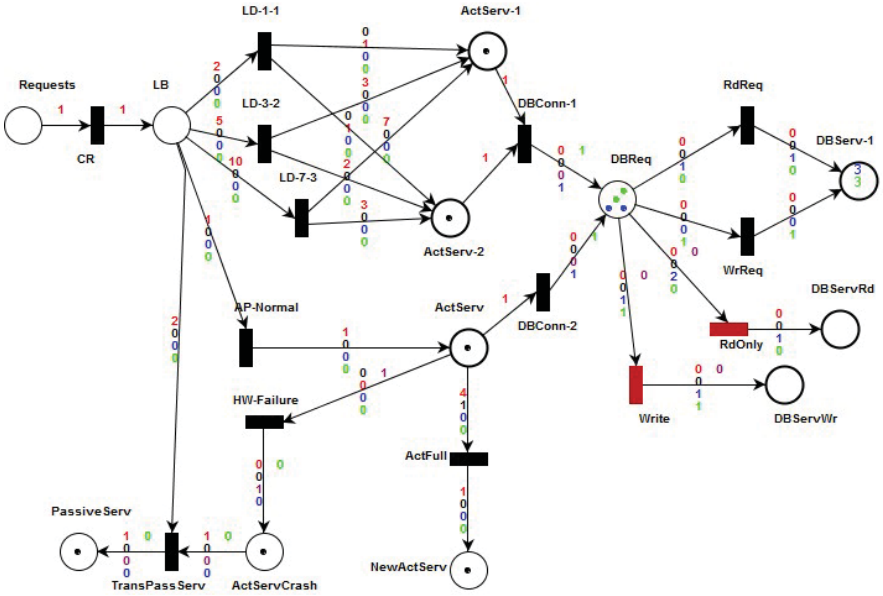

Fig. 5 Simulation

C. CPN Model for Web Application Architecture

Fig. 1 showing Web Server Model and Fig. 3 showing Database Server Sizing Model are integrated to form the CPN Model for Web Application Architecture. Two models are connected by DB Connection as shown in Fig. 4.

Two different transitions *DBConn*−1 and *DBConn*−*2* are used to connect to *DBReq* instead of using a single transition. If a single transition is used, it will be active when both connections are one i.e. AND operation. However, as only one of the configurations will be active at any given point in time the transition will never be executed. To avoid the representation of this wrong scenario two different transitions are used.

1. Simulation

One of the Simulations run from the CPN Model for Web Application Architecture is shown in Fig. 5. When threshold for *DBServ*−1 reaches, the requests are directed to the *DBServRd* and *DBServWr* through enabled transitions *RdOnly* and *W rite*.

IV. Analysis

The model should adhere to the basic properties of Petri Nets like Reachability, Cover-ability, Boundedness. To cover behavioural and structural properties of Petri-nets, behavioural and structural analysis methods are applied to the model. The Cover-ability (Reachability) represents behavioural aspect A. Kostin [22] while Incidence matrix, Minimal Siphon and Minimal Trap, and Invariant analysis characterises the structural aspect of Petri Net Models.

A. Analysis of Web Server Model

Web Server Model (WSM) belongs to asymmetric choice class. This class is also known as *Simple Net*. The two properties of the asymmetric choice net on the basis of which Web Server Model is classified as *Simple Net* are conflict transitivity and place liveness. If either *e*1 or *e*2 can occur but not both then two events are in conflict , If both events can occur in any order without conflicts then they are concurrent.

In the asymmetric choice net any pair of transitions among transitions in choice are in a conflict relation. At a time, we can have only one of the configurations activated or enabled. There is a choice. But we cannot use all configurations together. For example, in WSM, transition *LD* −1−1 is in conflict with *LD*−3−2 and *LD*−3−2 is in conflict with *LD*−7−3. Again *LD*−7−3 is in conflict relation with *LD*−1−1. Similarly, active-active configuration is in conflict relation with active-passive.

