A Hierarchical Availability Model for Multiprocessor Computers

A. Gaber Mohamed
Salim Hariri
Hasan B. Mutlu

CRPC-TR92211
April, 1992

Center for Research on Parallel Computation
Rice University
P.O. Box 1892
Houston, TX 77251-1892

A HIERARCHICAL AVAILABILITY MODEL FOR MULTIPROCESSOR COMPUTERS

A. Gaber Mohamed
Northeast Parallel Architectures Center (NPAC)
Syracuse University, 111 College Place, Syracuse, New York
agm@nova.npac.syr.edu

Salim Hariri
Electrical and Computer Engineering
Syracuse University, 111 Link Hall, Syracuse, New York
hariri@snow-white.ece.syr.edu

and

Hasan B. Mutlu
AT&T Bell Laboratories, Naperville, Illinois

Abstract-- The importance of availability of parallel computers increases as their uses continuously spread to many areas where failures can have catastrophic costs. The increased complexity of such systems and the sophistication of the availability measures of interest makes evaluation of these metrics more challenging. The most promising practical evaluating techniques are based on Markovian and network graph models. In this paper, we propose a two-level availability model, that integrates Markovian model and network graph techniques, to analyze the availability of large parallel computers.

1. INTRODUCTION

Markovian models are the most widely used technique because of their generality and applicability to model a large class of applications. For modeling the availability of parallel systems with complex characteristics (complex interactions between components, complex criteria for a system to be operational, etc.), the number of states grows exponentially (2^n states, where n denotes the number of components). Consequently, the solution of a Markovian model becomes intractable; most acceptable techniques tackle this problem via truncation of state space or aggregation of states [1, 2].

The techniques based on network graphs are simple and can be used to model the availability of computing systems of reasonably large size. However, the main limitation of this method is the difficulty in modeling complex system behaviors.

Our approach integrates these two techniques into one hierarchical model that has all the advantages of these methods but none of their limitations [3, 4]. We use Markovian technique to model component availabilities which will enable us to take into consideration dependent failures among components, software failures, performance constraints, etc. At the component level, the number of states for most system components is relatively small and closed form solutions are obtainable. At the system level, we use a network graph to describe the interactions and the connectivities among the components of the system.

2. PROPOSED HIERARCHICAL AVAILABILITY MODEL

2.1. Modeling Availability at the Component Level

Figure 1 shows a 4-state Markovian model to evaluate the availability of a processing unit that takes into consideration the failures caused by hardware and software. The definitions of notations and states used in this model are as follows:

λ_h hardware failure rate from the healthy state.

λ_s permanent software failure rate from the healthy state.
λ_{se} transient software failure rate from the healthy state.
λ_{sf} permanent software failure rate from the transient software failure state.
μ_h hardware failure repair rate from the hardware failure state.
μ_s software failure repair rate from the permanent software failure state.
μ_{se} software fast recovery rate from the transient software failure state.
μ_{sf} software recovery rate from the permanent software.

state 0 — correct operation state- the state in which a *PU* physically operates correctly and has no software errors.

state 1 — hardware failure state- the state in which a *PU* failure occurs due to a hardware or environmental problem. Typical environmental problems are power outages or errors caused by improper operation of the component.

state 2 — transient software failure state- the state in which a *PU* failure occurs due to a transient software error.

state 3 — permanent software failure state- the state in which a *PU* failure occurs due to a permanent software error.

