Dependability evaluation of a distributed shared memory multiprocessor system

Mourad Rabah, Karama Kanoun

DEPENDABILITY EVALUATION OF A DISTRIBUTED SHARED MEMORY MULTIPROCESSOR SYSTEM

M. RABAH, K. KANOUN

LAAS Report N° 99100

March 1999

Dependability Evaluation of a Distributed Shared Memory Multiprocessor System

Mourad Rabah and Karama Kanoun
LAAS-CNRS — 7 Avenue du Colonel Roche
31077 Toulouse Cedex 4 — France
e-mail: rabah, kanoun@laas.fr

Abstract

This paper deals with the dependability evaluation of a Multipurpose, Multiprocessor System under investigation by a system manufacturer. MMS is a symmetric multiprocessor system with a Distributed Shared Memory and a Cache-Coherent Non-Uniform Memory Access. As the system is scalable, we consider two architectures: a reference one composed of four nodes and an extended one with an additional spare node. A set of dependability and performability measures is evaluated for these architectures with two categories of application users, accepting different service degradation levels. Modeling is based on Generalized Stochastic Petri Nets to which reward rates are added to evaluate performability measures. The originality of our approach is the clear separation between the architectural and the service concerns. The results allow the quantification of the influence of the main parameters and comparison of the dependability of the systems under consideration. They can thus be used for supporting the manufacturer design choices as well as the potential user configuration choices.

Index Terms:

Dependability evaluation, multiprocessor systems, graceful degradation of service, Generalized Stochastic Petri Nets

Approximate word count: 7500 words including tables and references + less than one page figures.

1. Introduction

Nowadays, computer systems are becoming more and more complex. They are composed of hardware and software components interacting together to ensure the required service. Even though specific applications may need the design of specific systems to achieve the required functions, economic reasons lead to use basic support systems (computers, operating systems, middleware) available on the market. This situation gave rise to a category of systems that can be referred to as multipurpose systems that are — to some extent — application-independent basic systems. Many system providers have developed generic systems that can be used in several application domains. Such systems can be either complete support systems (such as: LOGISIRE, developed by ABG, Germany [Kloppenburg 1987], the TRICON system developed by TRICONEX, USA [Triconex 1996], Alpsa-8000-P320 developed by CEGELEC, France [Poueyo and Dalzon 1998]), or generic components that can be integrated into complete systems (such as the Votrics system for hardware fault tolerance developed by ALCATEL Austria [Wirthumer 1989]). Even though most of the time, basic support systems are composed of Commercial Off-The-Shelf components (COTS), careful design, development and validation are needed to provide dependable and high-performance support systems. The evaluation of performability measures constitutes a powerful means for assessing the dependability and performance of such systems.

The Multipurpose, Multiprocessor System, MMS considered in this paper is under investigation by a system manufacturer. MMS is scalable and most of its components are COTS. It uses a Distributed Shared Memory (DSM) and a Cache-Coherent Non-Uniform Memory Access. A recent tutorial on "Distributed Shared Memory Concepts and Systems" [Protic, Tomasevic et al. 1998] showed that 1) building of commercial systems that follow the DSM paradigms is still in its infancy despite the amount of research work performed in the domain and 2) most of the existing multiprocessor systems based on DSM are research prototypes. Hence the importance of the commercialization of such systems.

A reference MMS architecture is defined. It is composed of four nodes. It can be used for various applications requiring different performance and/or dependability levels. Based on this reference architecture, a whole family can be defined for applications necessitating either higher performance or higher dependability levels or both. Our aim is to provide a framework for the dependability evaluation of the reference architecture with different application needs and more generally to define a framework (for the system manufacturer) for modeling efficiently new systems of the same family based on this reference architecture.

In this paper, we define a set of dependability and performability measures that are evaluated for the reference architecture and an extended one (using a spare node) with two examples of application users requiring different service degradation levels. The results allow the quantification of the influence of the main parameters on the defined measures and the comparison of the considered systems.