Calculations of Invariant analysis for Web Server Model are done using Farkas algorithm [23]. After analysis, only one row with all zero entries remains corresponding to which there are non-zero entries in the augmented side of matrix obtained from incidence matrix analysis shown in Fig. 6. There is only one *P* invariant in the Web Server Model shown in Fig. 7, which proves the boundedness of the model. The *P* invariant obtained is used to calculate the weighted sum of tokens. The number of tokens at each place is multiplied by the weightage obtained from the *P* invariant and the sum should be constant to prove the boundedness of the Petri net.

WSM is a bounded Petri Net hence there is reachability graph. There is no need of coverability graph. As we use coverability graph when Petri net is unbounded. Reachability graph analyses the reachability of various server places in the Web Server Model. It ensures that requests from the users are reaching the servers.

B. Analysis of DSS Model

The Database Server Sizing(DSS) Model belongs to *State Machine* class as each transition has exactly one input and one output. *Asymmetric Choice Net* has a marking at which one transition is enabled while other is disabled in a choice. Hence, DSSM belongs to *Asymmetric Choice Net* class as well.

In DSSM where priority is given to read and write query, transitions going to common DB Server over only read and only write transitions which are going to separate servers. So, when token comes at place *DBReq*, read query and write query transitions are enabled while only read and only write remains disabled. When threshold reaches for *DBServ*−1, *RdOnly* and *Write* transitions are enabled so as to redirect the requests henceforth.

The incidence matrix is obtained from incidence matrix analysis which is used to perform invariant analysis. We have analysed that after performing the invariant analysis there is only one P-invariant in the DSS Model, as there is only one row left with all the zeros and corresponding non-zero entries in the augmented side of matrix which gives the places that forms the P-invariant.

In the model four tokens at place *LB* and one token at *webserver* place makes total of five tokens, while zero tokens at other places as shown in Fig. 9. The DSS Model consists of a place *LB* as minimal siphon as this is the place in Petri Net which once becomes empty (free of tokens) in the marking then never gains the tokens. The minimal trap can be from either of the places *DBServ*−1, *DBServRd*, *DBServWr*. These are the places in the Petri Net which once gains token in the marking then never lose it according to the definition of trap. The Database Server Sizing Model is a bounded Petri Net and hence there is reachability graph.

C.Analysis of CPN Model for Web Application Architecture

The integrated performance model falls into category of *Asymmetric Choice Net* due to the property of conflict transitivity. The asymmetric choice of transitions *LD*−1−1, *LD*−3−2, *LD*−7−3 and active-passive in WSM, also asymmetric choice of transitions *RdReq*, *WrReq* and *RdOnly*, *W rite* makes the model *Asymmetric Choice Net*. As per the invariant that the integrated CPN Web Application Architecture Model is also bounded as shown in Figs. 10 and 11.

The integrated model consists of place *Requests* as minimal siphon, as this is the place in Petri Net which once becomes empty (free of tokens) in the marking then never gains the tokens.

Petri net incidence and marking

Forwards incidence matrix I^+

	ActFull	LD-1-1	LD-3-2	LD-7-3	AP Normal	HW- Failure	TransPassServ	CR
ActServ	0	0	0	0	1	0	0	0
ActServ-1	0	1	3	7	0	0	0	0
ActServ-2	0	1	2	3	0	0	0	0
PassiveServ	0	0	0	0	0	1	0	0
LB	0	0	0	0	0	0	0	1
NewActServ	1	0	0	0	0	0	0	0
ActServCrash	0	0	0	0	0	0	1	0
Requests	0	0	0	0	0	0	0	0

Backwards incidence matrix I^-

	ActFull	LD-1-1	LD-3-2	LD-7-3	AP Normal	HW- Failure	TransPassServ	CR
ActServ	3	0	0	0	0	4	0	0
ActServ-1	0	0	0	0	0	0	0	0
ActServ-2	0	0	0	0	0	0	0	0
PassiveServ	0	0	0	0	0	0	1	0
LB	0	2	5	10	1	0	1	0
NewActServ	0	0	0	0	0	0	0	0
ActServCrash	0	0	0	0	0	0	0	0
Requests	0	0	0	0	0	0	0	1