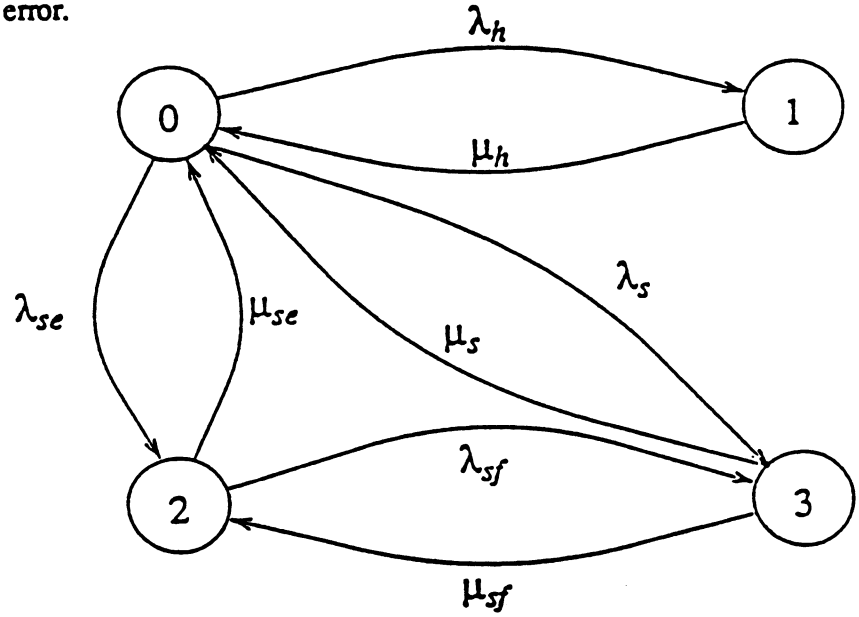

Figure 1. Transition Diagram for the 4-State Markovian Model.

By solving the Markovian model shown in Figure 1, the steady-state availability of a processing unit can be given as

$$A = \frac{\mu_h(\lambda_{sf}\mu_s + \mu_s\mu_{se} + \mu_{se}\mu_{sf})}{(\lambda_h + \mu_h)(\lambda_{sf}\mu_s + \mu_s\mu_{se} + \mu_{se}\mu_{sf}) + \mu_h(\lambda_s\mu_{sf} + \lambda_{se}\mu_s + \lambda_{se}\mu_{sf} + \lambda_s\lambda_{sf} + \lambda_{se}\lambda_{sf} + \lambda_s\mu_{se})}. \quad (2.1)$$

It is interesting to note that as we omit the failures due to the software (i.e., λ_{se} and λ_s), equation (2.1) is reduced to a two-state Markovian model. That is,

$$A = \frac{\mu_h}{\mu_h + \lambda_h} = \frac{MTTF}{MTTF + MTTR}. \quad (2.2)$$

2.2. Modeling Dependent Failures of Components

Most of the techniques developed to analyze availability assume that individual components fail independently. This tends to significantly simplify the computation even though it is not a

realistic assumption for some applications [5]. This subsection demonstrates a method to incorporate the effects of dependent failures into component models by properly modifying their failure rates. Consequently, the components at the system level can be assumed independent because the failure dependencies have already been accounted for at the component level.

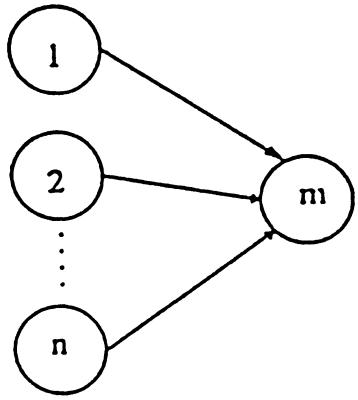

Figure 2. An example of components with failure dependencies.

Figure 2 shows a case of one component m whose failure depends on its own failure rate as well as on the failure rates of n other components. The effective failure rate that takes into consideration this type of strong dependency can be given as,

$$E\lambda_m = \lambda_m + \sum_{i=1}^{n} \lambda_i \tag{2.3}$$

where λ_i denotes the failure rate of component i.

In order to incorporate the effects of any correlation among the components, we introduce a dependency factor ρ_{im} that corresponds to the probability that the failure of component i will lead to a failure in component m. Hence, the effective failure rate of component m can be given as,

$$E\lambda_m = \lambda_m + \sum_{i=1}^{n} \rho_{im}\, \lambda_i . \tag{2.4}$$

2.3. Modeling Availability at the System Level

In the following, we use a fault-tolerant database system as an example to demonstrate the use of network graphs in modeling the system availability. Also, we compare our approach to solve this example with a Markovian method that gives exact results. Figure 3 shows a model of a fault-tolerant database system which comprises of the following components: a front-end processor, a database, and two processing subsystems formed by a switch, a memory and two processors [2]. Table 1 lists the failure and repair rates of these components. It is assumed that the failure of a processor might contaminate the database subsystem with probability 0.001. The links connecting these components are assumed to be perfectly reliable.

