

# Availability Prediction and Modeling of High Availability OSCAR Cluster

Chokchai Leangsuksun<sup>1</sup>, Lixin Shen  
Tong Liu, Hertong Song  
Computer Science  
Louisiana Tech University  
Ruston, LA 71272, USA  
{box, lsh007, tli001, hso001}@latech.edu

Stephen L. Scott<sup>2</sup>  
Computer Science and Mathematics Division  
Oak Ridge National Laboratory  
Oak Ridge, TN 37831, USA  
scottsl@ornl.gov

## Abstract

*Since the initial introduction of Open Source Cluster Application Resources (OSCAR), this software package has been a well-accepted choice for building high performance computing systems. As it continues to be applied to mission-critical environments, high availability (HA) features therefore are needed to be included in OSCAR cluster. In this paper, we provide a HA solution for OSCAR cluster. As a widely used technique in HA solutions, component redundancy is adopted to improve the system availability. Based on the proposed architecture, we develop a detailed failure-repair model for predicting the availability of HA OSCAR cluster. Stochastic reward nets (SRN) are used to model the behavior of the system. We specify our SRN model to Stochastic Petri Net Package, describe and compute several interesting output measures that characterize availability features of HA OSCAR cluster.*

## 1. Introduction

As an enormous computational power for the scientific, commercial, and educational communities, clusters are becoming increasingly cost effective and popular [1] [2]. Open Source Cluster Application Resources (OSCAR) is a fully integrated bundle of software designed for building, maintaining, and using a Linux cluster for high performance computing (HPC) [3] [4] [5]. Since the initial introduction of OSCAR, this software package has become a well-accepted choice for

building HPC clusters. As cluster computing systems continue to be applied to mission-critical environments, high availability (HA) features are therefore required to be included in OSCAR cluster.

The current OSCAR cluster suffers from several single point of failure. The failure rates of some key components dominate the availability of the entire system. In order to improve the system availability, component redundancy is adopted in HA OSCAR cluster to eliminate the single point of failure. First, a standby server is introduced as a duplication to take over the control of the cluster from the primary server when it is failed. A communication feature with heartbeat mechanism is applied as health detection between the two servers. Second, redundant network connection provides an alternative communication path in the case of network card, wire or switch failure happens. Last, a quorum voting mechanism is used for the multiple clients. Component redundancy will significantly improve the system availability [6] [7].

Availability analysis and modeling of the HA OSCAR cluster can predict the availability measures in the design stage and help us to determine the range of parameters so that the system availability goal will be met. Stochastic reward nets (SRN) have been successfully used in the availability evaluation and prediction for complicated systems, especially when the time-dependent behavior is of great interest [8]. We apply SRN for aiding in the automated generation and solution of the underlying large and complex Markov chains. The Stochastic Petri Net Package (SPNP) [9] allows the specification of SRN models and the computation of steady and transient-state. We utilize SPNP to build and solve our SRN model.

In this paper, we provide a HA solution for OSCAR cluster. Based on the system architecture, we use SRN to develop a detailed failure-repair behavior model for the system. Several interesting output measures that characterize the availability features of the system are described and computed. The use of the model in studying various alternative configurations of the HA OSCAR

---

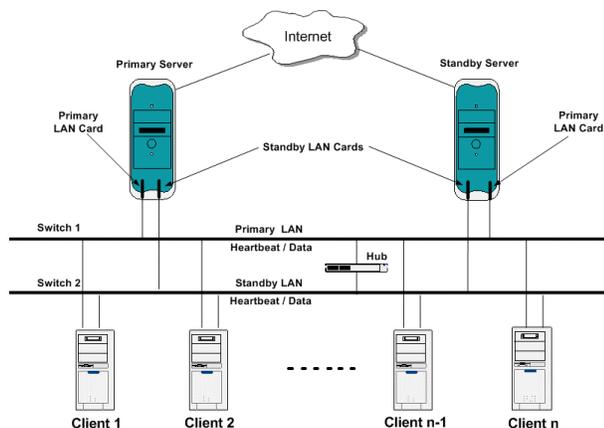
<sup>1</sup>Research supported by Center for Entrepreneurship and Information Technology, Louisiana Tech University.

