ISA Transactions 84 (2019) 225–236

Contents lists available at ScienceDirect

ISA Transactions

journal homepage: www.elsevier.com/locate/isatrans

Practice article

Load balanced scheduling and reliability modeling of grid transaction processing system using colored Petri nets

Dharmendra Prasad Mahato [a,*], Ravi Shankar Singh [b]

[a] Department of Computer Science and Engineering, BIT Sindri, Dhanbad, 828123, India
[b] Department of Computer Science and Engineering, Indian Institute of Technology (BHU), Varanasi, 221005, India

HIGHLIGHTS

- Reliability Modeling and Analysis has been carried out; first analytically and secondly by CPNs Tool.
- Load Balanced Scheduling Modeling has been carried out by CPNs Tool.
- The necessity of On-demand Computing System when an enterprise faces unlimited transaction processing.
- All the modeling has been carried out by Colored Petri nets Tool. This paper also describes the procedure of the modeling step by step.

ARTICLE INFO

Article history:
Received 22 April 2018
Received in revised form 31 July 2018
Accepted 20 August 2018
Available online 12 October 2018

Keywords:
Reliability modeling
Reliability analysis
Load balanced scheduling
Colored Petri nets

ABSTRACT

On-demand computing is a popular enterprise model in which the computing resources are made available to the users as needed. On-demand computing based transaction processing system which has grown rapidly in recent years is an information processing system with the stringent requirements of resources to meet the fluctuating demands. Concepts such as grid computing, utility computing, autonomic computing, and adaptive management seem very similar to the concept of on-demand computing. When demands of resources fluctuate, the system needs load balancing for the efficient utilization of the computational resources. Furthermore, scheduling is needed to assign the transactions to the appropriate resources. Thus, modeling of load balanced scheduling along with reliability analysis for this system is a challenging task.

This paper presents the load balanced scheduling and reliability modeling in such an environment by using colored Petri nets (CPNs). CPNs which combine Petri nets with programming languages is a powerful modeling technique. The proposed CPN-based modeling pattern formally describes the process of transaction distribution and execution within the on-demand computing environment. Moreover, the CPN-based model uses the hierarchical modeling capability of CPNs, including different levels of abstraction (sub-modules). This helps easily handling and extending the model. Since, on-demand computing based transaction processing system executes a number of concurrent transactions. The CPN-based model is extended to express the concurrency, thus improving the reliability results. This paper takes the example of grid transaction processing (GTP) system with the problem of load balanced scheduling modeling and reliability evaluation.

1. Introduction

On-demand computing based transaction processing e.g. grid transaction processing is a largely invisible back-end business computing function [1–9]. However, it is a key application used for an enormous range of economic activities, from travel reservation, ticketing systems and electronic banking, to financial transactions and e-commerce. It certainly has become a critical component of the world's economic engine which is responsible for handling trillions of transactions every year. Because, the never-ending growth of the global economy along with e-business and web-based commerce are placing more demands on transaction processing systems. New trends and emerging technologies, such as the increasing use of mobile messaging technology and the employment of Radio Frequency Identification (RFID) tracking, are generating more transaction traffic. Therefore, the workload volume managed by on-demand computing based transaction processing systems is enormous.

When choosing an on-demand based transaction processing strategy, corporations must rely on solutions that ensure data availability, reliability, performance, and scalability. Thus the load

* Corresponding author.
E-mail addresses: dpmahato.rs.cse13@iitbhu.ac.in (D.P. Mahato), ravi.cse@iitbhu.ac.in (R.S. Singh).

https://doi.org/10.1016/j.isatra.2018.08.022
0019-0578/

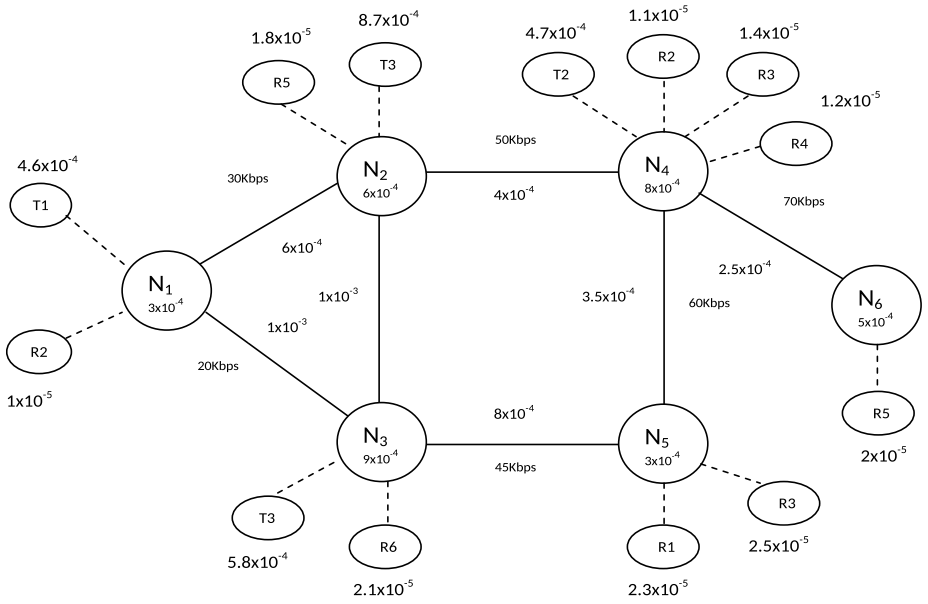

Fig. 1. Topology of the network, transactions, and resources (This image is taken from [10]). The deadline-miss failure rate is ψ_D throughout the problem.

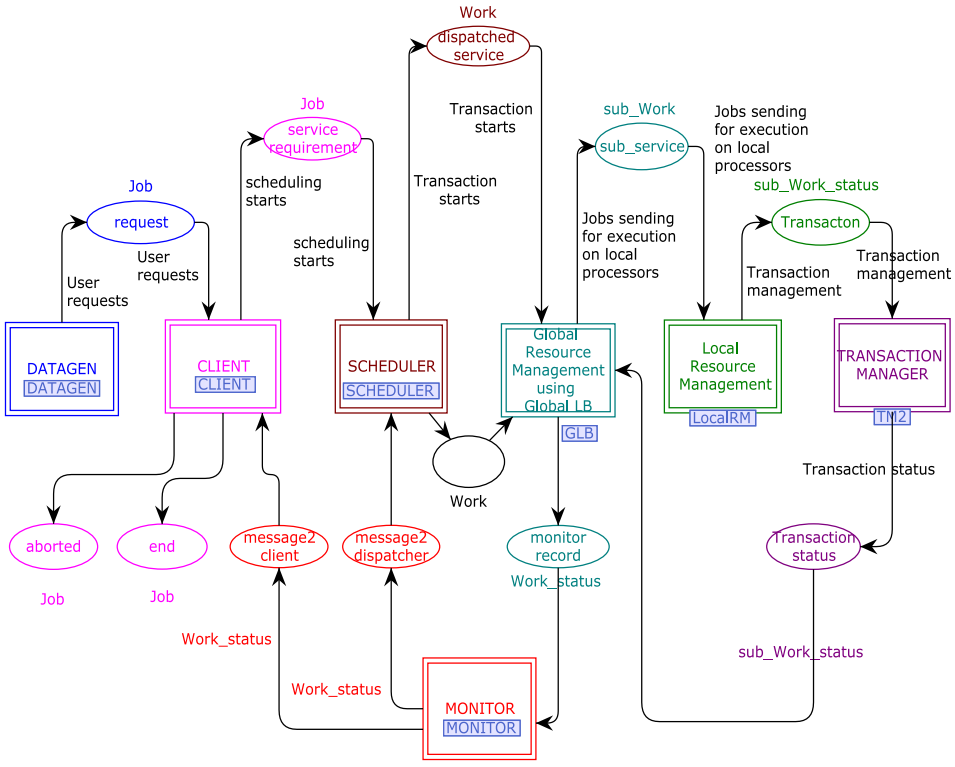

Fig. 2. CPN model of GTP.

balanced scheduling and reliability modeling of the system is a need.

The execution of on-demand computing based transaction generally requires communication paths between several on-demand resources and access to them. The reliability, i.e. probability that an on-demand computing based transaction can run successfully, depends on the reliabilities of communication links, softwares (middleware or distributed operating system), and processing elements as well as on the distribution of files. In order to have a meaningful and effective analysis of the system, the modeling or representation must be easy for fault detection and diagnosis techniques [11–15].

