A Generic Availability Model For Clustered Computing Systems

Hairong Sun, Jame J. Han, Haim Levendel
High Availability and Reliability Technology Center
Motorola, Elk Grove Village, IL 60007 USA
{ahs013, cjh048}@email.mot.com

Abstract

In this paper, we study the availability of clustered computing system with one cluster manager and "N+M" processing nodes, where M processing nodes serve as spares for the N active processing nodes. The functionality of individual processing node is dissected into application software, management software, OS and hardware. The dependency among these entities is considered. Stochastic Petri net models are constructed to investigate the cluster availability. In order to deal with the cluster with very large size, a solution based on state aggregation and fixed-point iteration is proposed. The existence and uniqueness of the fixed point is proved. The impact of cluster manager, switchover time and coverage ratio are quantitatively studied. From the numerical results of a simple cluster with "2+1" processing nodes, we find that: (1) the availability of cluster manager does not have significant impact to the system availability, (2) system availability increases with the coverage ratio and decreases with the switchover time. The mechanisms to improve the system availability are discussed.

In order to deal with the largeness of the state space, an approximate approach based on state aggregation was developed in [4]. An alternative approach, i.e., stochastic Petri net model, which can tolerate largeness by automating the generation of the large Markov model was proposed in [5].

The reliability for the Lucent Technologies Reliable Clustered Computing (RCC) was studied in [6][7]. Various levels of software fault tolerance were investigated and an escalating recovery model was constructed. It was found that higher system availability could be achieved over a cluster of computer with non-fault-tolerant architecture if higher coverage ratio could be well provided.

The analysis and models in the above-mentioned papers [4]-[7] were product-specific and based on some oversimplified assumptions, for example,

- The core of a cluster is a cluster manager dedicated to fault recovery, load-balancing and cluster maintenance, e.g., the Netfinity Manager for IBM’s Netfinity Cluster [1], the Watchdog in Lucent’s RCC [6][7]. However, the availability and failure/recovery behavior of the cluster manager were not studied in [4]-[7].

- In [6][7], failures were assumed to be Poisson, and the escalating recovery was modeled. In [4][5], the failures in VAXcluster were categorized into two types: permanent and intermittent. These assumptions on the failure behavior of clustered computing systems did not look into its internal structure. In a practical clustered system, the causes of failures are more complicated and heavily depend on the architecture of the processing nodes. The failure in the cluster might be caused by hardware, operating system (OS), application software or fault management software. Different failures should be recovered in different ways, e.g., application failure just needs a program restart, OS failure needs a reboot, and hardware failure needs repair. Traffic redirection should be activated when hardware or OS fails and should not be activated when failure is just caused by application software.

This paper presents a generic and comprehensive availability model to an "N+M" clustered computing...
system, where M processing nodes serve as spares for the N active processing nodes. A cluster manager is dedicated to recover the faults and balance the load in the cluster. Each processing node is comprised of hardware, operating system, application software and fault management software. Similarly, the cluster manager is comprised of hardware, operating system and fault management software. This kind of dissection gives us insight on the impact of software reliability and fault management to the system availability. Stochastic Petri nets are used to study the availability of the cluster. The dependency among the entities and traffic switchover/switchback are taken into considerations. The number of state for the Markov chain underlying the comprehensive Petri net model is $4N+17^{M+N}$, which will face largeness problem when the cluster size increases. In order to study the cluster with large size, an approach based on state aggregation and fixed-point iteration is proposed.

The paper is organized as follows. In section 2, fault management for a clustered architecture is discussed. In section 3, the failures in the cluster manager are discussed and the corresponding stochastic Petri Net model is proposed. In section 4, the failures in the processing nodes are addressed and the corresponding availability model is presented. System availability model, state aggregation and fixed-point iteration are discussed in section 5. Some numerical results and discussions are given in section 6. Conclusions are drawn in section 7.

2. Cluster architecture

![Figure 1 Clustered Architecture](image)

A clustered computing system is illustrated in Figure 1, which has redundant "N+M" configuration, with M processing nodes serving as spares for the N active processing nodes. Each processing node is comprised of hardware, operating system, application software and fault management software. At the heart of the clustered system is a dedicated recovery and maintenance processor known as the cluster manager or system Watchdog [6][7] which along with associated configuration management and fault recovery software running on the processing nodes, controls the cluster configurations. The cluster manager is comprised of hardware, operating system and fault management software.