Combined incidence matrix I

	ActFull	LD-1-1	LD-3-2	LD-7-3	AP Normal	HW- Failure	TransPassServ	CR
ActServ	-3	0	0	0	1	-4	0	0
ActServ-1	0	1	3	7	0	0	0	0
ActServ-2	0	1	2	3	0	0	0	0
PassiveServ	0	0	0	0	0	1	-1	0
LB	0	-2	-5	-10	-1	0	-1	1
NewActServ	1	0	0	0	0	0	0	0
ActServCrash	0	0	0	0	0	0	1	0
Requests	0	0	0	0	0	0	0	-1

Inhibition matrix H

	ActFull	LD-1-1	LD-3-2	LD-7-3	AP Normal	HW- Failure	TransPassServ	CR
ActServ	0	0	0	0	0	0	0	0
ActServ-1	0	0	0	0	0	0	0	0
ActServ-2	0	0	0	0	0	0	0	0
PassiveServ	0	0	0	0	0	0	0	0
LB	0	0	0	0	0	0	0	0
NewActServ	0	0	0	0	0	0	0	0
ActServCrash	0	0	0	0	0	0	0	0
Requests	0	0	0	0	0	0	0	0

Marking

	ActServ	ActServ-1	ActServ-2	PassiveServ	LB	NewActServ	ActServCrash	Requests
Initial	1	1	1	1	0	1	1	9
Current	1	1	1	1	0	1	1	9

Enabled transitions

ActFull	LD-1-1	LD-3-2	LD-7-3	AP Normal	HW- Failure	TransPassServ	CR
no	no	no	no	no	no	no	yes

Fig. 6 Incidence matrix analysis of Web Server Model

Petri net invariant analysis results

T-Invariants

ActFull	AP Normal	CR	HW- Failure	LD-1-1	LD-3-2	LD-7-3	TransPassServ

The net is not covered by positive T-Invariants, therefore we do not know if it is bounded and live.

P-Invariants

ActServ	ActServ-1	ActServ-2	ActServCrash	LB	NewActServ	PassiveServ	Requests
2	2	2	10	2	6	8	2

The net is covered by positive P-Invariants, therefore it is bounded.

P-Invariant equations

2M(ActServ) + 2M(ActServ-1) + 2M(ActServ-2) + 10M(ActServCrash) + 2M(LB) + 6M (NewActServ) + 8M(PassiveServ) + 2M(Requests) = 50

Analysis time: 0.0s

Fig. 7 Invariant analysis of Web Server Model

The minimal trap can be from either of the places *DBServ*−1, *DBServRd*, *DBServWr*. These are the places in the Petri net which once gains token in the marking then never lose it. The Integrated CPN Web Server Model is a bounded Petri Net hence there is reachability graph.

