

# Modeling High Availability Systems

Kishor S. Trivedi, Ranjith Vasireddy  
Department of ECE, Duke University  
Durham, NC 27708, USA  
{kst, rv5}@ee.duke.edu

David Trindade, Swami Nathan, Rick Castro  
Sun Microsystems, USA  
{david.trindade, swami.nathan, rick.castro}@sun.com

## Abstract

*Carrier grade high availability platforms are designed to enable the development and deployment of highly available services in the telecommunications industry. In order to build-in high availability and compare availabilities that differ in the sixth decimal place during the design phase, fairly detailed stochastic models are needed to evaluate the design and perform design trade offs. This paper describes an availability model for a high availability platform using three-level hierarchical decomposition that mixes reliability block diagrams and Markov chains. The model is built and evaluated using the SHARPE software package. Sensitivity analysis is performed to identify the effects of critical parameters.*

**Keywords:** *Telecommunications, Modeling, High Availability, Reliability Block Diagram, Markov Chain*

## 1 Introduction

The need for high availability (HA) in telecommunications is more stringent than most other sectors of industry. The Carrier Grade High Availability platform is for “five-nines and better” availability and code portability/re-usability, with no scheduled downtime for either software (SW) or hardware (HW) upgrades. System architecture plays an important role in determining the system’s availability. In this paper we describe a version of the system from the early conceptual design phase. Although the system about to be described does not exist in the field in its entirety, various aspects of the system can be found in products currently operating at customer sites. Some of the

other features mentioned are part of active research and development and might enter into future product lines.

The typical approach to model availability of systems is to use Markov chains [4, 9]. In the early design stage only gross numbers are required, and so analysts resort to a lot of shortcuts and assumptions to reduce the complexity of the model. Often, factors such as coverage probabilities, latent failures, and increased failure rates in degraded states are neglected. Recovery scenarios are most often highly simplified since there is tremendous variety in recovery actions based on the failure mechanism. Though there may be focus on failure scenarios, analysts often ignore the varied recovery scenarios in favor of something simplistic like a reboot or a replacement. Neglecting either failures of components during recovery or probabilities of unsuccessful recovery is not uncommon in the industry.

If detailed failure/recovery behavior is to be considered then a single monolithic Markov model for the entire system will become prohibitively large. In fact, for a large system with many components, a reliability block diagram or a fault tree is often used instead of a Markov chain in order to avoid a large state space. Reliability block diagram models can be solved very easily assuming independence but they are very restricted in the types of failure, recovery and repair scenarios. For a high availability platform, one has to deal with accuracies beyond the norm. Every second of downtime has to be captured explicitly to get a handle on the availability beyond five decimal places. This requirement leads to more extensive models like detailed Markov chains. When there are numerous components, Markov chains become unwieldy to construct, and simpler representation schemes are required. This paper advocates a multi-level hierarchical composition approach with

block diagrams at the top level and Markov chains with detailed failure/recovery scenarios in the lower levels. Thus, we attempt to model the system to the greatest level of detail while balancing complexity of model building. The SHARPE modeling tool is quite adept at building hierarchical models that seamlessly combine multiple modeling frameworks such as block diagrams and Markov chains in a graphical form. The models described in this paper relate to a single drawer and do not encompass models for a clustered configuration, although the features of the cluster are described in the following sections.

In our approach, dependence within subsystems is captured by Markov models. The primary dependence across subsystems is the repair dependence. However, the effect of such repair dependence is minimal allowing independence to be reasonably assumed. The repair dependence across subsystems could be captured by first constructing an overall stochastic Petri net model [1] and then, automatically converting and solving the underlying Markov model with packages such as SHARPE [3] and SPNP [2]. Alternatively, fixed-point iteration methods [8] can be used to approximate the dependence.

### 1.1 Hardware architecture

The hardware configuration includes multiple independent drawers interconnected by dual-redundant SCSI chains or Ethernet. Each drawer has

- A compact PCI backplane
- Redundant power supplies and fans
- An alarm card or system service processor that manages power to all slots as well as power supplies, fans, environmental alarms, and provides interfaces to internal satellite CPU management
- Dual SCSI chains for internal and external disks
- Ultra-Sparc based CPU
- Several hot swappable devices and line cards.