The fault management software residing in the processing nodes may include two parts: (1) Fault detection, isolation and recovery for the local components, e.g., harddisk driver, power supply and fan, etc. (2) Fault detection, isolation and recovery associated with the cluster, e.g., communicate with the cluster manager and take action upon a command from the cluster manager.

The fault management software residing in cluster manager may include two parts as well: (1) Fault detection, isolation and recovery for the local components on the cluster manager, e.g., harddisk driver, power supply and fan, etc. (2) Fault detection, isolation and recovery associated with the cluster, e.g., detect failures in cluster, and then redirect traffic to the backup nodes.

3. Model for the cluster manager

3.1 Assumptions

- Time to failure for the fault management software is assumed exponentially distributed with mean $1/\lambda_{s.m}$. Time to detect the failure and restart the management software is assumed exponentially distributed with mean $1/\mu_{s.m}$.
- Time to failure for the OS is exponentially distributed with mean $1/\lambda_{s.o}$. Time to detect the failure and restart the OS and fault management software is exponentially distributed with mean $1/\mu_{s.o}$. The fault management software does not work when the OS fails.
- Time to failure for the hardware is exponentially distributed with mean $1/\lambda_{s.h}$. Time to detect the failure, repair the hardware and restart the OS/fault management software is exponentially distributed with mean $1/\mu_{s.h}$. The fault management and OS do not work when there is a hardware failure.

3.2 Analytical Model

Figure 2 gives the Petri Net model for the cluster manager. There are six places in the model:
- $P_{s.m\_on}$, one token in it means the management software is working
- $P_{s.m\_off}$, one token in it means the management software is not working
- $P_{s.o\_on}$, one token in it means the OS is working
- $P_{s.o\_off}$, one token in it means the OS is not working
- $P_{s.h\_on}$, one token in it means the hardware is working
• \( P_{s\_h\_off} \), one token in it means the hardware is not working

\[
\begin{align*}
\text{Figure 2 Stochastic petri net model for the cluster manager} \\
\text{There are six timed transitions in the model:} \\
\text{• The firing of transition } T_{s\_m\_f}, \text{ with rate } \lambda_{s\_m}, \text{ represents the failure of the management software.} \\
\text{• The firing of transition } T_{s\_m\_r}, \text{ with rate } \mu_{s\_m}, \text{ represents the restart of the management software.} \\
\text{• The firing of transition } T_{s\_o\_f}, \text{ with rate } \lambda_{s\_o}, \text{ represents the failure of the OS.} \\
\text{• The firing of transition } T_{s\_o\_r}, \text{ with rate } \mu_{s\_o}, \text{ represents the restart of the OS and deposits one token to } P_{s\_m\_on} \text{ and one token to } P_{s\_o\_on}. \\
\text{• The firing of transition } T_{s\_h\_f}, \text{ with rate } \lambda_{s\_h}, \text{ represents the failure of the hardware.} \\
\text{• The firing of transition } T_{s\_h\_r}, \text{ with rate } \mu_{s\_h}, \text{ represents the repair of the hardware and deposits one token to } P_{s\_m\_on}, P_{s\_o\_on} \text{ and } P_{s\_h\_on} \text{ respectively.} \\
\text{There are four immediate transitions:} \\
\text{• } ts1 \text{ and } ts2 \text{ are enabled when the OS is down, i.e., } #P_{s\_o\_on}=0 \\
\text{• } ts3 \text{ and } ts4 \text{ are enabled when the hardware is down, i.e., } #P_{s\_h\_on}=0
\end{align*}
\]

It can be easily found that the underlying Markov chain has 4 states by running Stochastic Petri Net Package (SPNP) [8], a software package developed in Duke University.

The cluster manager is considered as available if and only if there is one token in \( P_{s\_m\_on} \).

4. Models for the processing nodes

4.1 Assumptions

• Time to failure for the application software is exponentially distributed with mean \( 1/\lambda_{c\_a} \). Time to detect the failure and restart the application software is exponentially distributed with mean \( 1/\mu_{c\_a} \).