For reliability analysis, a probabilistic graph $G(V, E)$ is usually required to represent a system where V is a set of nodes which represent computers in the case of distributed processing systems and E is the set of directed or undirected arcs representing the communication links. But the probabilistic graphs are static in nature. They are not able to model and analyze the dynamic behavior of on-demand computing based transaction processing system. Owing to the dynamic and real-time behavior of the system, a powerful modeling and analysis approach is required.

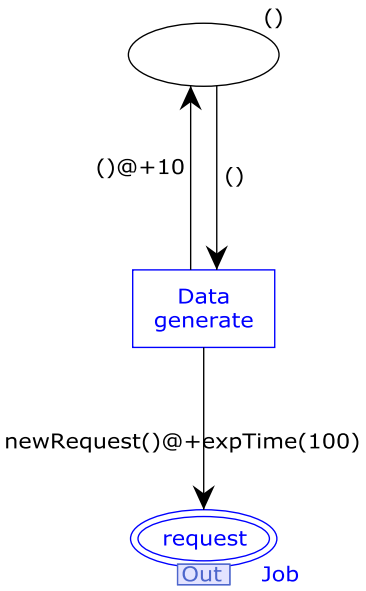

Fig. 3. CPN model to generate transactions.

Modeling of any system allows the designers to inspect and learn about the behavior of the system prior to implementation. In this way, the designer can find out many design problems and errors early in the system development phase. However, the modeling of the complex system such as on-demand based transaction processing system is not an easy task.

Petri nets based tools are recognized as the most powerful modeling and analysis approach for the systems like event condition occurrence system, semaphore systems, digital computers under the control of distributed operating system, transaction processing systems and communication protocols, etc. [16]. The transaction processing follows the parallel and real-time execution. Therefore, the advanced version of Petri nets are most capable to model the parallel and real-time execution of the transaction. CPNs is the latest and advanced version of Petri nets.

This paper uses CPNs to model the load balanced scheduling and reliability in the Grid Transaction Processing (GTP) system. The instance of the proposed model for the GTP system using CPN Tools is analyzed. The main contributions of this paper are two-fold. First, the load balanced scheduling with reliability is designed analytically to represent transaction oriented grid architecture. In reliability modeling, different faults are considered. Along with those faults, deadline-miss fault is also considered. Second, the reliability of the GTP system is modeled and analyzed using CPN Tools.

The rest of the paper is organized as follows. Section 2 reviews the related research work and discusses background information. In Section 3, the overview of CPNs is given. Section 4 presents the reliability model. Section 5 details results and discussion. Section 6 contains conclusions and avenues of future research.

2. Background and literature review

2.1. Grid transaction processing

A transaction is a short sequence of interactions with the database, by using operations such as finding a record or by modifying an item, which represents one meaningful activity in the user's environment [17]. When the number of users increases beyond limit, on-demand computing based transaction processing may be used to solve the problem of scalability and completion

```
Standard declarations
colset UNIT = unit timed;
colset INT = int ;
var n,n1,n2:INT;
colset BOOL = bool;
colset STRING= string;
var str:STRING;
val MaxPro=16;
colset Pro=
index pro with 1..MaxPro;
var p,p1:Pro;
val MaxSub=10;
colset Sub=
index sub with 1..MaxSub;
val MaxClient=100;
colset ClientSet=
index client with 1..MaxClient;
var c,c1,c2:ClientSet;
var s:Sub;
colset Service_item=
with service_a|service_b|service_c;
var sit,sit1:Service_item;
colset Status=with announcing
|waiting|
received_not|received_del|
received_order
|processing|complete|
received_payment
|paid|pay_not|not_complete|aborted;
var st:Status;
var tsi, at, wt, pt, proctime,proctime1:INT;
colset Request=
product ClientSet*Service_item;
colset Job=product
Request*STRING*INT*INT*INT*INT timed;
var JB JBt
colset Job_status=product Job*Status;
fun RanC()=ClientSet.ran();
fun RanS()=Service_item.ran();
colset TS= int with 1..20;
fun RanT()=TS.ran();
colset Work=product Job*Pro;
var W:Work;
```

Fig. 4. CPN model of monitoring process in Grid.

time [18–21]. Transactions in e-commerce, VisaNet, PayPal and financial trading systems are the kind of transactions which are processed with the help of the on-demand computing. The peculiar nature of the transaction processing in this system is that transactions are computed and completed on various nodes in the system. Therefore, this system is composed of various service calls executed by different peers of the system [22].

When the number of transactions increases, it is obvious that processing nodes will face the problem of overloading. In order to resolve this problem load balanced scheduling technique is needed. The objective of transaction scheduling is to map transactions onto the required nodes and to order their executions so that transactions are successfully completed within their respective deadlines and in the meanwhile, the minimum schedule length (makespan) can be achieved [23].

Load balanced transaction scheduling is one of the factors which enhance reliability and performance in the GTP system. Several

```
var W:Work;
colset Transaction=
with Begin|Commit|
Abort|Rollback|
Confirm|Cancel|
Prepare|Not_Prepare
|success|Wait;
var tr:Transaction;
colset Sub_status=with rollback|
abort|prepared|committed|
return_committed|
executed|exception_occurred|
not_executed|fault_occurred
|no_fault|soft_fault|
hard_fault|comm_fault;
colset Work_status=
product Work*Status;
var Ws:Work_status;
colset WS=list Work_status;
var ws,ws1:WS;
colset Work_status_transaction=
product Work_status*Transaction;
var WST:Work_status_transaction;
colset Req_status=product Job*Status;
colset Sub_service_item=
index sub_service with 1..10;
var pod_it,pod_it1:Sub_service_item;
colset sub_Work=
product Work*Sub*Sub_service_item;
colset LS=list sub_Work;
var ls:LS;
colset sub_Work2=
product Work*Sub*Sub_service_item*INT;
var pod,pod1:sub_Work;
colset SW=list sub_Work;
var sw,sw1:SW;
colset sub_Work_Timed=
product sub_Work*INT;
var ps:Sub_status;
colset sub_Work_status=
product sub_Work_Timed*Sub_status timed;
var sws,sws1:sub_Work_status;
colset SWS=list sub_Work_status;
var swsl:SWS;
colset Dispatcherset=with d1;
```

Fig. 5. Declaration of the color sets.

```
colset D_Pro=
product Dispatcherset*Pro;
colset Ten0=int with 1..100;
colset Ten1=int with 1..100;
var u,u1,u2:Ten0;
var v,v1,v2:Ten1;
fun Ok(u:Ten0,v:Ten1)= (v<=u);
fun RanSs()=Sub_service_item.ran();
fun dispatch(sit:Service_item)=
case sit of service_a=>
pro(discrete(1,10))|service_b=>
pro(discrete(1,10))|service_c=>
pro(discrete(1,10));
globref packets = empty :sub_Work ms;
colset r=int with 1..10;
fun RanSp()=r.ran();
fun assign(((c:ClientSet,sit:Service_item),
str:STRING,tsi:INT,at:INT,wt:INT,pt:INT),p:Pro)=
let
val k= r.ran()
val counter = ref 1
val d = ref k
in
while !counter <= !d do
(
packets := !packets++1`((((c,sit),str,tsi,at,wt,pt),p),
sub(!counter),sub_service(!counter));
counter := !counter + 1
);
!packets
end;
fun initialize()=
let
val counter = ref 1
in
while !counter <= 1 do
(
packets := empty;
counter := !counter + 1
);
!packets
end;
val avg_proc_time = 150;
fun intTime() =
IntInf.toInt (time());
```

Fig. 6. Declaration of functions used in the modeling.