For a host system without clustering, nodes within a single drawer would communicate across the compact PCI backplane. For systems with clustering, communications between nodes on different drawers would be over SCSI or Ethernet. CPUs are in active redundant configuration with a CPU in one drawer being the current cluster master while a CPU in a neighboring drawer is the vice-master, prepared to take over in case the master fails. The line cards are 2N redundant with active and stand-by cards in different drawers.

Such a system can survive the loss of any card with no loss of service and the loss of an entire drawer with only a partial loss of capacity.

### 1.2 Software architecture

The Carrier-Grade HA Software has several elements:

- Component integration services. A “component” in the software architecture is a collection of functionality that can be brought in and out of service, and managed as a unit. A highly available system is comprised of a collection of components that can be independently delivered, replaced, and managed.
- Availability management services. The availability management services are responsible for choreographing the assignment of available components to active and stand-by roles, and for promptly initiating recovery actions in response to failure reports. The key elements of the availability management framework are: a Component Role and Instance Manager to implement all of the availability management policies, a fault-detection model involving both in-line error detection and external audits, and an error correlation model for isolating errors inferable only by correlating multiple reports.
- Distributed system services. Distributed system services enable applications to be spread throughout a cluster of computers, and exchange services without regard for their physical location, network topology, and isolated link failures.
- Underlying Operating system. Much of the foundation for the Carrier-Grade HA Software Platform has been laid by ongoing RAS (Reliability, Availability, Serviceability) enhancements to these flagship products: remote management and debugging tools, general hot-swap support, general purpose logging facilities, process resource monitoring, and robust resource management and panic elimination.

### 1.3 High availability features

High availability features include:

1. A “Highly Available system” is a combination of hardware and software components that jointly provide a service whose availability is much greater than the availability of any individual component.

2. The most important components in the system are the “CPU nodes”, each of which is running an instance of the Solaris operating system. Individual nodes can be rebooted with no effect on other nodes.
3. Hardware components are distributed across multiple independent drawers and buses so the failure of a single fan, power-supply, CPU, and Ethernet cannot take out the entire cluster.
4. The individual nodes are all interconnected to form a “Highly Available Service Cluster”. The nodes cooperate to jointly provide the required highly available service. If one of these nodes that is a member of the cluster fails or has to be serviced, another node will assume this work, and the cluster will continue to provide its service.
5. OS kernel and middle-ware instrumentation continuously check for failure in system services.
6. A health monitoring framework exists to assist with the detection of errors in both system and application services. Watchdog Timers and Cluster Membership monitoring provide ultimate oversight of the operation of each node in the cluster.
7. Internal error detection and analysis within each individual component is the fastest and most reliable means for isolating the cause of an error.
8. An availability management framework assigns available components to act as stand-by for active components, and introduces the actives and stand-by to one another.

## 2 System model

### 2.1 Availability models

A three level hierarchical composition is adopted to analyze the system availability. Top level is the reliability block diagram (RBD) in which each block either corresponds to a sub-model of a subsystem or is an individual component with a specified failure rate and repair rate. Second level in the hierarchy is the Markov chains except for the satellite card subsystem. In the last level of the hierarchy the satellite card subsystem described by a reliability block diagram contains two Markov chain models, SatNet and SatCard. With the help of the software package, SHARPE [6], the model can be specified easily and solved efficiently. We present models drawn in SHARPE GUI [3].

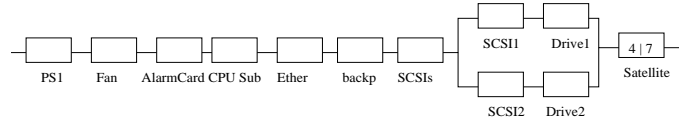


Figure 1. Top level block diagram

From an availability point of view, our top level system is essentially a reliability block diagram consisting of series, parallel, and k-out-of-n subsystems as shown in Figure 1. The configuration that is considered for analyzing the availability of the system is shown in Table 1. Power system, fan, Ethernet, PCI backplane, SCSI and Drive modules can fail due to hardware faults. Remaining modules can fail either because of software or hardware faults. To ensure system availability, it is required that at least four out of seven satellite card modules are functional.