• Time to failure for the management software is exponentially distributed with mean \( 1/\lambda_{c\_m} \). Time to detect the failure and restart the management software is distributed with mean \( 1/\mu_{c\_m} \).

• Time to failure for OS is exponentially distributed with mean \( 1/\lambda_{c\_o} \). Time to detect the failure and restart the OS, fault management software and application software is exponentially distributed with mean \( 1/\mu_{c\_o} \). When the OS is shut down, the fault management and application software do not work. Switchover will be executed if there is sufficient resource in the system, e.g., if there are more than \( N \) equipments working in "N+M" configuration. The time for switchover is exponentially distributed with mean value \( 1/\lambda_{s3} \). After the reboot, traffic will be switched back. The switchback time is exponentially distributed with mean value \( 1/\lambda_{s4} \).

• Time to failure for the hardware is exponentially distributed with mean \( 1/\lambda_{c\_h} \). Time to detect the failure and repair the hardware is exponentially distributed with mean \( 1/\mu_{c\_h} \). \( 1/\mu_{c\_h} \) also includes the time to restart the OS, application software and management software. The fault management, application software and OS do not work when there is hardware failure. Switchover will be executed if there is sufficient resource in the system. The time for switchover is exponentially distributed with mean value \( 1/\lambda_{s1} \). After the repair and restart, traffic will be switched back. The switchback time is exponentially distributed with mean value \( 1/\lambda_{s2} \).

4.2 Analytical Models

4.2.1 Model without Switchover. We build a Petri net model for the processing node without switchover in case of the OS and hardware failure at first. Figure 3 gives the Petri Net model for single processing node in the cluster. We call this node as tagged node. The model is similar to the cluster manager model in Figure 2, except the places and transitions related to application software.

The guard functions for the six immediate transitions are as follows:

• \( tc1, tc2, tc3, \) and \( tc4 \) are enabled when the OS is down, i.e., \( #P_{c\_o\_on}=0 \)

• \( tc5 \) and \( tc6 \) are enabled when the hardware is down, i.e., \( #P_{c\_h\_on}=0 \)

Running SPNP, it is found the underlying Markov chain has 6 states.

The tagged processing node is considered as available if and only if there is one token in \( P_{c\_a\_on} \).
4.2.2 Model with Switchover. Now we build a Petri net model with switchover in case of the OS and hardware failure. Compared with the model in Figure 3, the newly added parts in Figure 4 are highlighted in the dashed box.

- One token in $P_{ns1}$ represents that the traffic is being redirected to the redundant nodes because of the hardware failure.
- One token in $P_{ns2}$ represents that the traffic is being switched back.
- One token in $P_{ns3}$ represents that the traffic is being redirected to the redundant nodes because of the OS failure.
- One token in $P_{ns4}$ represents that the traffic is being switched back.
- The firing of transition $T_{s1}$ represents the successful switchover after the hardware failure.
- The firing of transition $T_{s2}$ represents the successful switch back after the hardware is repaired.
- The firing of transition $T_{s3}$ represents the successful switchover after the OS failure.
- The firing of transition $T_{s4}$ represents the successful switch back after the OS is restarted.
- The firing of $t_7$ represents that the switchover has not been completed yet (because of the longer switchover time) when the hardware is repaired.
- The firing of $t_8$ represents the switchover has not been completed yet (because of the long switchover time) after the OS is restarted.
- The firing of $t_{10}$ represents that the hardware fails during the switchover. It is enabled when the hardware is down, i.e., $#(P_{c \_h \_on})=0$.

- The firing of $t_9$ represents that the hardware fails during the switchback. It is enabled when the hardware is down, i.e., $#(P_{c \_h \_on})=0$.

Running SPNP, it is found the underlying Markov chain has 17 states.

The tagged processing node is considered as available if and only if there is one token in $P_{c \_a \_on}$.

5. System model and availability

5.1 System availability model

Putting the models of the cluster manager (Figure 2) and tagged node (Figure 4) together, we have the system model shown in Figure 5.