works have been done on reliability analysis in the literature. Tang et al. [23] designed a reliability-driven scheduling architecture that could effectively measure system reliability, based on an optimal reliability communication path search algorithm, and then they introduced reliability priority rank (RRank) to estimate the task's priority by considering reliability overheads. Furthermore, based on directed acyclic graph (DAG) they proposed a reliability-aware scheduling algorithm for precedence constrained tasks, which can achieve high quality of reliability for applications. Yen et al. [24] developed methods for assessing the reliability of these systems in terms of the timeliness and accuracy of the output, to aid decision making. They used two commonly used coordination structures, parallel and pipeline. Some adaptive schemes were proposed, i.e., strategies could be selected dynamically by agents at the run-time according to the input encountered in order to enhance the reliability of the system. The applicability of this approach was demonstrated by a practical multimedia client–server example. Shatz et al. [25], Kartik and Murthy [26], and Attiya and Hamam [27] proposed task allocation algorithm for maximizing the reliability of distributed computing system. Vidyarthi and Tripathi [28] proposed a task algorithm based on genetic algorithm for maximizing the reliability of the distributed system. Kang et al. [29] proposed task allocation algorithm for maximizing the reliability using honeybee mating optimization. Yin et al. [30] proposed the task allocation algorithm based on hybrid swarm optimization for maximizing the reliability. Pezoa et al. [31] modeled the service reliability based on random failures of the processing nodes. While Chen et al. [32] proposed task scheduling algorithm for maximizing the reliability of heterogeneous distributed systems considering fault recovery. while scheduling the tasks of processes/jobs with load balancing and without load balancing in different distributed computing environments including the grid system. In the GTP system, a transaction arrives at any node and may execute at several nodes before it commits [18–20]. Due to dynamic behavior of grid computing system, some nodes are added while some nodes are forced to leave the system. Hence, some nodes are heavily loaded

and some nodes are lightly loaded. Therefore, most of the transactions rollback or abort due to long waiting time at heavily loaded nodes [33]. The unavailability situation in this scenario arises. As a result, most of the transactions cannot be completed within their respective deadlines. Therefore, load balanced scheduling becomes important for transaction processing in grid computing system.

The load balanced scheduling can be accomplished by two methods;

- after scheduling the transactions
- before scheduling the transaction

The first method creates interprocessor overhead. Thus, the system is less reliable [34]. This approach is not suitable as the delay due to the interprocessor overhead in the system causes deadline-miss fault [35]. Whereas, the second method minimizes the delay in scheduling. The deadline-miss fault in this method is also less as compared to the former method of the scheduling. However, both the methods need a good scheduling technique [36]. Thus, load balanced transaction scheduling is such a method which schedules the transactions after balancing the loads of the assigned nodes. By using this method of scheduling, the availability of resources is maximized reducing the amount of downtime. Maximization of steady-state resource availability minimizes the occurrence of deadline-miss fault. Thus the minimization in the occurrence of deadline-miss fault may maximize the reliability of the system [37].

During the execution of transaction in the GTP system, various types of failures (as broadly discussed in [10]) may occur. These failures are software failure, blocking failure, time-out failure, matchmaking failure, network failure, resource failure. These failures have been considered in [10] for the definition and modeling of grid service reliability. Since each transaction has its deadline constraint, another type of failure similar to time-out failure caused by deadline-miss fault [35], may also occur in this condition. With the consideration of these failures, the reliability of the GTP system is defined as the probability that the set of actions belonging to transaction can be successfully executed.

3. An overview of colored Petri nets

The aim of this paper is not only modeling the load balanced transaction scheduling and workflow of the transaction execution but also evaluating the reliability. Colored Petri nets (CPNs) are appropriate for the modeling and analysis of reliability in on-demand computing based transaction processing system. CPNs combine the strengths of Petri nets with the facilities of high-level programming languages. Colored Petri nets (CPNs or CP-nets) are class of high- level nets that extend ordinary Petri nets. In CPNs, tokens can carry arbitrarily complex data, arcs can be annotated with input inscriptions influencing the enabling of a transition, or output inscriptions stating the production rule of tokens when a transition fires. Input/output inscriptions can be functions or variables. Colored Petri Nets (CPN) [38–43] is a graphical language used for modeling and validation of distributed systems including grid computing.

A CPN model of a system is an executable model representing the states of the system and the events (transitions) that can cause the system to change state. CPN is a discrete-event modeling language which combines Petri nets with the functional programming language Standard Markup Language (ML). Petri nets provide the foundation of the graphical notation and Standard ML provides the primitives for the definition of data types, describing data manipulation. It also includes a time concept for representing the time taken to execute events in the modeled system.

CPN Tools is a computer tool to construct and analyze CPN models [38]. Using CPN Tools, it is easier to investigate the behavior of the modeled system using simulation, to verify properties by means of state space methods and model checking, and to conduct simulation-based performance analysis.

4. Reliability of the GTP system

4.1. Nomenclature and decision variables

See Table 1.

Table 1
Definitions.

Decision variables		
r_i	=	$\begin{cases} 1, & \text{if resources are available} \\ 0, & \text{otherwise} \end{cases}$
x_{ik}	=	$\begin{cases} 1, & \text{if transaction } T_i \text{ is scheduled to execute on node } N_k \\ 0, & \text{otherwise} \end{cases}$

Nomenclature

T	Set of transactions
N	Set of nodes in the grid system
T_i	ith transaction $\forall i = 1, \ldots, m$
N_k	kth node $\forall k = 1, \ldots, n$
X	An m by n binary matrix corresponding to a transaction scheduling
γ	Failure rate of node
e_{ik}	Expected execution time of transaction i running on node k
l_{kb}	The communication link from kth to bth node
w_{kb}	The transmission rate of link l_{kb}
σ_{kb}	The failure rate of communication link l_{kb}
ψ_D	The rate of deadline-miss of a transaction
η	Repair rate of node
t	Time
$R_k(X)$	The reliability of system when node k is operational
$R_{kb}(X)$	The reliability of system when link l_{kb} is operational
$R_{k,kb}(X)$	The reliability of system when both node k and link l_{kb} are operational
c	number of servers available at time t
DM	Deadline-miss failure of a transaction
$R_{DM}(X)$	The reliability of system when there is no deadline-miss DM
$R_{k,kb,DM}(X)$	The reliability of system with no deadline-miss DM in addition to node k and link l_{kb} are operational
λ	Arrival rate of transaction
μ	Processing rate of transaction
y_{ik}	The memory required by all the transactions at kth node $\forall i = 1, \ldots, m$
M_k	The available memory size at kth node
L_{ik}	The load required by all the transaction at kth node $\forall i = 1, \ldots, m$
C_{N_k}	The available processing capacity of the kth node
A_λ	Availability of the resources under load λ
$A_{c,\lambda}$	The conditional steady state availability of servers c with load λ
Q_c	The probability that there are exactly c servers are available
K	A certain threshold value after which all r_i's with $i > k$ will be 0
q_i	The steady state probability for an $M/M/c$ model
$R_{k,kb,DM,A_\lambda}(X)$	The reliability with no deadline-miss DM considering the conditional steady state availability A_λ of resources
TM	Transaction management
Miss Ratio	Transaction miss ratio

```
fun newRequest() =
    (Request.ran(), ModelTime.toString(time()), intTime(),
     intTime(), 0, 0)
fun expTime (mean: int) =
  let
    val realMean = Real.fromInt mean
    val rv = exponential((1.0/realMean))
  in
    floor (rv+0.5)
  end;
fun startProc ((((Request,str,tsi,at,wt,pt),p),s,pod_it):sub_Work)
 =
    let
       val proc_time = expTime(avg_proc_time)
       val time_stamp =
          ModelTime.add(time(),ModelTime.fromInt(proc_time))
       val new_str=ModelTime.toString(time_stamp)
       val new_tsi = IntInf.toInt(time_stamp)
       val new_wt = wt+ (intTime() - tsi)
       val new_pt = pt+proc_time
    in
       ((((Request,new_str,new_tsi,at,new_wt,new_pt),p),s,
pod_it),proc_time)
    end
colset E= with e;
colset F_status=
with h_fault|c_fault|nofault;
colset sub_Work_Fstatus=
product sub_Work_status*F_status;
var swfs:sub_Work_Fstatus;
var fs,fs1,fs2,fs3:F_status;
fun error_gen()=discrete(0,1);
fun error_value(n2:int):int=
(if n2>0 then discrete(1,99) else 0);
fun err_no()=discrete(0,2);
fun error_gen1():int=
(error_value(error_gen()));
fun random_error(fs:F_status,n2:int):
F_status=F_status.ran();
```

Fig. 7. Declaration of new request and start process.