<sup>2</sup>Research supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, Office of Science, U. S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

cluster is also illustrated. The architecture and mechanism of the HA OSCAR cluster will be described in section 2. In section 3, we will develop the SRN model corresponding to the system. The behavior of servers, network connections, and clients will be discussed in detail. In section 4, we will provide some numerical results obtained from the model. Finally, conclusions will be presented in section 5.

## 2. System architecture and mechanism

The hardware configuration of HA OSCAR cluster is shown in Figure 1. The system consists of a primary server, a standby server, two LAN connections, and multiple clients, where all the clients have homogeneous hardware. A server is responsible for serving user's requests and distributing the requests to specified clients. A client is dedicated to computation [10]. Each server has three network interface cards, one is connected to the Internet by a public network address, and the other two are connected to a private LAN, which consists of a primary Ethernet LAN and a standby LAN. Each LAN consists of network interface cards, switch, and network wires, and provides communication between servers and clients, and also between the primary server and the standby server.



**Figure 1. Architecture of the HA OSCAR cluster**

Initially, every component in the cluster is functioning. The primary server provides the services and processes all the user's requests. The standby server activates its services and waits for taking over the primary server when its failure is detected. The periodical transmission of heartbeat messages travels across the Ethernet LAN between the two servers, and works as health detection of the primary server [11]. When a primary server failure occurs, the heartbeat detection on the standby server cannot receive any response message from the primary

server. After a prescribed time, the standby server takes over the alias IP address of the primary server, and the control of the cluster transfers from the primary server to the standby server. The user's requests are processed on the standby server from later on. From the user's point of view, the transfer is almost seamless except the short prescribed time. The failed primary server gets repaired after the standby server taking over the control. Once the repair is completed, the primary server activates the services, takes over the alias IP address, and begins to process user's requests. The standby server releases its alias IP address and goes back to initial state.

At regular interval, the running server polls all the LAN components specified in the cluster configuration file, including the primary LAN cards, the standby LAN cards, and the switches. Network connection failures are detected in the following manner. The standby LAN interface is assigned to be the poller. The polling interface sends packet messages to all other interfaces on the LAN and receives packets back from all other interfaces on the LAN. If an interface cannot receive or send a message, the numerical count of packets sent and received on an interface does not increment for an amount of time. At this time, the interface is considered down [12]. When the primary LAN goes down, the standby LAN takes over the network connection. When the primary LAN gets repaired, it takes over the connection back from the standby LAN.

Whenever a client fails or gets repaired and the cluster is functioning, then the cluster undergoes a reconfiguration to remove the corresponding client from or admit the client into the cluster. This process is referred to as a cluster state transition. The HA OSCAR cluster uses a quorum voting scheme to keep the system performance requirement, where the quorum  $Q$  is the minimum number of functioning clients required for a HPC system. We consider a system with  $N$  clients, and assume that each client is assigned one vote. The minimum number of votes required for a quorum is given by  $(N+2)/2$  [13]. Whenever the total number of the votes contributed by all the functioning clients falls below the quorum value, the system suspends operation. Upon the availability of sufficient number of clients to satisfy the quorum, a cluster resumption process takes place, and brings the system back to operational state.

## 3. System model

We divide the system behavior model into server submodel, network connection submodel and client submodel. A brief description about each place and transition is presented in a tabular form for each of these submodels.