Switchover may be executed when there are OS/hardware failures and sufficient resources in the cluster, i.e., there are more than N equipments being working. The switchover also depends on the status of the server. Therefore, the guard functions for the transitions accomplishing switchover and switchback in Figure 5, i.e., the transitions in the dashed ovals, are

\[ #P_{s \_m \_on} = 1 \]

\&\& (number of working node is greater than N) \hspace{1cm} (1)
Expression (1) reveals the dependency between the cluster members.

Figure 5 System Availability Model
The model shown in Figure 5 will face largeness problem as the cluster size increases. The number of states for the stochastic Petri net shown in Figure 4 is 17, and the number of states corresponding to the stochastic Petri net in Figure 5 is $4 \times 10^{30}$ N. When the cluster size is larger than 7, the underlying Markov chain will have billions states! In this section, we will present an alternative solution, which is based on state aggregation and fixed-point iteration.

5.2 State aggregation

The Markov chain corresponding to the stochastic Petri net model shown in Figure 4 can be aggregated to the following four macro-states.

5.2.1 State 1. In state 1, the tagged node is working and loaded with traffic, i.e., all the following conditions are met:
- The hardware is functional.
- The OS is functional
- The application is functional
- $(P_{ns1})=(P_{ns2})=(P_{ns3})=(P_{ns4})=0$

The probability is denoted as $p_1$.

5.2.2 State 2. In state 2, the tagged node is not working but still receiving traffic, i.e., one of the following conditions is met:
- The application is not working, hardware and OS are working and $(P_{ns2})=(P_{ns4})=0$.
- The hardware is not working and $(P_{ns1})=1$.
- The OS is not working and $(P_{ns3})=1$.

The probability is denoted as $p_2$.

5.2.3 State 3. In state 3, the tagged node is working, but traffic is switched to the redundant equipment and has not been switched back yet, i.e., one of the following conditions is met:
- The application, OS and hardware are working and $(P_{ns2})=1$.
- The application, OS and hardware are working and $(P_{ns4})=1$.

The probability is denoted as $p_3$.

5.2.4 State 4. In state 4, the tagged node is not working, and traffic has been switched to the redundant equipment, i.e., one of the following conditions is met:
- The hardware is not working and $(P_{ns1})=0$.
- The OS is not working and $(P_{ns3})=0$.

The probability is denoted as $p_4$.

5.3 Fixed-point iteration

The key point to deal with the largeness problem in Figure 5 is avoiding using the model in Figure 5. We do not need to put the cluster manager model and processing node model together to study the availability of the cluster. We just need to study the cluster manager model in Figure 2 and processing node model in Figure 4, separately. Because the cluster manager does not depend on the processing nodes, the model in Figure 2 can be solved independently. However, in Figure 4, the transitions $T_{s1}, T_{s2}, T_{s3}$ and $T_{s4}$ are not only guarded by the state of the cluster manager, but also depend on the number of working nodes in the cluster. The former can be obtained by solving the cluster manager model. However, the latter is a random variable whose distribution depends on the solution of the processing node model! This means the input parameter of the model in Figure 4 depends in the output of the model. This kind of dilemma can be solved with fixed-point iteration.

- Step 1: solve the model in Figure 2 and calculate the probability of the cluster manager being working, $p_w$.
- Step 2: solve model in Figure 4 with $T_{s1}, T_{s2}, T_{s3}$ and $T_{s4}$ always being enabled, calculate the probability of the tagged node being at state 1, i.e., $P_{1\text{(on)}}$
- Step 3: solve model in Figure 4 with $T_{s1}, T_{s2}, T_{s3}$ and $T_{s4}$ always being disabled, calculate the probability of the tagged node being at state 1, i.e., $P_{1\text{(off)}}$
- Step 4: sum up the probabilities of the tagged node being at state 1,

$$P_i = cp_w \Phi P_i(\text{on}) + (1 - cp_w \Phi) P_i(\text{off}) \quad (2)$$

where $\Phi$ is the probability that there are more than $N$ nodes working, and $c$ ($0 \leq c \leq 1$) is the fault coverage, the
probability that the failures in the processing nodes can be detected and recovered