4.2. Assumptions

- Transactions arrive according to a Poisson process (i.e., exponentially distributed interarrival times).
- The failure, repair, and transaction processing times are exponentially distributed.
- There is an infinite buffer space for queuing transactions in the system. In modern system, it is likely because, memory is fairly cheap.
- Transaction buffer space is infinite.
- The probability of having i transactions in grid computing system follows a simple $M/M/c$ model. Because, transaction buffer sizes are assumed to be quite large.
- The network topologies in the system are cycle-free. It means that there will be unique path between any pair of edges.
- The failure of a component follows Poisson process (constant failure rate) and are statistically independent.
- Statistically, the failure of components are independent.
- The system considers steady state user-perceived availability for the resources. It is strongly based on the performance (especially the response time) of the system.

4.3. Analytical study on reliability modeling of GTP system

The reliability of transaction in grid computing system is the probability that over a given period of time t, the entire transaction executes properly without failure. The failure of transaction in grid computing system is caused not only by node and link fault but also by deadline-miss fault. Therefore, it is essential to introduce deadline-miss fault while formulating the reliability. Resource availability also plays an important role when transaction is executed in grid computing system within its deadline. In [25], Shatz et al. proposed the reliability formulation for distributed computing system including grid considered the failure caused by node and link fault only. This paper modifies Shatz's formulation [25] by introducing deadline-miss fault [35] and steady

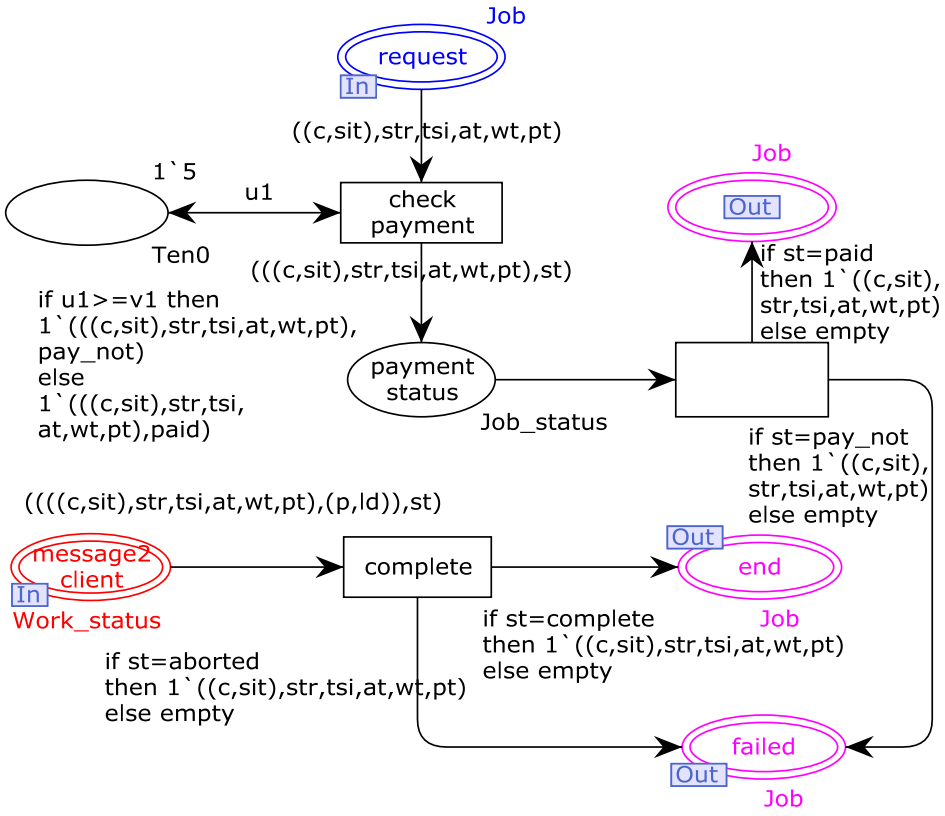

Fig. 8. CPN model of client in Grid.

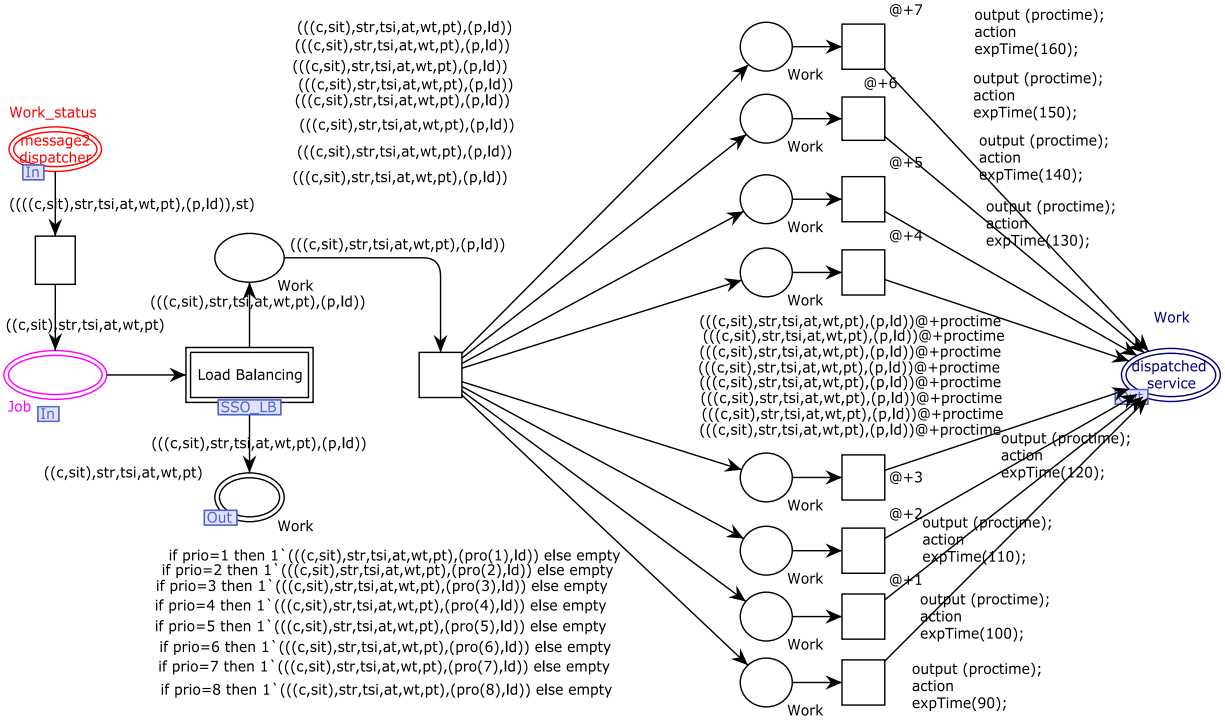

Fig. 9. CPN model of load balancing before scheduling in Grid.

state user-perceived availability [37] of resources. The reliability formulation expressed by [25] is as:

$$R_{k,kb}(X) = \left[\prod_{k=1}^{n} R_k(X)\right].\left[\prod_{k=1}^{n-1}\prod_{b>k} R_{kb}(X)\right] \quad (1)$$

where the node reliability $R_k(X)$ of a node N_k during a time interval t has been computed as