Module	Redundancy	Outages
Power System	2	HW
Fan	2	HW
Alarm card	1	HW, SW
CPU	1	HW, SW
Ethernet	2	HW
Backplane	1	HW
SCSI	2	HW
Drive	2	HW
Satellite Card	4:7	HW, SW

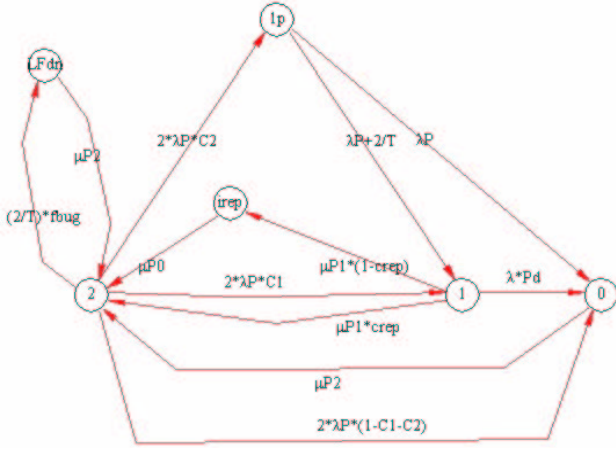
Table 1. Platform availability without clustering

### 2.2 Subsystem models

In this section, we construct the Markov chain models capturing the dynamic behavior of each subsystem. In the last part of this section, we construct Markov chains for each of the two blocks in series of SatNet subsystem.

#### 2.2.1 Power supply subsystem

The power supply (PS) subsystem is modeled by means of a six-state continuous-time Markov chain as shown in Figure 2. In state 2 the two units of subsystem are up. If the subsystem encounters a failure, then it will move from state 2 to state 1 or state 1P depending on whether the fault is covered or latent respectively. Another failure from state 1 or state 1P will move the system to state 0, which indicates that the power supply subsystem is down. When the failed unit is repaired

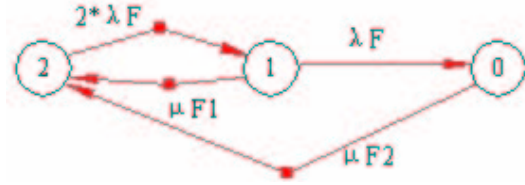


**Figure 2. Markov chain for the power supply subsystem**

successfully in state 1, the subsystem enters state 2. If the repair is unsuccessful, the subsystem enters state irep. At the end of the repair transition from irep, subsystem enters state 2. State LFdn is used to allow for diagnostics to bring the subsystem down. State 1p is used to indicate that a latent-fault has occurred, which will eventually lead to failure of the power supply subsystem. Transition from state 1p to state 1 can occur if a non-latent fault occurred in the subsystem with a latent fault or if diagnostics detected a latent fault. Failure and repair rates of power supply subsystem are given in Table 2. The power system is up as long as it is in states 2, 1p and 1. The reason for the term  $2/T$  in the transition rates out of state 2 is that on the average the time to occurrence of diagnostics is half the diagnostics period.

Symbol	Meaning
$\lambda P$	Failure rate
C1	Coverage for first fault
C2	Probability of latent fault
T	Diagnostics period
fbug	Probability of diagnostics bringing system down
$\mu P0$	Repair rate when person on site
$\mu P1$	Repair rate for one unit
$\mu P2$	Repair rate for two units
$\lambda P$	PS Failure rate
$\lambda Pd$	PS Failure rate in degraded state
crep	Probability of successful repair

**Table 2. Failure and repair rates of power system submodel**



**Figure 3. Markov chain for the fan subsystem**

### 2.2.2 Fan subsystem

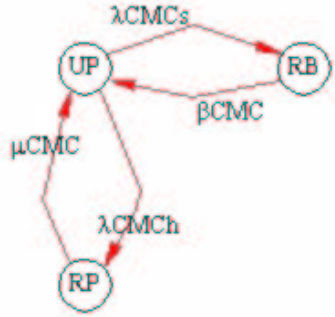
The fan subsystem has one type of failure and two kinds of recovery. A continuous-time Markov chain for fan subsystem is shown in Figure 3. In state 2, the two units of subsystem are up. If either of them suffers a failure with a total failure rate of  $2*\lambda F$ , then the subsystem enters state 1. When the failed unit is repaired in state 1 with a repair rate of  $\mu F1$ , the subsystem enters state 2. The second unit can fail in state 1 with a failure rate of  $\lambda F$ , before the repair of the first failed unit. This case leads to a transition from state 1 to state 0. From state 0, both the units of the fan subsystem can be repaired with a repair rate of  $\mu F2$ . The fan subsystem is up as long as at least one of the units is up. The subsystem is down when it is in state 0 of the Markov chain.