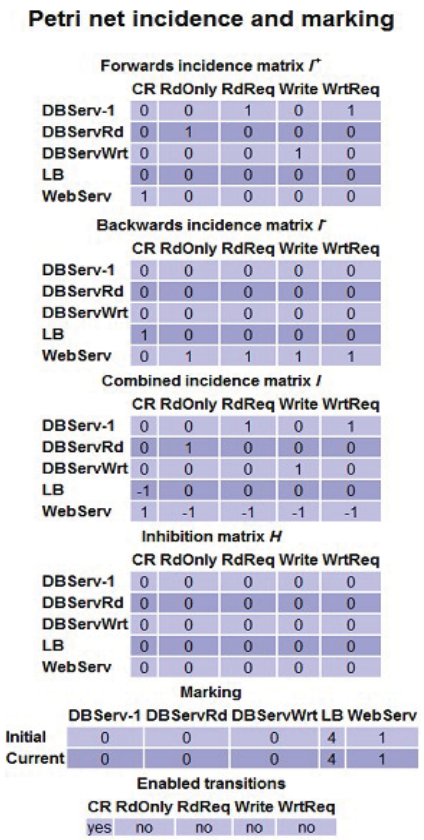

Fig. 8 Incidence matrix analysis of DSS Model

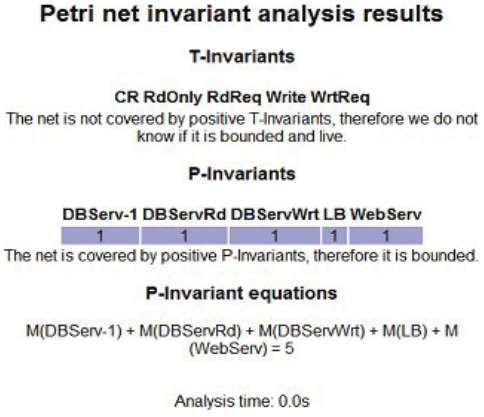

Fig. 9 Invariant analysis of DSS Model

V. Conclusions

The paper proposed a model for web and database server sizing, and architecture change based on the varying concurrent users. The model focuses on need to restructure the architecture on server side so as to improve the performance. The proposed model mainly identifies how the increase in number of concurrent users and type of query affects the sizing and architecture of the web and database servers. It also helps in bringing out the appropriate architecture for web and database servers given the number of concurrent users and type of queries. The proposed model for Database is based on the experimentation on PostgreSQL and hence considers architectures supported by PostgreSQL only.

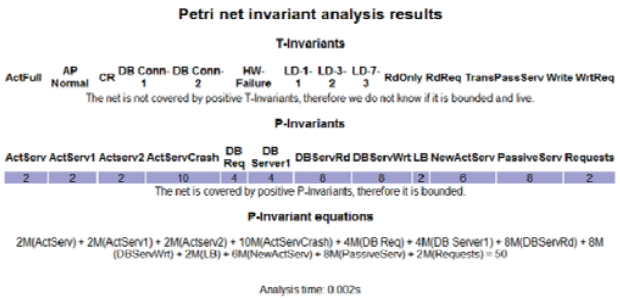

Fig. 10 Invariant analysis of Web Application Architecture Performance

The proposed model currently assumes specifications for servers and bandwidth. In future, we would like to extend the model to modify the architecture based on server configuration and available bandwidth to cater the concurrent users.

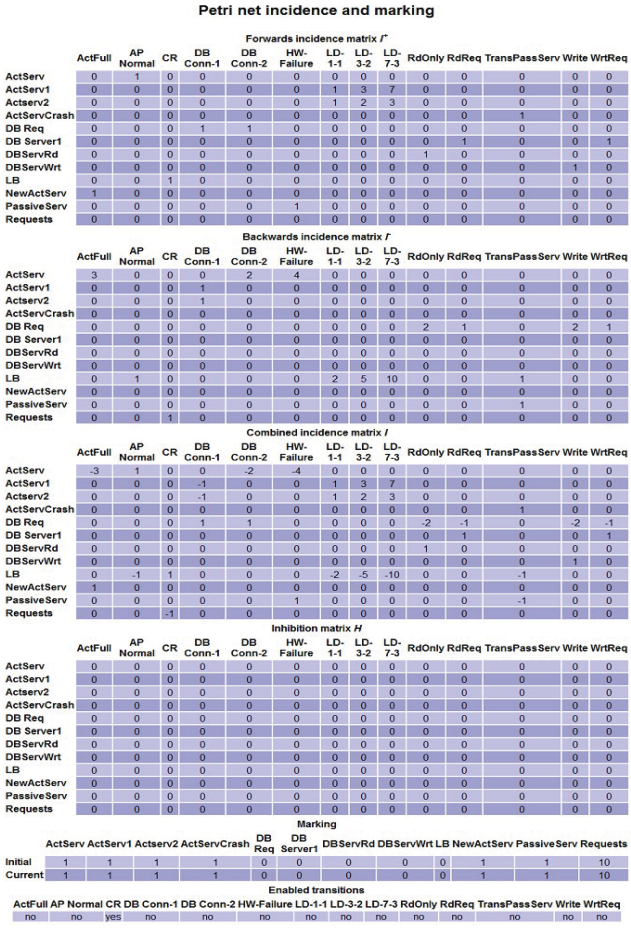

Fig. 11 Incidence matrix analysis of Web Application Architecture Performance Model