$$e^{-\gamma_k \sum_{i=1}^{m} x_{ik} e_{ik}} \quad (2)$$

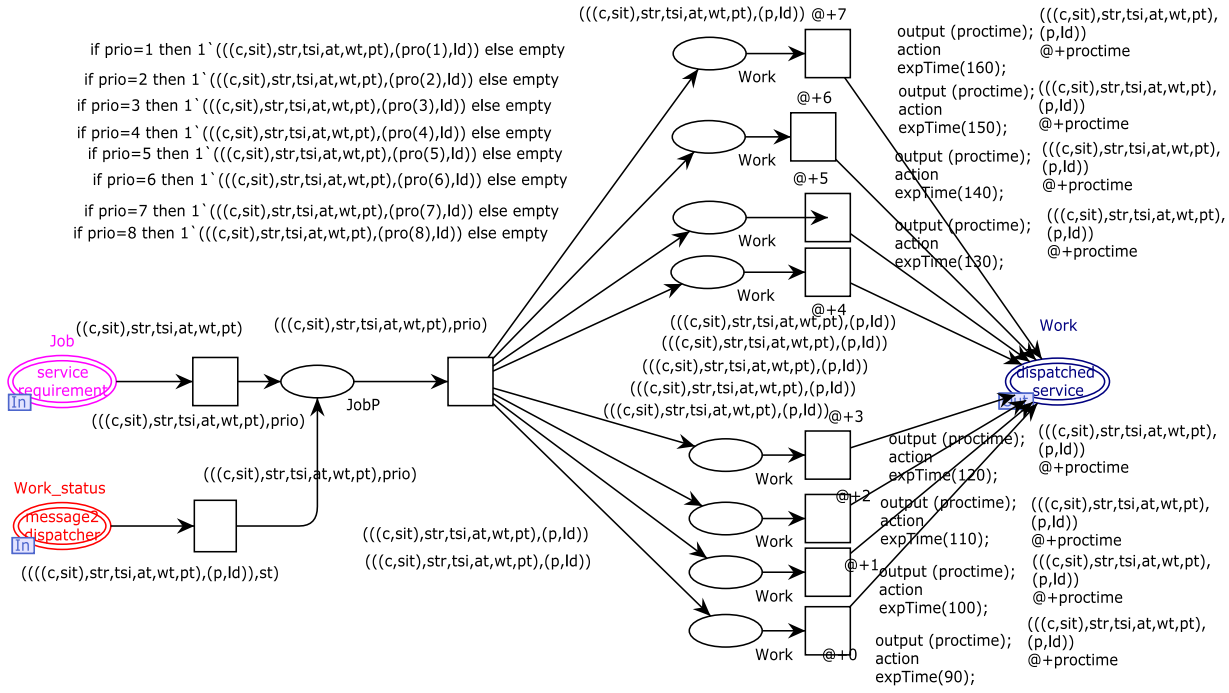

Fig. 10. CPN model of load balancing after scheduling in Grid.

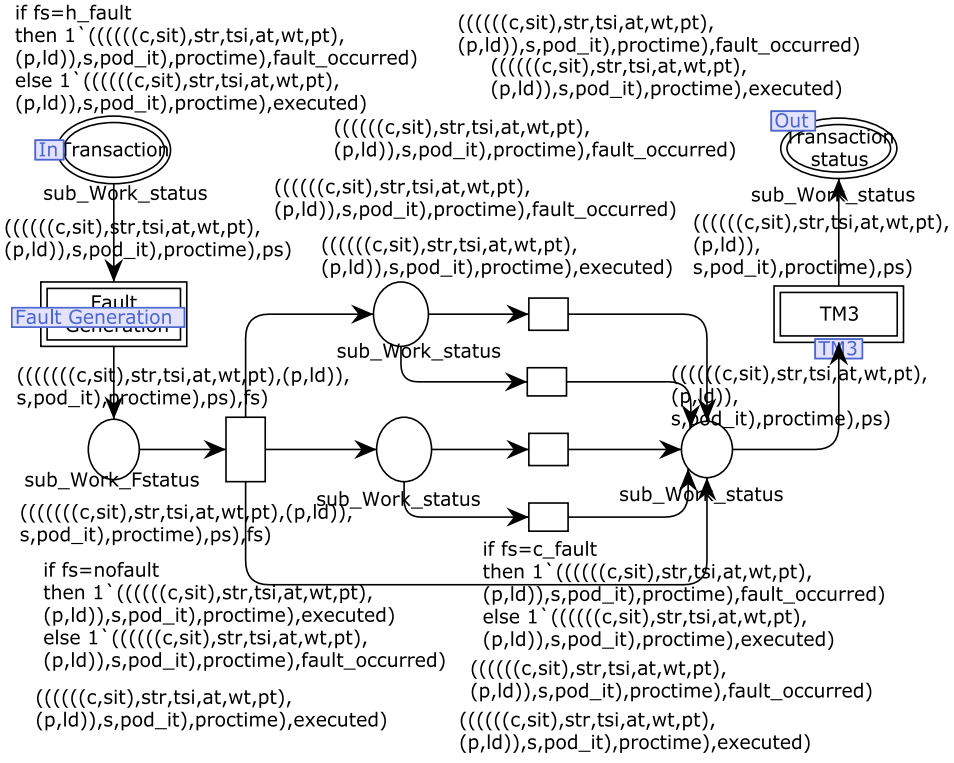

Fig. 11. CPN model of transaction processing in Grid.

where $\sum_{i=1}^{m} x_{ik}e_{ik}$ is the total elapsed time t for executing the transactions assigned to N_k.

The communication link reliability $R_{kb}(X)$ at a time interval t has been computed as

$$e^{-\sigma_{kb}\sum_{i=1}^{m}\sum_{g\neq i}x_{ik}x_{gb}(cost_{ig}/w_{kb})} \tag{3}$$

where the total elapsed time for transmitting the transaction communication via l_{kb} is $\sum_{i=1}^{m}\sum_{g=1}^{m}x_{ik}x_{gb}(cost_{ig}/w_{kb})$.

The system reliability that there is no transaction deadline-miss fault in addition to nodes and communication links are operational during the elapsed time for the execution can be computed as

$$R_{k,kb,DM}(X) = \left[\prod_{k=1}^{n}.R_k(X)\right].\left[\prod_{k=1}^{n-1}\prod_{b>k}.R_{kb}(X)\right].\left[\prod_{i=1}^{m}.R_{DM}(X)\right] \tag{4}$$

where the transactions are steady state and follow queuing system model $M/M/c$ [44] and $R_{DM}(X)$ which is defined as the probability that there is no deadline-miss with rate ψ_D. When transaction T_i is scheduled on N_k can be computed by using the Markov model as

$$R_{DM}(X) = e^{-\psi_D \cdot \left[\frac{1}{\mu} + q_0 \cdot \frac{\rho(c\rho)^c}{c!(c\mu-\lambda)(1-\rho)}\right]}, \quad \forall c \in N \tag{5}$$

where q_0 is given by

$$q_0 = \left[\sum_{k=0}^{c-1} \frac{(c\rho)^k}{k!} + \frac{(c\rho)^c}{c!(1-\rho)}\right]^{-1} \tag{6}$$

where $\rho = \frac{\lambda}{c.\mu} < 1$.

Finally, the reliability of transaction in grid computing system considering conditional steady state user-perceived availability of resources [45] is computed as:

$$R_{k,kb,DM,A_\lambda}(X) = \left[\prod_{k=1}^{n} R_k(X).A_\lambda\right].\left[\prod_{k=1}^{n-1}\prod_{b>k} R_{kb}(X)\right] .\left[\prod_{i=1}^{m} R_{DM}(X)\right] \tag{7}$$

where A_λ is the steady state availability of the resources under the load λ. This paper takes its formula from [37] expressed as:

$$A_\lambda = \sum_{c=1}^{n} A_{c,\lambda} Q_c \tag{8}$$

where $\forall c = 1, \ldots, n$ and $A_{c,\lambda}$ of available servers which has been computed as $\sum_{j=0}^{K} r_j q_k$ with the steady state probabilities for the model which are given as

$$q_k = \frac{(c\rho)^k}{k!} q_0, \quad 1 \le j \le c-1 \tag{9}$$

$$q_k = \frac{c^c \rho^k}{c!} q_0, \quad k \ge c \tag{10}$$

In the above formulation, $\rho = \frac{\lambda}{c\mu}$ and Q_c has been expressed as $Q_c = \frac{n!}{c!(n-c)!} q^c (1-q)^{n-c}$, where $q = \frac{\eta}{\gamma+\eta}$ be the availability of a single server.

4.4. Illustrative example

Suppose there are six virtual nodes which are executing three transactions and contain the resources required by those transactions as shown in Fig. 1.

The procedure for calculating the reliability is proposed as follows:

Step 1. Compute R_{GTP} without considering the deadline-miss failure by solving Eqs. (1)–(3).

Step 2. Solve Eqs. (4)–(6) to obtain R_{DM}.

Step 3. Compute Eq. (7) to obtain the final reliability of the GTP system by considering the deadline-miss failure.

To calculate the reliability of the GTP system, we take the same data sets, network topology image and examples used by [10].

First of all, we calculate R_{DM}. Suppose, $\psi_D = 4 \times 10^{-4}$. Since there are six virtual nodes and three transactions, then $c = 6$ and $i = 3$. Suppose, $\lambda = 1.4\ (s^{-1})$ and $\mu = 0.8\ (s^{-1})$. Then ρ can be calculated as 0.292. Then q_0 can be calculated as 0.240174491. Therefore, q_k is calculated as 0.009647149. The delay by each transaction is computed as 0.006747346. Then R_{DM} is calculated as 0.999498791.