### 2.2.3 Alarm card subsystem

The alarm card subsystem is modeled by a three-state continuous-time Markov chain as shown in Figure 4. The alarm card subsystem has two kinds of failure/recovery, including both hardware and software failure-repairs. The system starts in state UP. The subsystem enters state RP with a hardware failure rate of  $\lambda CMCh$ , and can return to state UP with a hardware repair rate of  $\mu CMC$ . From state UP, the subsystem enters state RB with a software failure rate of  $\lambda CMCs$ , and returns to state UP with a reboot rate of  $\beta CMC$ . The subsystem is considered up, when it is in state UP.

### 2.2.4 CPU card subsystem

A non-redundant module with multiple types of failure/recovery, is modeled by a five-state continuous-time Markov chain as shown in Figure 5. It can experience failures due to hardware or software. The failure types are distinguished by type of recovery action required. A cold reboot may be required to deal with hardware transients and some portion of OS failures. A panic reboot is software reboot where the OS kernel or application decides to reboot because of a se-



**Figure 4. Markov chain for the alarm card subsystem**

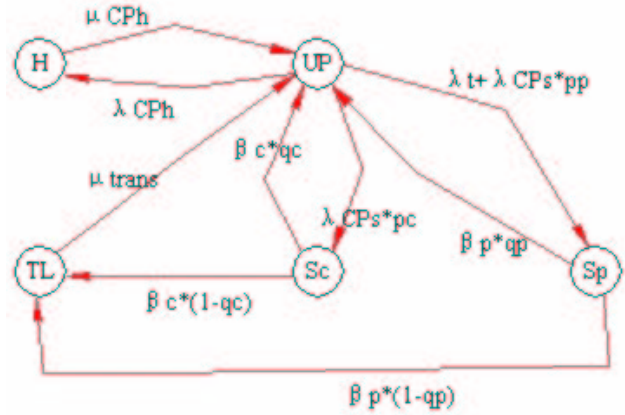
vere situation to prevent further errors, which could get more serious. The subsystem starts in state UP. With a hardware failure, the system enters state H and returns with a repair to state UP. State Sp represents panic reboot state of the subsystem. Transition from UP to Sp is triggered either by a transient hardware fault or by a fault in software, which may lead to the panic reboot with a probability of pp. System enters state Sc from state UP, when a cold reboot is required. Reboots can be successful, leading the system to return to state UP. Reboots of the system may not be successful, triggering a need for the database rebuild. Transition rates and the Markov chain are given in Table 3, and Figure 5 respectively. CPU card subsystem is up, as long as the Markov chain is in state UP.

Symbol	Meaning
$\lambda CPh$	Hardware failure rate
$\mu CPh$	Hardware repair rate
$\lambda CPs$	Software failure rate
$\lambda t$	Transient failure rate
$\mu trans$	Database rebuild rate
$\beta c$	Cold reboot rate
$\beta p$	Panic reboot rate
qp	Probability of successful panic reboot
qc	Prob. of successful cold reboot
pp	Prob. of occurrence of panic reboot
pc	Prob. of occurrence of cold reboot

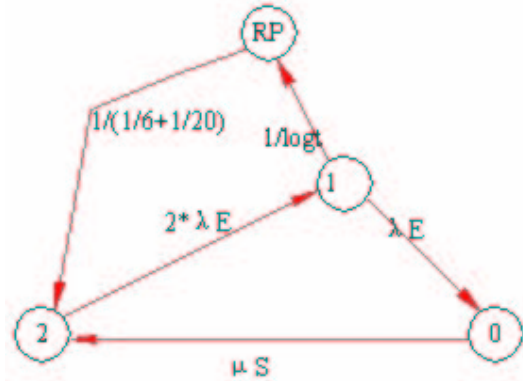
**Table 3. Failure and recovery rates of CPU card subsystem submodel**

### 2.2.5 Ethernet subsystem

The Ethernet subsystem is modeled by a four-state continuous-time Markov chain as shown in Figure 6.