### 3.1 Server submodel

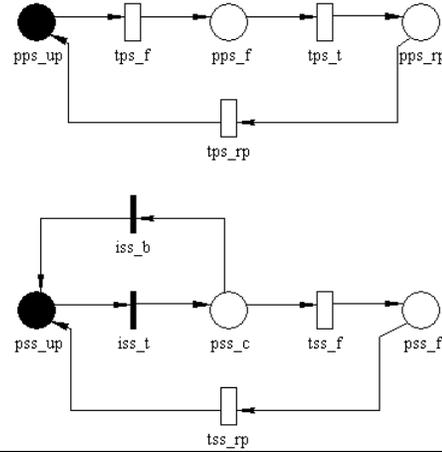
**Table 1. Places and transitions of the server behavior**

Place	Represents
pps_up	primary server functioning, controlling the cluster
pps_f	primary server failure occurred
pps_rp	failed primary server waiting for repair
pss_up	standby server functioning, waiting for taking over control
pss_c	standby server controlling the cluster
pss_f	standby server failure occurred, waiting for repair

Transition	Represents
tps_f	failure of primary server
tps_t	failed primary server being taken over
tps_rp	repair of primary server
iss_t	standby server taking over control
tss_f	failure of standby server
tss_rp	repair of standby server
iss_b	standby server being taken over

The failure-repair behavior of servers is modeled as shown in Figure 2. The places and transitions are shown in Table 1. Both the primary server and the standby server are either up or down. Initially, a token presents in places  $pps\_up$  and  $pss\_up$ , respectively, representing that both the primary server and the standby server are functioning. The primary server controls the cluster, and the standby server waits for taking over the control of the cluster. When a primary server failure occurs, a token is moved from place  $pps\_up$  to  $pps\_f$ , which is modeled by timed transition  $tps\_f$ . The time to primary server failure is exponentially distributed with rate  $\lambda_{ps}$ . The standby server detects the health of the primary server from time to time. It notices the failure of primary server at the end of a prescribed time interval. This is modeled by the timed transition  $tps\_t$ , and the time to failure detection is exponentially distributed with rate  $\mu_{st}$ . So, a token is moved from place  $pps\_f$  to  $pps\_rp$ . Once the token presents in the place  $pps\_rp$ , it triggers the immediate transition  $iss\_t$ , which means that the standby server takes over the control of the cluster from the primary server. The repair of the primary server is modeled by timed transition  $tps\_rp$ , and the primary server repair time is exponentially distributed with rate  $\mu_{psr}$ . When the primary server gets repaired, a token moves from place  $pps\_rp$  to  $pps\_up$ . Once the token presents in the place  $pps\_up$ , it triggers the immediate transition  $iss\_b$ , which means that the primary server takes over the control of cluster back from the standby server. The standby server goes back to the original state, in which it keeps detecting the health of

the primary server and waiting for taking over the control. The failure of the standby server is modeled by the timed transition  $tss\_f$ . The time to standby server failure is exponentially distributed with rate  $\lambda_{ss}$ . The repair of standby server is modeled by timed transition  $tss\_rp$ , and the time to standby server repair is exponentially distributed with rate  $\mu_{ssr}$ .



Transition	Priority
tps_f, tps_t, tps_rp, iss_t, tss_f, tss_rp, iss_b	500

Transition	Enabling Function
iss_t	$(\#(pps\_rp)=1)$
iss_b	$(\#(pps\_up)=1)$

Transition	Rate Function
tps_f	$\#(pps\_up)\lambda_{ps}$
tps_t	$\#(pps\_f)\mu_{st}$
tps_rp	$\#(pps\_rp)\mu_{psr}$
tss_f	$\#(pss\_c)\lambda_{ss}$
tss_rp	$\#(pss\_f)\mu_{ssr}$

**Figure 2. SRN model of the server behavior**

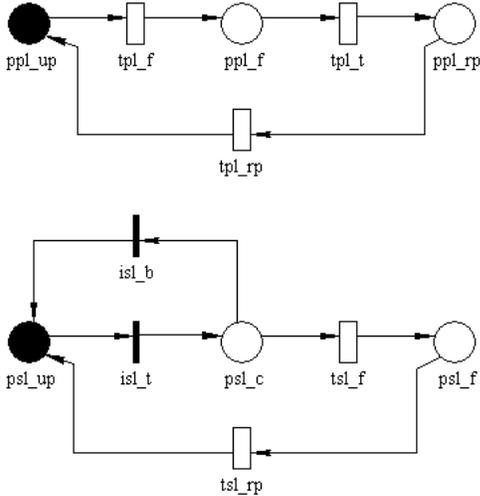
### 3.2 Network connection submodel

**Table 2. Places and transitions of the network connection behavior**

Place	Represents
ppl_up	primary LAN functioning, connecting the network
ppl_f	primary LAN failure occurred
ppl_rp	failed primary LAN waiting for repair
psl_up	standby LAN functioning, waiting for taking over network connection
psl_c	standby LAN connecting the network
psl_f	standby LAN failure occurred, waiting for repair