- Step 5: calculate $\Phi$ with the following equation

$$
\Phi = \sum_{i=N+1}^{N+M} \frac{(N+M)!}{i!(N+M-i)!} p_1^i (1-p_1)^{N+M-i}
$$

(3)

- Repeat step 4 and 5 to get the fixed-point solution for $\Phi$ and $p_1$ and calculate $p_2$, $p_3$ and $p_4$
- Stop

Expression (2) and (3) are the fixed-point equations for $p_1$ and $\Phi$. With the fixed-point iteration, we just need to solve the model in Figure 2 once and the model in Figure 4 twice. The former only has 4 states and the latter has 17 states. Recalling the number of states for the model in Figure 5 is $4 \cdot 17^{M,N}$. The number of states shrinks significantly.

Next we will prove that the solution to (2) and (3) exists and is unique. Before doing that, we prove two Lemmas.

[Lemma 1] $p_1$ is a decreasing function with $\Phi$ as expressed in (2).

[Proof] The definition of state 1 is that hardware/OS/application is functional and the node is loaded with traffic. Whether $T_{s1}$, $T_{s2}$, $T_{s3}$ and $T_{s4}$ are enabled or disabled, the probability of hardware/OS/application being functional does not change. However, it does affect the probability of the equipment being loaded with traffic. When $T_{s1}$, $T_{s2}$, $T_{s3}$ and $T_{s4}$ are disabled, there is neither traffic switch over nor switchback. Therefore, the equipment is more likely loaded with traffic when $T_{s1}$, $T_{s2}$, $T_{s3}$ and $T_{s4}$ are disabled, which means $p_{(0,0)} > p_{(1,0)}$. Therefore, $p_1$ is a decreasing function with $\Phi$ as expressed in (2).

[Lemma 2] $\Phi$ is an increasing function with $p_1$ as expressed in (3)

[Proof] Denote $a_i$ as

$$
a_i = \frac{(N+M)!}{i!(N+M-i)!} p_1^i (1-p_1)^{N+M-i}
$$

(4)

We have

$$
\frac{\partial a_i}{\partial p_1} = \frac{(N+M)!}{i!(N+M-i)!} [i-(N+M)p_1]
$$

which means there exists an $i$, $\frac{\partial a_i}{\partial p_1} \geq 0$, when $i \geq (N+M)p_1$

$\frac{\partial a_i}{\partial p_1} < 0$, when $i < (N+M)p_1$

Recalling

$$
\sum_{i=N+1}^{N+M} a_i = 1
$$

which means

$$
\sum_{i=N+1}^{N+M} \frac{\partial a_i}{\partial p_1} = 0
$$

Therefore, we have

$$
\frac{\partial a_i}{\partial p_1} > 0
$$

$\Phi$ is an increasing function with $p_1$ as expressed in (3).

With Lemma 1 and Lemma 2, we have

**[Theorem 1]** The solution to $\Phi$ and $p_1$ exists and is unique (as shown in Figure 6).

![Figure 6 Existence and uniqueness of the fixed point](image)

5.4 Weighted availability

For a cluster with $N+M$ processing nodes and evenly distributed load, each node is therefore assigned with $1/(N+M)$ load. When one of the nodes is down, its load will be shifted to the other working nodes. If the switchover is successful, then the cluster can still provide full capacity. However, if the switchover fails, the cluster can only provide $(N+M-1)/(N+M)$ capacity. Generally, assume there are

- $j$ nodes being working and loaded with traffic
- $k$ nodes being working but without traffic because the traffic has been redirected and has not been switched back.

The cluster can then provide $\min\{(j+k)/(N+M), j/N\}$ capacity.

The weighted availability should be calculated accordingly.

**[Theorem 2]** For an “N+1” cluster, the weighted availability is given as
\[ A = \sum_{i=0}^{N+1} \sum_{j=0}^{N+1-i-j} \sum_{m=0}^{N+1-i-j} \min\{(i+m)/(N+1), i/N(N+1)!p_1^i p_2^j (p_3 + p_4)^m \} \frac{i! \beta(N+1-i-j)!}{i! \beta(N+1-i-j)!} \]

where \(p_1, p_2, p_3\) and \(p_4\) are the probabilities of the tagged node being at macro state 1, 2, 3 and 4 respectively.