Table 2
Running time (in seconds) of transactions.

Transactions	T_1 on N_1	T_2 on N_4	T_3 on N_2	T_3 on N_3
Time	30	35	15	13

Table 3
Running time (in seconds) of resources.

Time	R_1 on N_5	R_2 on N_1	R_2 on N_4	R_3 on N_4	R_3 on N_5	R_4 on N_4	R_5 on N_2	R_5 on N_6	R_6 on N_3
T_1	5	30	30	20	50				
T_2				15	30	25	20	40	
T_3							25	30	33

Table 4
Reliability of transactions with varying deadline-miss failures.

Deadline-miss failure	R_{DM}	A_λ	R_{GTP}
8×10^{-4}	0.998997834	0.99999	0.871516995
4×10^{-4}	0.999498791	0.99999	0.871954026
3×10^{-4}	0.99962407	0.99999	0.872063318
2×10^{-4}	0.999749364	0.99999	0.872172623
1×10^{-4}	0.999874674	0.99999	0.872281943
0.5×10^{-4}	0.999937335	0.99999	0.872336608
0.25×10^{-4}	0.999968667	0.99999	0.872363941
0.125×10^{-4}	0.999984333	0.99999	0.872377608
0.062×10^{-4}	0.999992229	0.99999	0.872384497
0.031×10^{-4}	0.999996115	0.99999	0.872387887

The reliability of GTP system when it does not consider the deadline-miss failure has been calculated (computed in [10]) as 0.8724. Therefore, the final reliability of the GTP system is calculated by taking the data sets from Tables 2 and 3 as $R_{GTP} = 0.999498791 \times 0.99999 \times 0.8724 = 0.871954026$, where A_λ is supposed to be 0.99999.

4.5. Reliability of the GTP system: Observation

Table 4 depicts the reliability of the GTP with varying deadline-miss failure and resource availability. Table 4 shows that the reliability of the GTP tends to be equal to the grid service reliability 0.8724 calculated in [10] when the deadline-miss failure tends to less than 0.031×10^{-4}. The observation indicates that the deadline-miss failure should also be taken notice of while evaluating the reliability of the GTP system.

The calculated reliability of grid service [10] without deadline-miss failure is 0.8724. The outline for the reliability modeling of GTP can be shown in Table 4.

4.6. The CPN model of the GTP system

This section presents a CPN model for modeling load balanced transaction scheduling in on-demand computing based transaction processing system. In addition, this section also presents the reliability modeling and evaluation for the on-demand computing based transaction processing system.

The model of on-demand based transaction processing is a hierarchical CPNs based model (as shown in Fig. 2). This CPN model consists of several nets where each net describes the behavior of each module of the system. This paper takes the example of grid computing model here. The CPN model consists of various hierarchical nets to model the Grid Transaction Processing system properly. The user requests are generated for the transactional services in the DATAGEN net using exponential distribution. The CLIENT net receives these requests abiding by the rules of the system. Then the requests are scheduled in the SCHEDULER net which decides the order of arrival of the requests and schedules the job. The GLB (Global Load Balancing) net models the global

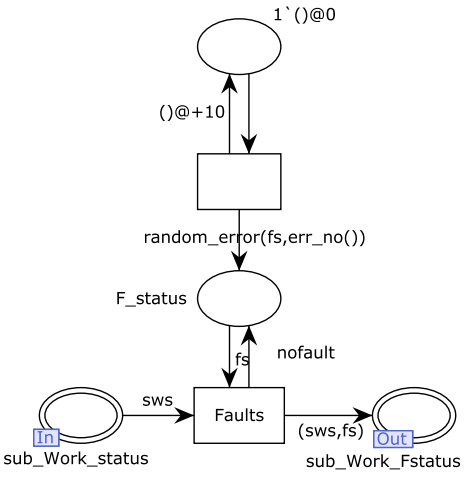

Fig. 12. CPN model of faults generation in Grid.

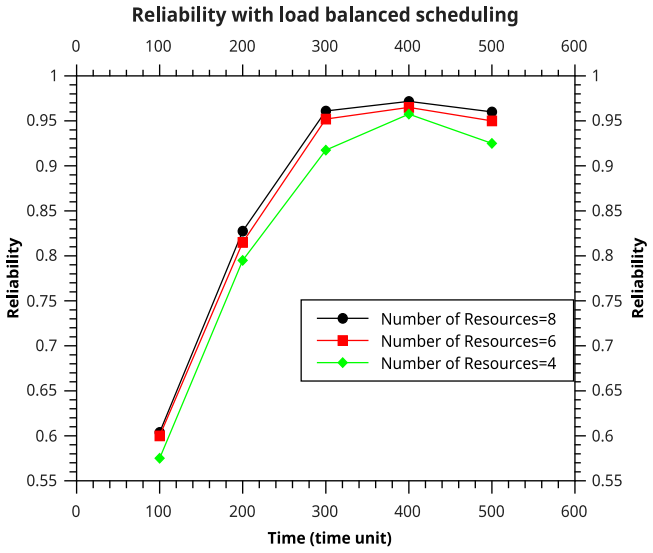

Fig. 13. Reliability of the system with load balancing before scheduling of transaction.

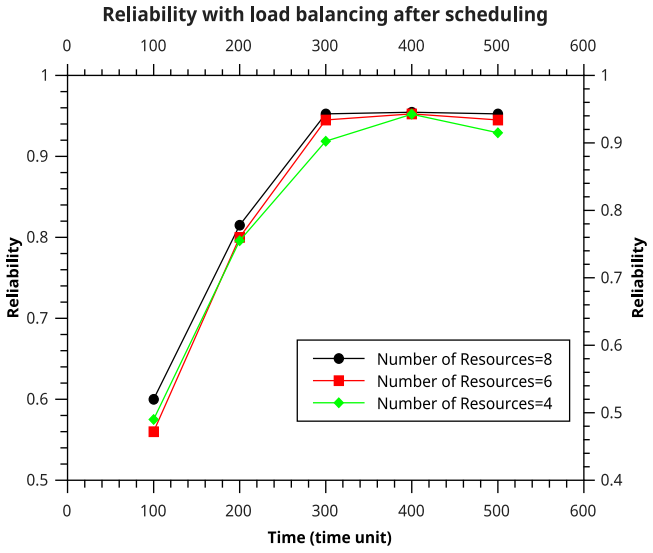

Fig. 14. Reliability of the system with load balancing after scheduling of transaction.

resource management system where the transactional tasks are subdivided into multiple subtasks and those subtasks are sent to the resources for the execution. The LocalRM net describes the local level resource management. After that the transaction management is performed in the TRANSACTION MANAGER net. Then, MONITOR net monitors these tasks. It decides where to send the received tasks. It either sends the tasks to the users or to the SCHEDULER net. If the subtasks are executed unsuccessfully, they are again rescheduled either on the local node (where the subtask has faced failure or on the replicated nodes (any other nodes rather than the local one). In the case of replicated node, the replication method is used. The basic idea of the model is taken from [21].

4.6.1. Transaction generation

The first step in modeling of the GTP system using CPN Tools is to generate the transactional data. In the CPN model of GTP shown in Fig. 2, when we open the DATAGEN net, we find the net which generates the transactional request data randomly using exponential distribution shown in Fig. 3. For data generation,

`newRequest()@ + expTime(100)`

command generates the random data in the model shown in Fig. 3. Each transaction is modeled as

`(((c, sit), str, tsi, at, wt, pt))`

where `c` represents the client set, `sit` represents service item, `str` represents the string type, `at` represents arrival time of the transaction, `wt` represents waiting time of the transaction to be processed, lastly `pt` represents the processing time of the transaction. The detailed information with snapshots of codes used for the simulation are shown in Figs. 4–6 (see Fig. 7).

4.6.2. Client

In Section 3, when we select CLIENT net we find another net which describes the Client net (as shown in Fig. 8). This net is designed to work just like a client (as in client–server system). It checks the payment status of the user and also checks the authentication of the users. This net distinguishes the transaction after the completion of its execution whether it is failed or completed successfully.