Transition	Represents
tpl_f	failure of primary LAN
tpl_t	failed primary LAN being taken over
tpl_rp	repair of primary LAN
isl_t	standby LAN taking over connection
tsl_f	failure of standby LAN
tsl_rp	repair of standby LAN
isl_b	standby LAN being taken over

The failure-repair behavior of LAN is modeled as shown in Figure 3. The places and transitions are shown in Table 2. The behavior of the primary LAN and the standby LAN is similar to that of the primary server and standby server in server submodel. We also assume that the timed transitions are exponentially distributed. But, the priority of the transitions is lower than the priority of transitions in the server submodel. The detailed behavior discuss may refer to the server submodel.



Transition	Priority
tpl_f, tpl_t, tpl_rp, isl_t, tsl_f, tsl_rp, isl_b	400

Transition	Rate Function
tpl_f	$\#(ppl\_up)\lambda_{pl}$
tpl_t	$\#(ppl\_f)\mu_t$
tpl_rp	$\#(ppl\_rp)\mu_{plr}$
tsl_f	$\#(psl\_c)\lambda_{sl}$
tsl_rp	$\#(ppl\_f)\mu_{slr}$

Transition	Enabling Function
isl_t	$(\#(ppl\_rp)=1)$
isl_b	$(\#(ppl\_up)=1)$

Figure 3. SRN model of the LAN behavior

### 3.3 Client submodel

Table 3. Places and transitions of the client behavior

Place	Represents
pc_up	client functioning
pc_pf	client permanent failure occurred
pc_pwo	client with permanent failure waiting for reconfiguring out of cluster
pc_pwr	client with permanent failure waiting for reboot cluster
pc_rp	failed client waiting for repair
pc_wi	repaired client waiting for reconfiguring into cluster
pc_if	client intermittent failure occurred
pc_iwr	client with intermittent failure waiting for reboot cluster
pc_iwo	client with intermittent failure waiting for reconfiguring out of cluster
pc_rb	failed client waiting for reboot

Transition	Represents
tc_pf	permanent failure of client
ic_cpf	covered permanent failure
ic_upf	uncovered permanent failure
tc_pro	client with covered permanent failure reconfiguring out of cluster
tc_prb	cluster with uncovered permanent failure rebooting
tc_rp	repair of client
tc_ri	client reconfiguring into cluster
ic_in	quorum not present, immediate transiting into cluster
tc_if	intermittent failure of client
ic_cif	covered intermittent failure
ic_uif	uncovered intermittent failure
tc_irb	cluster with uncovered intermittent failure rebooting
tc_iro	client with covered intermittent failure reconfiguring out of cluster
tc_crb	client rebooting
tc_pfr	permanent failure when client being rebooted

The failure-repair behavior of a sub system with  $N$  clients is modeled as shown in Figure 4. Its places and transitions are shown in Table 3 [14] [15]. The  $N$  clients are either up or down. Initially, the  $N$  tokens are present in place  $pc\_up$  representing that every client in the sub system is functioning. The clients can suffer two kinds of failures: permanent failure and intermittent failure. A permanent failure is any failure that requires the physical repair of the failed component, which is modeled by timed transition  $tc\_pf$ , and an intermittent failure is one that can be corrected by rebooting the corresponding



## 4. Example and numerical results

In this section, we use the SRN model described above to study the availability of HA OSCAR cluster through an example. The input parameter values used in the computation of the measures are shown in Table 4. These values are for illustrative only.