**Proof** For an “N+1” cluster,

- when \(N+1\) processing nodes are in state 1 OR N in state 1 and one in state 3 (or 4), the cluster can provide full capacity.
- when \(N\) processing nodes are in state 1 and one in state 2, the system can provide \(N/(N+1)\) capacity
- when \((N-1)\) processing nodes are in state 1 and two in state 2, the system can provide \((N-1)/(N+1)\) capacity.
- when \((N-1)\) processing nodes are in state 1 and at least one in state 3 (or 4), the system can provide \((N-1)/N\) capacity
- when \((N-2)\) processing nodes are in state 1 and three in state 2, the system can provide \((N-2)/(N+1)\) capacity
- when \((N-2)\) processing nodes are in state 1 and at least one in state 3 (or 4), the system can provide \((N-2)/N\) capacity

Summing them up, we can get expression (6).

Extending Theorem 2, we have

**Theorem 3** For an “N+M” cluster, the weighted availability is given as

\[ A = \sum_{i=0}^{N+M} \sum_{j=0}^{N+M-i-j} \sum_{m=0}^{N+M-i-j} \min\{(i+m)/(N+M), i/(N+M)\}(N+M)!p_1^i p_2^j (p_3 + p_4)^m \frac{i! \beta(N+M-i-j)!}{i! \beta(N+M-i-j)!} \]

where \(p_1, p_2, p_3\) and \(p_4\) are the probabilities of the tagged node being at macro state 1, 2, 3 and 4 respectively.

### 6. Numerical results and discussions

In this section, we present some numerical results for a “2+1” cluster. The models are solved with the software package developed in Duke University [8]. The switchover and switchback time are assumed same and denoted as Ts. The mean time to failure for the processing node is 30 days for the application software and management software, 100 days for the OS and 1 year for the hardware. The mean time to repair for the processing node is 3 seconds for the application software and management software, 3 minutes for the OS and 4 hours for the hardware. The parameters are summarized in Table 1. The availability of individual node is therefore 0.999521607278.

The mean time to failure for the cluster manager is 30 days for the management software and 100 days for OS.

(6) The mean time to repair for the cluster manager is 3 seconds for the management software, 3 minutes for the OS and 4 hours for the hardware. The parameters are summarized in Table 2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1/\lambda_{c-a})</td>
<td>30 days</td>
</tr>
<tr>
<td>(1/\lambda_{c-m})</td>
<td>100 days</td>
</tr>
<tr>
<td>(1/\lambda_{c-o})</td>
<td>1 year</td>
</tr>
<tr>
<td>(1/\lambda_{c-h})</td>
<td>3 seconds</td>
</tr>
<tr>
<td>(1/\mu_{c-a})</td>
<td>3 seconds</td>
</tr>
<tr>
<td>(1/\mu_{c-m})</td>
<td>3 minutes</td>
</tr>
<tr>
<td>(1/\mu_{c-o})</td>
<td>4 hours</td>
</tr>
</tbody>
</table>

**Table 1 Parameters for the tagged node**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1/\lambda_{c-a})</td>
<td>30 days</td>
</tr>
<tr>
<td>(1/\lambda_{c-m})</td>
<td>100 days</td>
</tr>
<tr>
<td>(1/\lambda_{c-o})</td>
<td>3 minutes</td>
</tr>
<tr>
<td>(1/\lambda_{c-h})</td>
<td>240 min</td>
</tr>
</tbody>
</table>

**Table 2 Parameters for the cluster manager**

6.1 The impact of cluster manager

Assume mean switchover time being 1 minute and coverage ratio being 0.9 respectively. Table 3 lists the system availability vs. the availability of the cluster manager. It can be seen that

- The benefit brought by increasing the availability of cluster manager is not significant. In this example, when we improve the availability of the cluster manager from 0.999065621 to 0.999944947, we can only improve the system availability from 0.999944043 to 0.999944440.
- 4-nine system availability can be achieved with 3-nine cluster manager in this example.