`if u1 >= v1 then 1′(((c, sit), str, tsi, at, wt, pt), pay_not)else 1′(((c, sit), str, tsi, at, wt, pt), paid)`

This function used in Fig. 8 is used to allow those transactions for further processing which has declared paid. In the net, 5% of all transactions are assumed as non paid. The non paid transactions are not allowed to go further and these transactions go to `failed` place.

4.6.3. Load balancing before scheduling

The net shown in Fig. 9 models the load balancing to be done before scheduling the transaction. It actually balances the load on the resources before dispatching the transaction to their required nodes or resources to execute.

`if pro = 1 then 1′(((c, sit), str, str, tsi, at, wt, pt), (pro(1), ld)) else empty`

This function sends the transaction to `pro1` i.e., processor 1. `proctime` mentioned in this net represents the process time of the processor.

4.6.4. Load balancing after scheduling

In order to compare the effectiveness of load balancing, we also model load balancing to be done after scheduling the transaction

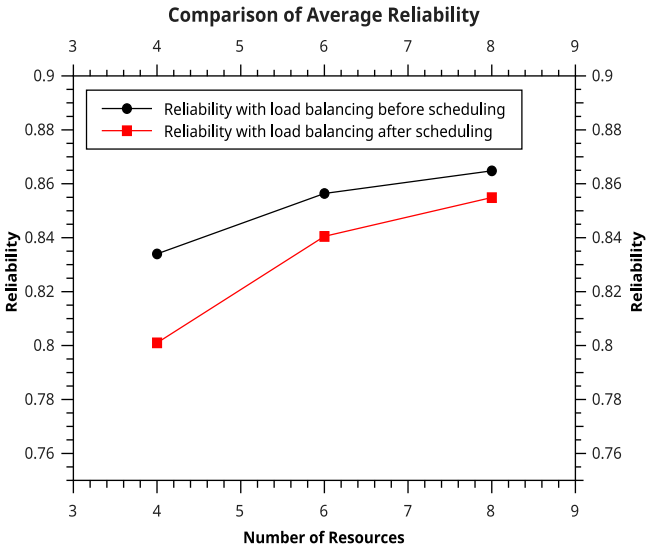

Fig. 15. Comparison of reliability.

as shown in Fig. 10. In this net, load balancing is modeled after the scheduling of the transactions.

4.6.5. Transaction processing

Fig. 11 shows the model of the detailed operation of transaction processing. Here `proctime` denotes the process time of the transaction and `avg_proc_time` denotes the deadline of the transaction. In this net, faults have been modeled. If the execution time is greater than the deadline time of the transaction, this transaction is rolled back.

```
if fs = h_fault then 1'((((((c, sit), str, tsi, at, wt, pt),
                (p, ld)), s, pod_it), proctime), fault_occurred)
                else 1'((((((c, sit), str, tsi, at, wt, pt), (p, ld)),
                                s, pod_it), proctime), executed)
```

This function describes the hardware fault of the system. Similarly, the software fault and the deadline-miss fault have been modeled.

4.6.6. Failure modeling

The fault generation is modeled in Fig. 12 using exponential distribution. All the faults have been generated along with deadline-miss fault.

```
random_error(fs, err_no())
```

This function generates the faults randomly.

4.7. Reliability analysis

Reliability of the GTP system using the CPN Tools by analyzing its CPN models presented in the paper can be calculated as [46]:

$$R_{GTP} = \sum_{i=1}^{m} P_i \tag{11}$$

where P_i is the probability of executing transaction in time t and m is the different realizations of transaction.

5. Results

When we implemented our proposed CPN models on CPN Tools version 4.0 running on Windows 7 with *i*7 processors, we found the following results.

Fig. 13 shows the reliability of the GTP system when it is evaluated using the CPN Tools, when load balancing operation is performed before scheduling of transaction. The simulation is carried out with several experiments. Category 1: when the number of resources is 8, Category 2: when the number of resources is 6, and Category 3: when the number of resources is 4. The result shows that reliability is higher in Category 1 i.e. when the number of resources is 8. The average reliability category wise is shown in Fig. 15. The average reliability are 0.834, 0.8564, and 0.8648 at the respective number of resources 4, 6, and 8.

Fig. 14 depicts the reliability results of the GTP system, when load balancing is performed after scheduling of the transaction. Here also there are three categories; in Category 1, there are 8 number of resources. In Category 2, There are 6 number of resources and in Category 3, they are 4. Here also, we find that the reliability is higher when the number of resources are more in the system. The average reliability category wise is shown in Fig. 15. The average reliability are 0.801, 0.8405, and 0.8549 at the respective number of resources 4, 6, and 8.

6. Conclusions and future aspects

On-demand computing based transaction processing system has gained the popularity rapidly. Load balanced scheduling in this environment is needed due to the dynamic and real-time behavior of the system. The reliability of on-demand computing based transaction processing is defined as the probability of all the transactions involved in the given service being executed, successfully. Due to the dynamic and real-time behavior of this system, high-level extensions of Petri nets are required for the modeling and analysis of this system. Colored Petri nets (CPNs), one of the extensions of Petri nets can be used to model on-demand computing based transaction processing system.

The paper models the load balanced scheduling and reliability of the GTP system based on CPNs. For evaluation of the CPN models, the paper uses CPN Tools. For results verification, first analytical reliability model of the GTP system was implemented . Then reliability of the system using CPN Tools was calculated. It was found that reliability are approximately same which verified the accuracy of the model. Also the comparison between the reliability of the system when simulated with load balancing before scheduling and that of the system when simulated with load balancing after scheduling was performed.

The future works for further research in this area consist of the following topics :

- The use of the CPN to model and analyze the dependability of on-demand computing based transaction processing system.
- Proposing a general model for workflow scheduling in on-demand computing based transaction processing system.
- Proposing a model for virtual machine placement in cloud computing based transaction processing system.
- Using the other extensions of the Petri nets, such as generalized stochastic Petri nets (GSPNs) to evaluate other performance and dependability measures with load balanced scheduling in on-demand computing based transaction processing system.

References

[1] Mahato DP, Singh RS. Reliability modeling and analysis for deadline-constrained grid service. In: 2018 32nd international conference on advanced information networking and applications workshops (WAINA). IEEE; 2018.

[2] Mahato DP, Maurya AK, Tripathi AK, Singh RS. Dynamic and adaptive load balancing in transaction oriented grid service. In: 2016 2nd international conference on green high performance computing (ICGHPC). IEEE; 2016, p. 1–5.