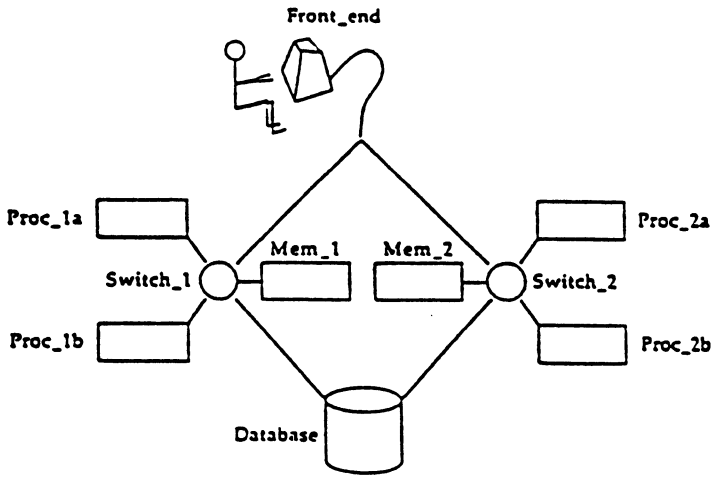

Figure 3. A fault-tolerant database system.

component	Mean Failure Rate (hours)	Mean Repair Rate (hours)	Availability
Database	1/2400	1	0.9995500
Frontend	1/2400	1	0.9995835
Switch	1/2400	1	0.9995835
Memory	1/2400	1	0.9995835
Processor	1/120	1	0.9917355

Table 1. Component failure rates and their availabilities.

The system is considered available (functioning properly) when a transaction (task) initiated from the front-end processor can successfully access the database subsystem. Hence, the success behavior of the system can be described by a probabilistic graph as shown in Figure 4.

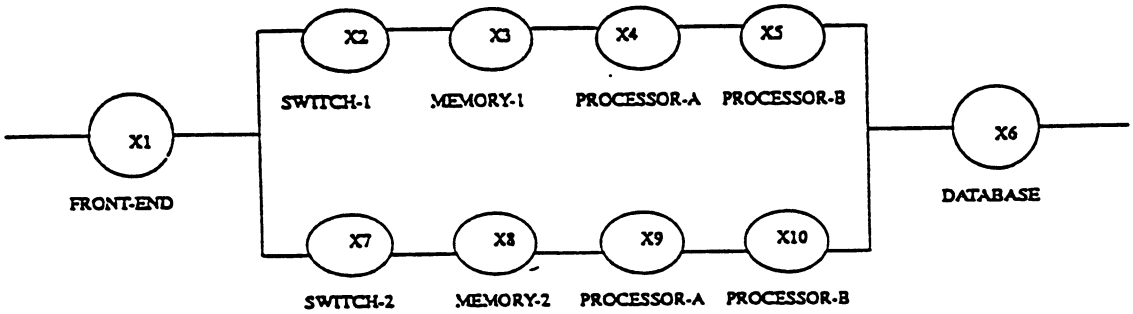

Figure 4. A network graph for the fault-tolerant database system.

The system availability expression can be obtained directly as,

$$A\ (system) = A_1 A_2 A_3 A_4 A_5 A_6 + A_1 (1 - A_2 A_3 A_4 A_5) A_7 A_8 A_9 A_{10} A_6. \qquad (2.5)$$

where A_i denotes the availability of component i.

Discussion
Evaluating the availability of the system shown in Figure 3 using a Markovian method requires determining and manipulating a state-transition matrix and state space of order 1000 (since there are 10 components and each one has only two states). The numerical result obtained without approximation is 0.9988353, as reported by Muntz *et al* [2].