REFERENCES

[1] J. T. Vicenc Beltran and E. Ayguade, "Improving web server performance through main memory compression," *14th IEEE International Conference on Parallel and Distributed Systems*, 2008.
[2] D. K. Mahnaz Shams and B. Far, "A model-based approach for testing the performance of web applications," *Proceedings of the Third International Workshop on Software Quality Assurance*, 2006.
[3] S. M. Manish Marwah and C. Fetzer, "Fault-tolerant and scalable tcp splice and web server architecture," *25th IEEE Symposium on Reliable Distributed Systems*, 2006.
[4] R. H. Zhong Xu and L. N. Bhuyan, "Load balancing of dns-based distributed web server systems with page caching," *Proceedings of the Tenth International Conference on Parallel and Distributed Systems*,2004.
[5] E. Casalicchio and S. Tucci, "Static and dynamic scheduling algorithms for scalable web server farm."
[6] B. Spitznagel and D. Garlan, "Architecture based performance analysis," *Proceedings of 1998 Conference on Software Engineering and Knowledge Engineering*, pp. 146–151, 1998.
[7] Q. Zhaoyang, W. wei, and L. Zhiqian, "Web server optimization model based on performance analysis," *In IEEE Proceedings of 6th International Conference on Wireless Communication Networking and Mobile Computing*, pp. 1–4, 2010.
[8] J. Zhang, C. Chang, S. Kim, and J. Y. Chung, "Ws-net: A petri-net Model based specification model for web services," *In IEEE of International Conference on Web Services*, 2004.
[9] S.Kounex and A.Buchhman, "Performance modelling of distributed e-business applications using queuing petri nets," *IEEE Proceedings*, 2003.
[10] X. Hai-yan and W. Yan, "Workflow model based on stochastic petri nets and performance evaluation," *IEEE International Symposium on IT in Medicine Education*, vol. 1, pp. 245–249, 2009.
[11] J. Wang, *Petri nets for dynamic event driven system modeling*, 2007.
[12] S. Samolej and T. Rak, "Simulation and performance analysis of distributed internet systems using tcpns," *Informatica*, vol. 33, pp. 405–415, 2009.
[13] L. M. K. Lisa Wells, Sprren Christensen and K. H. Mortensen, "Simulation based performance analysis of web servers," *IEEE*, 2001.
[14] C A Petri, "Kommunikation mit automaten," *Schriften des Rheinisch-6Westflischen Institutes fr Instrumentelle Mathematik an der UniversittBonn Nr. 2*, 1962.
[15] T. Muratha, "Petri nets:properties analysis and applications," *Proceedings of IEEE*, vol. 77, no. 4, pp. 541–580, November 1989.
[16] R. Zurawaski and M. Zhou, "Petri nets and industrial applications: a tutorial," *IEEE Transactions on Industrial Electronics*, vol. 41, no. 4, pp. 567–583, December 1994.
[17] D. G. M. Heiner and R. Donaldson, "Petri nets for systems and synthetic biology," *Formal Methods for Computational Systems Biology, LNCS,2008*, vol. 5016, pp. 215–264, 2008.
[18] K. Jensen, "Coloured petri nets. basic concepts, analysis methods and practical use," *Basic Concepts. Monographs in Theoretical Computer Science, Springer-Verlag*, vol. 1, 1997.
[19] PIPE-2, "http://pipe2.sourceforge.net/," 2007.
[20] A. Charalambous, "Extension of pipe2 to support coloured generalised stochastic petri nets," *PhD thesis, Imperial College of London*, 2010.
[21] R. P. Pere Bonet, Catalina M. Llado, "Pipe v2.5: a petri net tool for performance modeling."
[22] A. Kostin, "A reachability algorithm for general petri nets based on transition invariants," *Springer-Verlag, ISBN 978-3540377917, Lecture Notes in Computer Science*, vol. 4162, pp. 608–621, 2006.
[23] J. Farkas, "Theorie der einfachen ungleichungen," *ur die Reine und Angewandte Mathematik*, vol. 6, pp. 124–127, 1902.