[3] Mahato DP, Singh RS. Balanced task allocation in the on-demand computing-based transaction processing system using social spider optimization. Concurr Comput: Pract Exper 2017;29(18). e4214.
[4] Mahato DP, Singh RS. On maximizing reliability of grid transaction processing system considering balanced task allocation using social spider optimization. Swarm Evol Comput 2018;38:202–17.
[5] Mahato DP, Umrao LS, Singh RS. Recovery of failures in transaction oriented composite grid service. In: IJCA proceedings on computing communication and sensor network 2013, no. 2013. 2013, p. 38–42.
[6] Mahato DP, Singh RS, Tripathi AK, Maurya AK. On scheduling transactions in a grid processing system considering load through ant colony optimization. Appl Soft Comput 2017;61:875–91.
[7] Mahato DP, Singh RS. Load balanced transaction scheduling using Honey bee optimization considering performability in on-demand computing system. Concurr Comput: Pract Exper 2017;29(21). e4253.
[8] Mahato DP. Cuckoo search-ant colony optimization based scheduling in grid computing. In: Proceedings of the 47th international conference on parallel processing companion. ACM; 2018, p. 39.
[9] Mahato DP. Cpns based reliability modeling for on-demand computing based transaction processing. In: Proceedings of the 47th international conference on parallel processing companion. ACM; 2018, p. 24.
[10] Dai Y-S, Pan Y, Zou X. A hierarchical modeling and analysis for grid service reliability. IEEE Trans Comput 2007;56(5):681–91.
[11] Trivedi KS, Muppala JK, Woolet SP, Haverkort BR. Composite performance and dependability analysis. Perform Eval 1992;14(3):197–215.
[12] Raza Z, Vidyarthi DP. Maximizing reliability with task scheduling in a computational grid using ga. Int J Adv Comput Technol 2009;1(2):40–7.
[13] Hsieh C-C, Hsieh Y-C. Reliability and cost optimization in distributed computing systems. Comput Oper Res 2003;30(8):1103–19. http://dx.doi.org/10.1016/S0305-0548(02)00058-8, URL http://www.sciencedirect.com/science/article/pii/S0305054802000588.
[14] Shatz SM, Wang JP. Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems. IEEE Trans Reliab 1989;38(1):16–27. http://dx.doi.org/10.1109/24.24570.
[15] Avizienis A, Laprie J-C, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans Dependable Secure Comput 2004;1(1):11–33.
[16] Murata T. Petri nets: Properties, analysis and applications. Proc IEEE 1989;77(4):541–80.
[17] Wang T, Vonk J, Kratz B, Grefen P. A survey on the history of transaction management: from flat to grid transactions. Distrib Parallel Databases 2008;23(3):235–70. http://dx.doi.org/10.1007/s10619-008-7028-1.
[18] Tang F, Li M, Huang JZ. Real-time transaction processing for autonomic grid applications. Eng Appl Artif Intell 2004;17(7):799–807. http://dx.doi.org/10.1016/j.engappai.2004.09.002, autonomic Computing Systems. URL http://www.sciencedirect.com/science/article/pii/S0952197604001228.
[19] Tang F-L, Li M-L, Huang Z-X, Wang C-L. Transaction service for service grid and its correctness analysis based on petri net. Jisuanji Xuebao/Chin J Comput 2005;28(4):667–76.
[20] Tang F, Guo M, Li M, Li L. Transaction management for reliable grid applications. In: International conference on advanced information networking and applications, 2009. AINA '09. 2009, p. 427–34. http://dx.doi.org/10.1109/AINA.2009.11.
[21] Mahato DP, Umrao LS, Singh RS. Adaptability in transaction oriented grid service. In: 2014 International conference on parallel, distributed and grid computing (PDGC). 2014, p. 239–44. http://dx.doi.org/10.1109/PDGC.2014.7030749.
[22] Türker C, Haller K, Schuler C, Schek H-J. How can we support grid transactions? towards peer-to-peer transaction processing. In: CIDR. Citeseer; 2005, p. 174–85.
[23] Tang X, Li K, Li R, Veeravalli B. Reliability-aware scheduling strategy for heterogeneous distributed computing systems. J Parallel Distrib Comput 2010;70(9):941–52. http://dx.doi.org/10.1016/j.jpdc.2010.05.002, URL http://www.sciencedirect.com/science/article/pii/S074373151000095X.
[24] Yen I, Chen I-R, et al. Reliability assessment of multiple-agent cooperating systems. IEEE Trans Reliab 1997;46(3):323–32.
[25] Shatz SM, Wang JP, Goto M. Task allocation for maximizing reliability of distributed computer systems. IEEE Trans Comput 1992;41(9):1156–68. http://dx.doi.org/10.1109/12.165396.
[26] Kartik S, Murthy CSR. Improved task-allocation algorithms to maximize reliability of redundant distributed computing systems. IEEE Trans Reliab 1995;44(4):575–86. http://dx.doi.org/10.1109/24.475976.
[27] Attiya G, Hamam Y. Task allocation for maximizing reliability of distributed systems: A simulated annealing approach. J Parallel Distrib Comput 2006;66(10):1259–66. http://dx.doi.org/10.1016/j.jpdc.2006.06.006, URL http://www.sciencedirect.com/science/article/pii/S0743731506001432.
[28] Vidyarthi DP, Tripathi AK. Maximizing reliability of distributed computing system with task allocation using simple genetic algorithm. J Syst Archit 2001;47(6):549–54. http://dx.doi.org/10.1016/S1383-7621(01)00013-3, URL http://www.sciencedirect.com/science/article/pii/S1383762101000133.
[29] Kang Q-M, He H, Song H-M, Deng R. Task allocation for maximizing reliability of distributed computing systems using honeybee mating optimization. J Syst Softw 2010;83(11):2165–74. http://dx.doi.org/10.1016/j.jss.2010.06.024, interplay between Usability Evaluation and Software Development. URL http://www.sciencedirect.com/science/article/pii/S0164121210001718.
[30] Yin P-Y, Yu S-S, Wang P-P, Wang Y-T. Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization. J Syst Softw 2007;80(5):724–35. http://dx.doi.org/10.1016/j.jss.2006.08.005, component-Based Software Engineering of Trustworthy Embedded Systems. URL http://www.sciencedirect.com/science/article/pii/S0164121206002214.
[31] Pezoa JE, Dhakal S, Hayat MM. Maximizing service reliability in distributed computing systems with random node failures: Theory and implementation. IEEE Trans Parallel Distrib Syst 2010;21(10):1531–44.
[32] Chen C-Y. Task scheduling for maximizing performance and reliability considering fault recovery in heterogeneous distributed systems. IEEE Trans Parallel Distrib Syst 2016;27(2):521–32.
[33] Haque W, Toms A, Germuth A. Dynamic load balancing in real-time distributed transaction processing. In: 2013 IEEE 16th international conference on computational science and engineering (CSE). IEEE; 2013, p. 268–74.
[34] Chu WW, Holloway LJ, Efe K, et al. Task allocation in distributed data processing. Computer 1980;(11):57–69.
[35] Xiao P, Hu Z. Workload-aware reliability evaluation model in grid computing. J Comput 2012;7(1):141–6.
[36] Chang R-S, Chang J-S, Lin P-S. An ant algorithm for balanced job scheduling in grids. Future Gener Comput Syst 2009;25(1):20–7. http://dx.doi.org/10.1016/j.future.2008.06.004, URL http://www.sciencedirect.com/science/article/pii/S0167739X08000848.
[37] Mainkar V. Availability analysis of transaction processing systems based on user-perceived performance. In: The sixteenth symposium on reliable distributed systems, 1997. Proceedings. IEEE; 1997, p. 10–7.
[38] Jensen K. Coloured petri nets. In: Petri nets: central models and their properties. Springer; 1987, p. 248–99.
[39] Jensen K, Kristensen LM, Wells L. Coloured petri nets and cpn tools for modelling and validation of concurrent systems. Int J Softw Tools Technol Trans 2007;9(3–4):213–54.
[40] Jensen K, Kristensen LM. Coloured Petri nets: modelling and validation of concurrent systems. Springer Science & Business Media; 2009.
[41] Jensen K. An introduction to the theoretical aspects of coloured petri nets. In: Workshop/school/symposium of the REX project (Research and education in concurrent systems). Springer; 1993, p. 230–72.
[42] Jensen K. An introduction to the practical use of coloured petri nets. In: Lectures on petri nets II: Applications. Springer; 1998, p. 237–92.
[43] Bratosin C, Van Der Aalst W, Sidorova N. Modeling grid workflows with colored petri nets. In: Eighth workshop and tutorial on practical use of coloured petri nets and the CPN tools. 2007, p. 67.
[44] Bertsekas DP, Gallager RG, Humblet P. Data networks, Vol. 2. Prentice-Hall International New Jersey; 1992.
[45] Teorey TJ, Ng WT. Dependability and performance measures for the database practitioner. IEEE Trans Knowl Data Eng 1998;10(3):499–503.
[46] Azgomi MA, Entezari-Maleki R. Task scheduling modelling and reliability evaluation of grid services using coloured petri nets. Future Gener Comput Syst 2010;26(8):1141–50. http://dx.doi.org/10.1016/j.future.2010.05.015, URL http://www.sciencedirect.com/science/article/pii/S0167739X10001093.

Dharmendra Prasad Mahato is Assistant Professor in the Department of Computer Science and Engineering, BIT Sindri, Dhanbad. He obtained his Ph.D. from Indian Institute of Technology (Banaras Hindu University) Varanasi, India. He received his A.M.I.E.T.E. Degree in Computer Science & Engineering and M.Tech. in Computer Science & Engineering from ABV-IIITM, Gwalior. His research interests include the Distributed Computing, Grid Computing, Cloud Computing.

Ravi Shankar Singh is Associate Professor in the Department of Computer Science and Engineering, Indian Institute of Technology (Banaras Hindu University), Varanasi. He obtained his Ph.D. from Indian Institute of Technology (Banaras Hindu University) Varanasi, India. His research interests include the Parallel and Distributed Computing, High Performance Computing, and Algorithms.