If a two state-Markovian model is used to represent component behaviors, the component availability can then be evaluated using Equation 2.2 (obtained by solving a two-state Markovian model). Since the failure of a processor might lead to a database failure with a probability of 0.001, Equation 2.4 will be used to obtain the effective database failure rate. Once the component availabilities have been evaluated, system availability can be computed by substituting these values in the system availability expression (Equation 2.5). If this is done, the same result (0.9988355) is obtained by solving a Markovian model without approximation. Also, if the effect of failure dependencies on the database unit is ignored, the system availability is 0.9988688, which is a reasonable approximation of the solution. The objective of solving this example using this hierarchical availability model is to demonstrate its simplicity (it tackles Markovian models with only a few states), accuracy, and potential applicability to analyze the availability of complex and large computing systems.

3. ANALYZING THE AVAILABILITY OF ALLIANT FX/80 MINISUPERCOMPUTER

The failures of this computer have been recorded for one year (from January 1989 until the end of December 1989) and will be analyzed to obtain the parameters of the Markovian models introduced in Section 2.1. The observed failures can be classified into four different types:
1. Type 1 — hardware and environmental failure.
This type of failure is caused by power outages and other hardware related failures. Table 2 shows all the information relevant to this failure type, including the occurrence of a failure time to failure (TTF), time to repair it (TTR), mean time to failure (MTTF), and mean time to repair (MTTR)

Date	Time went down	Time brought up	TTR minutes	TTF minutes
01-26	04:40	10:40	360	
03-14	05:30	06:40	070	67370
03-14	06:50	10:00	190	00010
05-15	06:30	09:30	180	89070
05-22	06:00	08:30	150	09870
07-25	06:45	09:15	150	92055
08-27	02:45	08:00	315	47130
08-29	18:25	19:15	050	03505
10-19	17:30	18:15	045	73335

MTTF: 47,793 MTTR: 167.8

Table 2. Hardware and Environmental Failures.

2. Type 2— transient software failure.
This failure occurs when the system hangs up (no more users can login to the system, but users already logged-in are not affected by this failure). The recovery procedure for this type is simple and usually takes less than 30 minutes. Table 3 shows all the information related to this failure type.

Date	Time went down	Time brought up	TTR minutes	TTF minutes
01-17	12:55	13:15	20	
01-20	09:30	09:50	20	004095
04-24	09:00	09:26	26	135310
07-22	09:00	09:10	10	128134
09-01	09:25	09:40	15	057615
09-22	16:15	16:35	20	030635
11-16	14:30	14:45	15	079075
11-17	14:30	14:45	15	001425

MTTF: 62,327 MTTR: 17.63

Table 3. Transient software failures.

3. Type 3— semi-permanent software failure.
This failure occurs when the transient software failure leads to a permanent software failure. The recovery procedure is done by partially recovering the system from the permanent failure state to the transient software failure state. At this point, a fast recovery is applied to move the system to the healthy state. The total recovery time varies between 30 and 60 minutes. Table 4 shows all the information related to this failure type.

Date	Time went down	Time brought up	TTR minutes	TTF minutes
01-04	10:40	11:10	30	
01-04	11:30	12:15	45	000020
01-13	15:20	16:00	40	013145
05-18	08:00	08:30	30	179520
06-07	13:15	14:05	50	027645
11-02	08:40	09:16	36	212795

MTTF: 86,625 MTTR: 38.50

Table 4. Semi-permanent software failures.

4. Type 4— permanent software failure.
This failure moves the system directly to a single user mode (a crash state). The system can be recovered only by shutting it down and booting it up again. The recovery procedure takes more than 60 minutes. Table 5 shows all the information relevant to this failure type.