Intuitively, the cluster manager providing cluster-level fault management should be highly available in order to achieve higher system availability. Based on this intuition, it was strongly recommended in [9] that the fault management software should run on fault-tolerant computers which can perform logic self-checking and have all of the main components (e.g., CPU, memory, I/O controller, bus, power supply and disk, etc.) physically duplicated. However, according to our study, this is not necessarily true. The reason is that the failure of cluster manager will not bring down the whole system, and it has impact only when one cluster member is down and traffic redirection is required.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1/\lambda_{c-h})</td>
<td>0.5 year</td>
</tr>
<tr>
<td>(1/\lambda_{c-h})</td>
<td>1 year</td>
</tr>
<tr>
<td>(1/\lambda_{c-h})</td>
<td>2 years</td>
</tr>
<tr>
<td>(1/\lambda_{c-h})</td>
<td>4 years</td>
</tr>
<tr>
<td>(1/\lambda_{c-h})</td>
<td>16 years</td>
</tr>
</tbody>
</table>

**Table 3 System availability vs. the availability of the cluster manager**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability of cluster manager</td>
<td>0.99907</td>
</tr>
<tr>
<td>Availability of cluster manager</td>
<td>0.99953</td>
</tr>
<tr>
<td>Availability of cluster manager</td>
<td>0.99975</td>
</tr>
<tr>
<td>Availability of cluster manager</td>
<td>0.99986</td>
</tr>
<tr>
<td>Availability of cluster manager</td>
<td>0.99995</td>
</tr>
<tr>
<td>Availability of cluster manager</td>
<td>0.99994</td>
</tr>
</tbody>
</table>

247
6.2 The impact of coverage ratio

Assume mean switchover time being 1 minute and 1/λ_c_h=1 year, we study the impact of coverage ratio to the system availability in Table 4. The availability of the cluster manager is 0.999526073. It can be seen that

- Coverage ratio plays a critical role in the system availability. Increasing the coverage ratio from 0 to 100%, we can improve the system availability from 0.999521607 (3 nines) to 0.999991398 (5 nines).
- 3-nine system availability can be achieved when the coverage ratio is 100% while availability of the management server is 3-nine (0.999526073).

Because of the complexities and uncertainties of the faults in the clustered system, 100% coverage is a very tough objective to reach. Improving the quality of fault management software, incorporating artificial intelligence [10][11], and neural networks [12] are solutions to enhance the fault coverage ratio.

<table>
<thead>
<tr>
<th>c</th>
<th>0</th>
<th>0.9</th>
<th>0.99</th>
<th>0.999</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.99952</td>
<td>0.99961</td>
<td>0.99971</td>
<td>0.9999</td>
<td>0.9999</td>
</tr>
</tbody>
</table>

Table 4 System availability vs. coverage ratio

6.3 The impact of switchover time

Assume 1/λ_c_h=1 year, coverage ratio being 0.9, we study the impact of mean switchover time to the system availability in Table 5. It can be seen that shorter switchover time can improve the system availability. In this example, when we reduce the switchover time from 4 minutes to 30 seconds, we can improve the system availability from 0.999933174 to 0.999947085. It is significant in high-availability system.

Reducing switchover time could be achieved by high-performance and low overhead interconnect between the processing nodes [13], simplified communication protocols [14], and deliberate design on the node status monitoring and keep-alive messages [15]. It is reported in [15] that Tandem's Nonstop Clusters Application Protection System offers a total failover time of approximately 10 seconds.

<table>
<thead>
<tr>
<th>Ts (second)</th>
<th>30</th>
<th>60</th>
<th>180</th>
<th>240</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.999947</td>
<td>0.999944</td>
<td>0.999936</td>
<td>0.99993</td>
</tr>
</tbody>
</table>

Table 5 System availability vs. switchover time

7. Conclusions

Analytical models were proposed to evaluate the system availability of clustered computing systems. We divided the functionality of cluster manager into management software, OS and hardware, and dissected the functionality of the cluster nodes into application software, management software, OS and hardware. The Petri net models took into account the dependency and switchover/switchback in the cluster. State aggregation along with fixed-point iteration was proposed to solve the largeness problem. The existence and uniqueness of the fixed-point was proved. The impact of cluster manager, switchover time and coverage ratio was quantitatively studied with the models.

Reference