**Figure 5. Markov chain for the CPU card subsystem**



**Figure 6. Markov chain for the Ethernet subsystem**

This subsystem is modeled with one type of failure and two types of recovery. The system starts initially in state 2 indicating that both ports of the subsystem are up. With a failure of one of the ports with a total rate of  $2*\lambda E$ , the subsystem enters state 1. The second port can fail in state 1 with the rate  $\lambda E$ , before the repair of the first failed port, leading the subsystem to enter state 0. The subsystem may undergo repair of the failed port which proceeds in two stages. First is the logistic time  $\log t$ . This explain the transition from state 1 to state RP. State RP represents the subsystem under repair. The Ethernet subsystem will enter state 2 from state 0 through the repair of the two failed ports with the rate  $\mu S$ . The Ethernet subsystem is considered up, as long as it is in states 2 or 1.

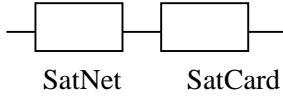


Figure 7. Block diagram for satellite model

### 2.2.6 Backplane, SCSI and drive subsystems

The PCI backplane block is a non-redundant module with only one kind of failure/recovery, which is modeled as a single component in the block diagram by simply specifying its failure rate and repair rate.

The SCSI block is split into two parts. That part of SCSI that brings both disks down is put in series while the part of SCSIs that only brings its own disk down is put in parallel. Each of the individual SCSI block is a non-redundant module with only one kind of failure/recovery, which is modeled as a single component in the block diagram by specifying its failure rate and repair rate.

The drive block is also a non-redundant module with only one kind of failure/recovery. It is also modeled as a single component in the block diagram by simply specifying its failure rate and repair rate.

Symbols used for representing failure and repair rates of these blocks are given in Table 4.

Symbol	Meaning
$\lambda_{BP}$	Backplane failure rate
$\mu_{BP}$	Backplane repair rate
$\lambda_{SC}$	SCSI failure rate
$\mu_{SC}$	SCSI repair rate
$\lambda_{DD}$	Drive failure rate
$\mu_{DD}$	Drive repair rate

Table 4. Failure and recovery rates of backplane, SCSI and drive blocks

### 2.2.7 Satellite card subsystem

A 4-outof-7 satellite block in the Figure 1 is modeled by a next level reliability block diagram shown in Figure 7. Each of the SatNet and SatCard blocks in Figure 7 are modeled using continuous-time Markov chains. SatNet block is modeled using the Markov chain for Ethernet subsystem discussed earlier. SatCard is a non-redundant module modeled by a five-state continuous-time Markov chain as shown in Figure 8. This model of a satellite card is very similar to that of the CPU card, except for the difference in some of the transition rates.