Date	Time went down	Time brought up	TTR minutes	TTF minutes
04-07	10:00	11:02	0062	
04-10	12:30	13:30	0060	04408
04-17	10:00	11:00	0060	04870
04-24	12:00	13:30	0090	10140
06-18	08:00	12:00	0240	78870
07-07	17:00	19:25	0145	27660
07-18	13:18	15:00	0102	15473
07-18	22:00	23:45	0105	00420
07-27	08:00	10:30	0150	12015
09-06	11:15	12:15	0060	59085
09-17	14:00	07:25	1055	15945
09-20	16:45	07:45	0900	03440
11-06	10:10	11:12	0062	66385
11-09	08:50	10:00	0070	05618
11-17	09:20	10:50	0090	11480
12-14	09:35	11:05	0090	38805
12-21	10:00	17:00	1860	10015

MTTF: 23,102 MTTR: 306

Table 5. Permanent software failures.

3.1. Evaluating Component Availabilities

The mean failure rates shown in Tables 2-5 will be used to evaluate the component availabilities. It is also assumed that the memory units (main memory and cache subsystems), memory bus, and the ACE-switch can be modeled using only a two-state Markovian model and thus, Equation 2.2 can be used to evaluate their availabilities. These components experience only hardware and environmental failures (Type 1). However, the processing units (IPs and ACEs) experience hardware and software failures (Types 1-4). Therefore, model developed in Section 2.1 can be used to evaluate the processing unit availability. This availability can be computed by substituting the means shown in Tables 2-5 in Equation 2.1. Table 6 shows the mean time to failure (MTTF), mean time to repair (MTTR), and the availabilities of the Alliant FX/80 components.

component name	MTTF (hours)	MTTR (hours)	availability
Memory, Cache	796.55	2.80	0.99650
Switch, Bus	796.55	2.80	0.99650

component name	$\frac{1}{\lambda_h}$ (hours)	$\frac{1}{\mu_h}$ (hours)	$\frac{1}{\lambda_s}$ (hours)	$\frac{1}{\mu_s}$ (hours)	$\frac{1}{\lambda_{se}}$ (hours)	$\frac{1}{\mu_{se}}$ (hours)	$\frac{1}{\lambda_{sf}}$ (hours)	$\frac{1}{\mu_{sf}}$ (hours)	availability
CE, IP	796.6	2.80	385	5.10	1038.8	0.29	1443.8	0.64	0.99408

Table 6. Component availabilities of Alliant FX/80.

3.2. Evaluating System Availability

Once the component availabilities have been evaluated, the next step is to describe the system behavior by a network graph. If we assume that the system is not available when at least one component is down, then system availability is a direct product of all its component availabilities, as illustrated in the following equation :