**Table 4. Input parameter for the HA OSCAR cluster**

Input Parameter	Numerical Value
Mean time to primary server failure, $1/\lambda_{ps}$	5,000 hrs.
Mean time to primary server repair, $1/\mu_{psr}$	4 hrs.
Mean time to takeover primary server, $1/\mu_{st}$	30 sec.
Mean time to standby server failure, $1/\lambda_{ss}$	5,000 hrs.
Mean time to standby server repair, $1/\mu_{ssr}$	4 hrs.
Mean time to primary LAN failure, $1/\lambda_{pl}$	10,000 hrs.
Mean time to primary LAN repair, $1/\mu_{plr}$	1 hr.
Mean time to takeover primary LAN, $1/\mu_{lt}$	30 sec.
Mean time to standby LAN failure, $1/\lambda_{sl}$	10,000 hrs.
Mean time to standby LAN repair, $1/\mu_{slr}$	1 hr.
Mean time to client permanent failure, $1/\lambda_p$	2,000 hrs.
Mean time to client intermittent failure, $1/\lambda_i$	1,000 hrs.
Mean time to system reboot, $1/\mu_{srb}$	15 min.
Mean time to client reboot, $1/\mu_{crb}$	5 min.
Mean time to client reconfiguration, $1/\mu_{rc}$	1 min.
Mean time to client repair, $1/\mu_{rp}$	4 hrs.
Permanent failure coverage factor, $c_p$	0.95
Intermittent failure coverage factor, $c_i$	0.95

We assume that the system is functioning only if either primary server or the standby server is functioning, either primary LAN or the standby LAN is functioning, the quorum for clients is present, and no system rebooting or reconfiguration is being processed. The availability of the system at time  $t$  is computed as the expected instantaneous reward rate  $E[X(t)]$  at time  $t$  and its general expression is

$$E[X(t)] = \sum_{k \in \tau} r_k \pi_k(t)$$

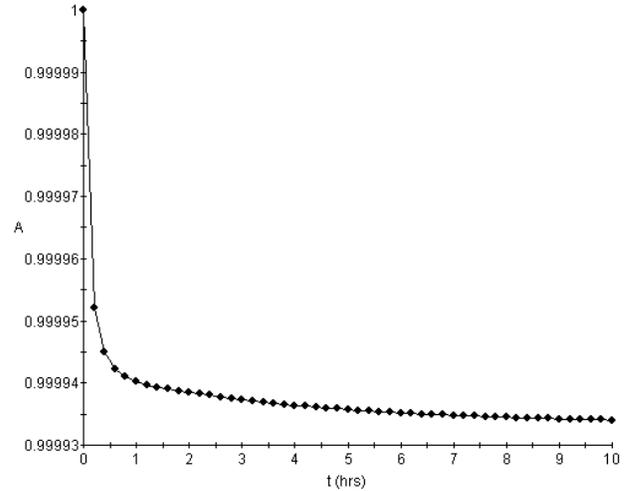
where  $r_k$  represents the reward rate assigned to state  $k$  of the SRN,  $\tau$  is the set of tangible marking, and  $\pi_k(t)$  is the probability of being in marking  $k$  at time  $t$  [16] [17].

The steady-state system availability and the mean cluster down time per year for the different configurations are shown in Table 5. We notice that the system availabilities for the various configurations are almost the same. After we introduce the quorum voting mechanism in the client submodel, the system availability is not sensitive to the change of client configuration. This implies that when we add more clients to improve the system performance, the availability of the system can almost remain unchanged. The instantaneous availabilities

of the system are shown in Figure 5 when it has 8 clients and the quorum is 5.

**Table 5. System availability for different configurations**

System Config. (N)	Quorum Value (Q)	System Availability (A)	Mean cluster down time (min./yr.) (t)
4	3	0.999933475091	34.9654921704
6	4	0.999933335485	35.0388690840
8	5	0.999933335205	35.0390162520
16	9	0.999933335204	35.0390167776



**Figure 5. System instantaneous availabilities**

## 5. Conclusion

Under the requirement of mission critical application, we introduce a HA solution for OSCAR cluster. We build a SRN model for the detailed failure-repair behavior of the system, and calculate and predict the system availability of the system. The results show that the component redundancy is efficient in improving system availability.