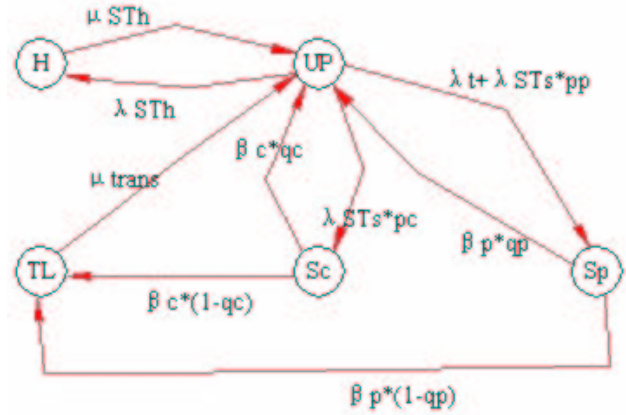


Figure 8. Markov chain for the SatCard sub-model

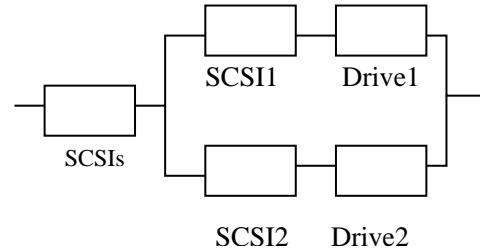


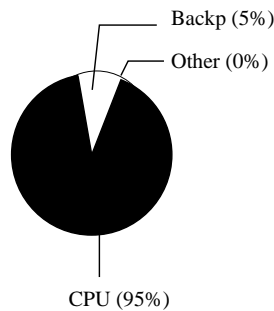
Figure 9. Disksub model

## 3 Results

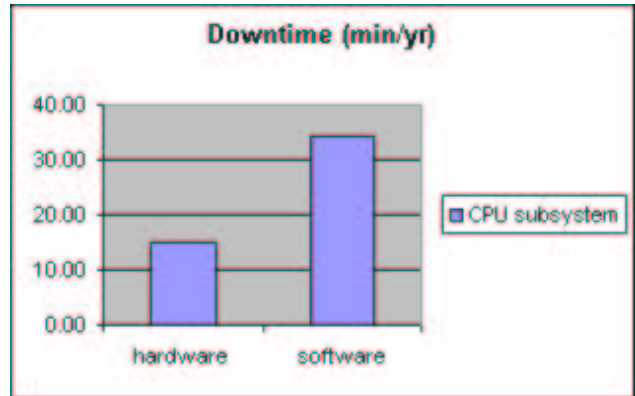
Table 6 lists the parameters that we considered for solving the hierarchical models [6]. These parameters are preliminary estimates based on experience and expert opinion. For the purpose of downtime decomposition, another model shown in Figure 9 is created called Disksub. Results shown in Table 5 were obtained by solving the subsystem models. Based on the downtimes of the subsystems in Table 5, a downtime pie chart is shown in Figure 10. Downtime breakdown for the CPU subsystem is given in Figure 11. With the parameters in Table 6, the CPU subsystem is responsible for 95 percent of the total downtime of the overall system. The possibilities to increase the availability of this subsystem include improving parameters like:  $\lambda_{CP}$ s (software failure rate),  $\beta_p$  (panic reboot rate),  $qp$  (probability of successful panic reboot) or  $qc$  (probability of successful cold reboot). The importance of these parameters in the final downtime result of CPU subsystem can be seen through the plots given in Figures 12, 13 and 14. Availability can also be increased by providing redundancy in the CPU subsystem.

System	Availability	Unavailability	Downtime(min/yr)
PS subsystem	9.9999840e-01	1.60e-07	0.0841
FAN subsystem	0.999999999955	4.48e-12	0.00000236
Alarm card subsystem	9.9999970e-01	3.00e-08	0.0158
CPU subsystem	9.99906213e-01	9.38e-05	49.3
ET subsystem	9.9999922e-01	1.10e-12	0.000000576
Disksub subsystem	9.9999967e-01	3.35e-08	0.0176
Backplane block	9.9995053e-01	4.95e-06	2.6
Satellite subsystem (4 out of 7)	1	0	0
System as a whole	9.99900966e-01	9.90e-05	52.1

**Table 5. Availability, unavailability and downtime**



**Figure 10. Downtime pie chart**



**Figure 11. CPU subsystem downtime break-down**

## 4 Conclusion

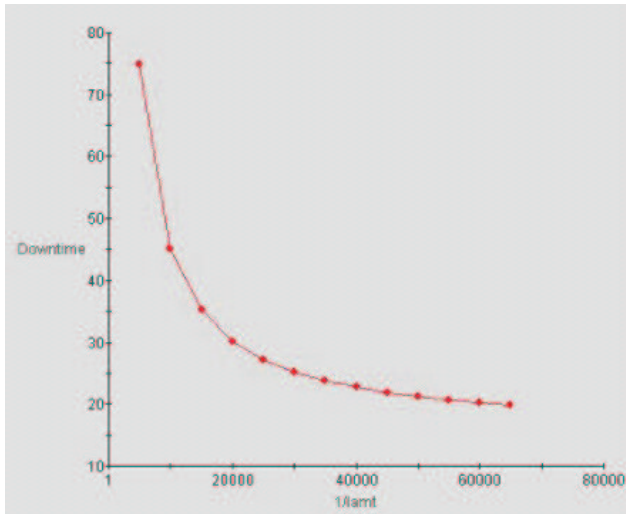
High availability platforms need to have availability and fault tolerance built into the system right from the start of the product life-cycle. This requires models that can capture every possible second of downtime (since there are not many to begin with). Gross models such as simple block diagrams or Markov chains which neglect states and transitions thought to have little influence will not suffice. A multi-level hierarchical composition approach is advocated that judiciously combines reliability block diagrams and Markov chains so as to allow detailed behavior to be captured while avoiding state space explosion.