$$A\ (FX/80) = \prod_{\text{for all component } i} A_i = 0.897969.$$

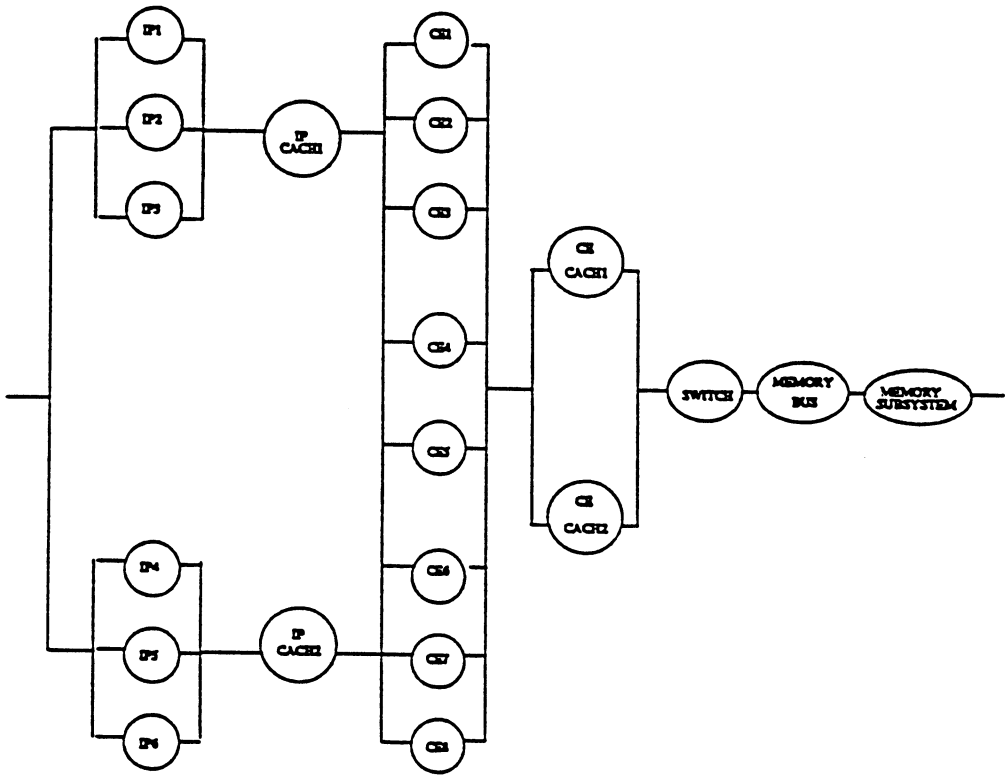

Figure 5. A network graph for Alliant FX/80.

However, the system can be operational with degraded performance when at least one configuration (IP, ACE, IP-cache, ACE-cache, memory system, and memory bus) is in operational state. The network graph corresponding to this system is shown in Figure 5; the system availability expression is,

$$A\ (FX/80) = (1 - U_{IP}^{3})\, A_{cache}\, (1 - U_{CE}^{8})\, (1 - U_{cache}^{2})\, A_{switch}\, A_{bus}\, A_{mem}\, [1 + U_{cache} + A_{cache}\, U_{IP}^{3}]$$

where U_i denotes the unavailability of component i which is equal to $1 - A_i$.

If the component availabilities are substituted in the above equation, this yields a system availability equal to 0.989516. In this section we demonstrated the applicability of the proposed model to analyze the availability of a parallel computer. If a Markovian model is used to solve this example without approximation, the state space can be of order one million states and this makes it infeasible to construct the state transition matrix and solve this Markovian chain. This limitation has been eliminated in the proposed hierarchical model, and thus can be used to model larger and more complex parallel computers.

4. SUMMARY AND CONCLUDING REMARKS

A hierarchical availability model has been proposed to analyze the availability of parallel

computers. At the component level, Markovian models are used to model sophisticated dependability measures that take into account software and hardware failures and some performance constraints. At the system level, the system is modeled as a probabilistic graph so that any efficient availability algorithm developed for communication networks can also be used to evaluate the overall system availability. The availability of a fault-tolerant database system evaluated by other methods has been solved by our model to demonstrate the simplicity, validity, accuracy, and potential applicability to analyze large parallel computers.

5. ACKNOWLEDGEMENT

This work was conducted using the computational resources of the Northeast Parallel Architectures Center (NPAC) at Syracuse University, which is funded by and operates under contract to DARPA, and the Air Force Systems command, Rome Air Development Center (RADC), Griffiss AFB, Rome, NY, under contract #F30602-88-C-0031. We also like to thank the Director and Deputy Director of NPAC for supporting this work and the NPAC Computer Operations Staff for providing the data used in this research.

6. REFERENCES

[1] R. Johnson, "Network Reliability and Acyclic Orientation," *Networks*, Vol. 14, pp. 489-505.

[2] R. Muntz, E. Silva, and A. Goyal, "Bounding Availability of Repairable Computer Systems," *IEEE Transactions on Computers*, December 1989, Vol. 38, pp. 1714-1723.

[3] S. Hariri, A. G. Mohamed, and H. B. Mutlu, "Modeling Availability of Parallel Computers," *ICPP 1990 Proceedings*, August 1990, Vol. I, pp. 559-560.

[4] S. Hariri, C. S. Raghavendra, "SYREL: A Symbolic Reliability Algorithm based on Path and Cutset Methods," *IEEE Transactions on Computers*, October 1987, Vol. C-36, pp. 1224-1232.

[5] S. N. Pan and J. Spragins, Dependence Failure Reliability Models for Tactical Communications Network," *IEEE Proceedings of the International Conference on Communications*, June 1983, pp. C5.5.1-7.