Several interesting HA features to be included in HA OSCAR cluster will be considered in the future. These include storage disk redundancy, application level failover, etc. Sensitive analysis and tradeoff analysis are also interesting areas left to be explored.

## 6. References

- [1] I. Ahmad, "Cluster Computing: A Glance at Recent Events", *IEEE Concurrency*, January-March 2000, vol. 8, no. 1, pp. 67-69.

- [2] M.J. Brim, T.G. Mattson, and S.L. Scott, "OSCAR: Open Source Cluster Application Resources", *Ottawa Linux Symposium 2001*, Ottawa, Canada, 2001.
- [3] J. Hsieh, T. Leng, and Y.C. Fang, "OSCAR: A Turnkey Solution for Cluster Computing", *Dell Power Solutions*, 2001, Issue 1, pp. 138-140.
- [4] <http://oscar.sourceforge.net/>
- [5] The Open Cluster Group, "OSCAR Cluster User's Guide, Software Version 2.2, Documentation Version 2.2", February 27, 2003.
- [6] C. Leangsuksun, L. Shen, H. Song, S.L. Scott, and I. Haddad, "The Modeling and Dependability Analysis of High Availability OSCAR Cluster", *The 17th Annual International Symposium on High Performance Computing Systems and Applications*, Quebec, Canada, May 11-14, 2003.
- [7] C. Leangsuksun, L. Shen, T. Liu, H. Song, and S.L. Scott, "Dependability Prediction of High Availability OSCAR Cluster Server", *The 2003 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'03)*, Las Vegas, Nevada, USA, June 23-26, 2003.
- [8] J. Muppala, G. Ciardo, and K.S. Trivedi, "Stochastic Reward Nets for Reliability Prediction", *Communications in Reliability, Maintainability and Serviceability: An International Journal published by SAE International*, July 1994, Vol. 1, No. 2, pp. 9-20.
- [9] G. Ciardo, J. Muppala, and K. Trivedi, "SPNP: Stochastic Petri net package", *Proc. Int. Workshop on Petri Nets and Performance Models*, IEEE Computer Society Press, Los Alamitos, CA, Dec. 1989, pp 142-150.
- [10] The Open Cluster Group, "How to Install an OSCAR Cluster, Software Version 2.2, Documentation Version 2.2", February 27, 2003.
- [11] O. Kolesnikov, and B. Hatch, "Building Linux Virtual Private Networks". New Riders Publishing, February 2002.
- [12] Hewlett Packard, "Managing MC/ServiceGuard", Hewlett-Packard Company, October 1998, pp. 60-68.
- [13] A.S. Tanenbaum, and M.S. Van, "Distributed Systems: Principles and Paradigms", July 2001, pp. 371-375.
- [14] A.S. Sathaye, R.C. Howe, and K.S. Trivedi, "Dependability Modeling of a Heterogeneous VAXcluster System Using Stochastic Reward Nets", *Hardware and Software Fault Tolerance in Parallel Computing Systems*, Ellis Horwood Ltd., 1992, pp. 33-59.
- [15] O. Ibe, A. Sathaye, R. Howe, and K.S. Trivedi, "Stochastic Petri Net Modeling of VAXcluster Availability", *Proc. Third Int. Workshop on Petri Nets and Performance Models (PNPM89)*, Kyoto, 1989, pp. 112-121.
- [16] D.I. Heimann, N. Mittal, and K.S. Trivedi, "Availability and Reliability Modeling for Computer Systems", *Advances in Computers*, 1990, Vol 31, pp 175-233.
- [17] M. Malhotra and K.S. Trivedi, "Dependability Modeling Using Petri-Nets", *IEEE Transactions on Reliability*, Sept. 1995. Vol. 44, No. 3, pp. 428-440.