## 5 Acknowledgements

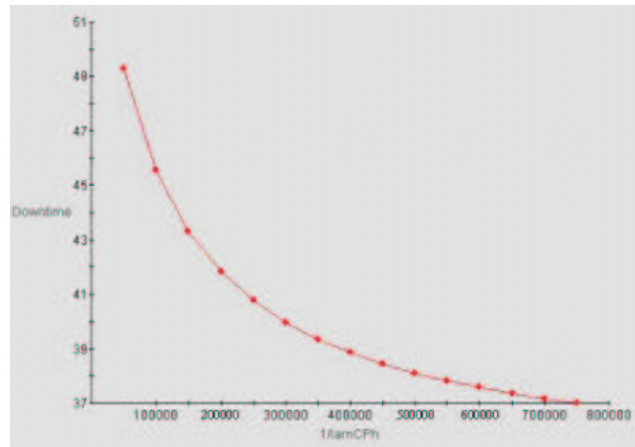
The authors wish to express their sincere thanks to Sanghamitra Sinha, Sanjay Agrawal, Swee Lim and Frederic Herrmann for their valuable technical suggestions and comments.

## References

- [1] G. Ciardo, A. Blakemore, P. F. Chimento, J. Muppala, and K. S. Trivedi, "Automated Generation and Analysis of Markov Reward Models Using Stochastic Reward Nets", *Linear Algebra, Markov Chains, and Queueing Models, IMA Volumes in Mathematics and its Applications*, C. Meyer and R. J. Plemmons (eds.), Vol. 48, pp. 145-191, Springer-Verlag, Heidelberg, 1993.
- [2] G. Ciardo, K. S. Trivedi, and J. Muppala, "SPNP: stochastic Petri net package", *In Proceedings of the Third International Workshop on Petri Nets and Performance Models (PNPM'89)*, pages 142-151, Kyoto, Japan, 1989, IEEE Computer Society Press.
- [3] C. Hirel, R. Sahner, X. Zang and K. S. Trivedi, "Reliability and Performability Modeling using SHARPE 2000", *International Computer Performance and Dependability Symposium (IPDS)*, Schamburg, Chicago, 2000.

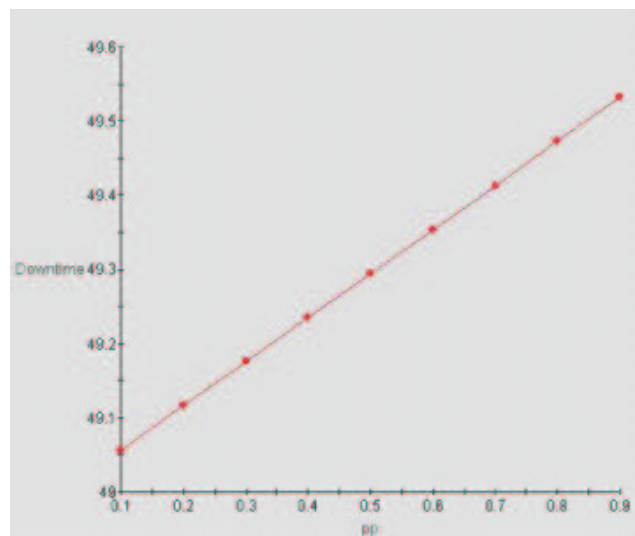


**Figure 12. CPU subsystem downtime vs  $1/\lambda t$  (transient MTTF)**



**Figure 13. CPU subsystem downtime vs  $1/\lambda CPh$  (hardware MTTF)**

- [4] D. R. Jeske and X. Zhang, "Some Successful Approaches to Software Reliability Modeling in Industry," *Journal of Systems and Software*, Vol. 74, pp. 85-99, 2006.
- [5] M. Kampe, "Sun Carrier-Grade High-Availability platform TM Architecture", White Paper, Sun Microsystems, 1999.
- [6] R. A. Sahner, K. S. Trivedi, and A. Puliafito, *Performance and reliability analysis of computer systems, An example-based approach using the SHARPE software package*, Kluwer Academic Publishers, 1996.
- [7] P. A. Tobias and D. C. Trindade, *Applied Reliability*, third edition, Chapman-Hall/CRC Press, 2007.
- [8] L. A. Tomek and K. S. Trivedi, "Fixed Point Iteration in Availability Modeling", *Proceedings of the 5th International GI/ITG/GMA Conference on Fault-Tolerant Computing Systems, Tests, Diagnosis, Fault Treatment*, 1991.
- [9] K. S. Trivedi, *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*, second edition, John Wiley, 2001.



**Figure 14. CPU subsystem downtime vs pp (panic reboot)**

Symbol	Meaning	Value	Unit
$\lambda F$	Fan failure rate	1/2864907	per hour
$\lambda P$	PS failure rate	1/245930	per hour
$\lambda P_d$	PS failure rate in degraded state	$2 * \lambda P$	per hour
$\lambda CPh$	Hardware failure rate for CPU	0.93/139282	per hour
$\lambda CP_s$	Software failure rate for CPU	1/500000	per hour
$\lambda_t$	Logistic time rate	1/8760	per hour
$\lambda E$	Ethernet failure rate	$0.025 * \lambda CPh / 0.93$	per hour
$\lambda SC$	SCSI failure rate	$0.01 * \lambda CPh$	per hour
$\lambda DD$	Drive failure rate	1/1000000	per hour
$\lambda CMC_s$	Software failure rate for CMC	0.01/100000	per hour
$\lambda CMCh$	Hardware failure rate for CMC	0.01/1396122	per hour
$\lambda STh$	Hardware failure rate for SatSub	0.95/143160	per hour
$\lambda ST_s$	Software failure rate for SatSub	1/50000	per hour
$\lambda BP$	Backup failure rate	1.12/1000000	per hour
C1	coverage for first fault for PS	0.998	
C2	Prob. of latent fault for PS	0.001	
T	Diagnostics period	10000	per hour
fbug	Prob. diagnostics brings system down	0.0001	
$\mu F1$	Repair rate of 1 fan	$1 / (\log t + 1/6)$	per hour
$\mu F2$	Repair rate of 2 fans	$1 / (\log t + 2/6 + 1/12)$	per hour
$\mu P0$	Repair rate when one person on site	$1 / (1/3 + 1/12)$	per hour
$\mu P1$	Repair rate for one unit for PS	$1 / (\log t + 1/6)$	per hour
$\mu P2$	Repair rate for two units for PS	$1 / (\log t + 1/3 + 1/12)$	per hour
$\mu S$	Repair rate for 2 ports (Ethernet)	$1 / (\log t + 1/6 + 1/12)$	per hour
$\mu CMC$	Repair rate of CMC	$1 / (\log t + 1/6)$	per hour
$\beta CMC$	Reboot rate of CMC	3600/5	per hour
$\mu STh$	Hardware repair rate for SatSub	$1 / (\log t + 1/6 + 1/20)$	per hour
$\mu SC_s$	SCSI repair rate	$1 / (1/6 + 1/12)$	per hour
$\mu CPh$	Hardware repair rate for CPU	$1 / (\log t + 1/6 + 1/\beta c)$	per hour
$\mu BP$	Backup repair rate	$1 / (\log t + 1/3 + 1/12)$	per hour
$\mu DD$	Drive repair rate	$1 / (\log t + 1/6 + 4.5)$	per hour
$\mu trans$	Database rebuild rate	$1 / (\log t + 4)$	per hour
crep	Prob. of successful repair for PS	0.99	
pp	Prob. of occurrence panic reboot	0.5	
pc	Prob. of occurrence cold reboot	0.5	
qp	Prob. of successful panic reboot	0.95	
qc	Prob. of successful cold reboot	0.99	
$\beta p$	Panic reboot rate	12	per hour
$\beta c$	Cold reboot rate	6	per hour
logt		4	hours

**Table 6. Parameters of the models**