The paper is organized as follows. Section 2 presents the systems under consideration. Section 3 defines the performability measures. Section 4 describes the modeling approach and illustrates it through an example, while Section 5 presents some results. Section 6 concludes the paper.

2. Presentation of the systems under consideration

We first present the reference architecture with its associated failure modes. Two types of users are then defined together with their associated service levels sand maintenance policies. Finally an extended architecture is defined.

2.1 The reference architecture

The reference architecture is composed of 16 processors grouped into 4 identical nodes as shown in Figure 1. The nodes are connected through an interconnection network composed of two redundant rings. Each ring is connected to each node via a controller. To ensure high availability level, all nodes are connected to a centralized diagnostic equipment (Deq) that is redundant. Its role is to log and analyze all error events in the system, initiate the reboots and control them. The DEq does not contribute to service accomplishment: the system can work without it but cannot reboot without it.

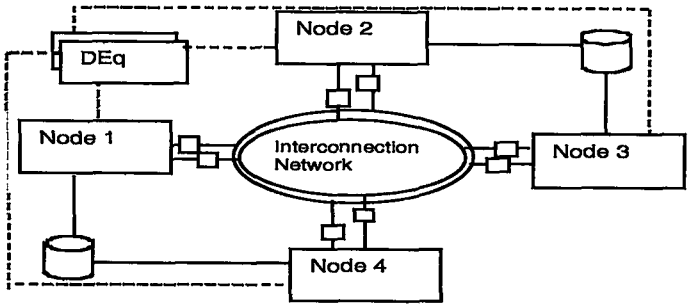


Figure 1: MMS reference architecture

MMS has two external disks. Each disk is a dual access RAID (Redundant Array of Independent Disks). The disks of the RAID can be stopped, changed and synchronized without stopping the RAID system (on-line maintenance). Each RAID is shared by two nodes. In case of failure of a node or of the interconnection controller in a node, the RAID can be used by the other nodes. In case of a failure of both RAIDs, no external storage is available.

As indicated in Figure 2, each node has:

- a set of four independent processors; each processor has a two-level private cache where data from other node memories can be replicated or migrated;
- a local bus;
- a set of dedicated controllers such as interruption and reboot controllers;
- a local Random Access Memory, RAM (composed of four banks) and its controller; a node is available as long as a memory bank is available and a node can use any other node memory bank as the RAMs of the four nodes constitute the distributed shared memory of MMS;
- a Remote Cache Controller (RCC), interconnecting the node to the rings via two controllers;
- a set of Peripheral Component Interconnect (PCI) buses, with slots for connecting external devices (through Ethernet, FDDI connections, etc.), connected to the local bus by a bridge;
- an internal disk, connected through an interface to the PCI bus;
- miscellaneous devices, such as power supply units and fans.

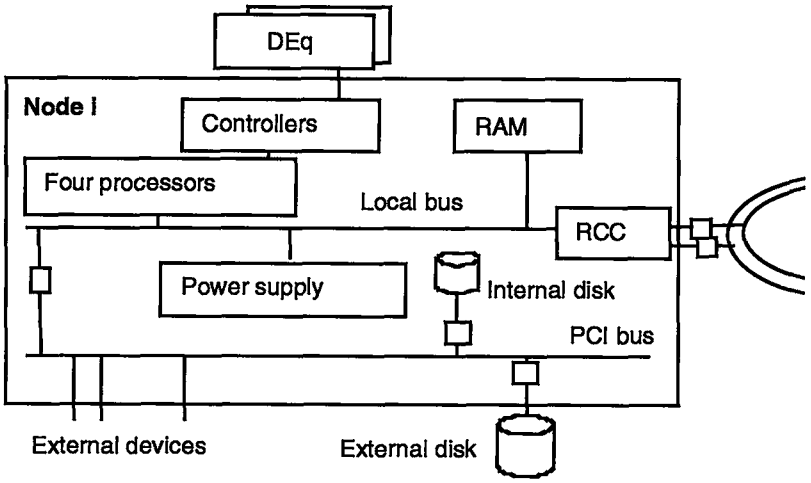


Figure 2: Main components of a node

In addition to the redundant diagnostic equipment, DEq, and to the redundancy of several components (interconnection network, power supply, system disk, swap and external disks), MMS includes other facilities to reduce system downtime following error detection, among which:

- a node is held available as long as at least one processor, one memory bank, one power supply unit, all fans, the local bus and PCI buses with their connectors and controllers are available;
- the memories feature single error correction and double error detection codes;
- the internal disks can be hot plugged;
- a software failure of a processor leads to system reboot;
- the architecture can be automatically reconfigured: the load of a failed processor can be partitioned among the other processors and the load of a whole node can be partitioned among the other nodes (the node is isolated). A system reconfiguration necessitates a reboot.

- the components of an isolated node are not accessible by the other nodes;
- the automatic system (re)configuration can be performed when the system is in operation or during reboot (if a failure is identified by on-line tests during system reboot);
- the automatic reboot can be complete with all on-line tests or fast (with a reduced test set);
- self-detected errors of the Operating System (OS) lead to system automatic reboot, while non-self-detected errors of the OS require a manual reboot;
- while detected errors of a component necessitating system reboot lead to a proper reboot, non-detected errors lead a brutal reboot (without saving properly the main parameters for example).

It is intended that some failures (whose nature is to be determined by the system manufacturer according to some criteria) will lead to system reconfiguration without reboot or with fast reboot. However, we assume in this paper that any component failure necessitating system reconfiguration forces a complete reboot. This assumption can be easily modified for those failures needing system reconfiguration without reboot or with a fast reboot.

2.2 Architectural failure modes

Given the large number of components (processors, memory banks, disks, busses, interconnections, controllers, interfaces, etc.) and the specific influence of their failure on system state and performance, a multitude of failure modes can be defined, related to the nature of the failed components. Generally, they are identified by a Preliminary Risk (or Hazard) Analysis: the consequences of all component failures are analyzed, individually and in combination. Usually states with equivalent consequences are grouped into the same failure mode.

For the purpose of the study, from the Preliminary Risk Analysis, seven architectural failure modes have been defined, including five levels of performance degradation. The table giving the detailed correspondence between elementary component failures and the architecture failure modes spreads in more than four pages for the reference MMS. The identified failure modes are:

- B: loss of one or more components among the redundant components; the processing capacity of the architecture is not impaired. B is referred to as the benign failure mode.
- D0: loss of one processor or one memory bank or connections to external devices or loss of the diagnostic equipment, DEq.
- DD: loss of the two external disks.
- Di: loss of i nodes[1] $i = \{1, 2, 3\}$.

---

[1] Let's recall that a node is available as long as at least one processor, one memory block out of four, one power supply unit, all fans, the local bus and PCI buses with their connectors or controllers are available.

- C: loss of the four nodes, or loss of a critical component: interconnection network, all power supply units, and non-self-detected errors of the Operating System.

In addition, two specific states are defined:

- OK: the state without any component failure.
- Reb: the reboot state.

Note that states with components in various failure modes are considered in the most severe failure mode. For example, when a node is in failure mode D1 and another is in D0 or B, the state is put in the failure mode D1.

2.3 Service degradation levels

The above MMS properties and the architecture failure modes are user-independent. However, even though graceful degradation of the architecture is provided by the supporting architecture, this possibility can be exploited differently by the users for different application requirements. For instance, some applications will accept a degraded service provided by three, two or even only one node, while others will only accept the loss of some processors, without loosing a whole node. Indeed, the architectural failure modes lead to service degradation levels that are strongly related to the user's specific application needs.

It is worth noting that B and C have the same meaning for all applications: in B the entire service is performed (**ES**) while in C there is no service (**NS**). Between B and C, several service degradation levels can be defined according to user's needs and to the architecture failure modes.

To illustrate the mapping between architectural failure modes and service degradation levels, we consider two examples of users, denoted X and Y.

- **User X**: The service is considered as entire in states OK and B. X accepts a degraded service as long as at least three nodes are available without a critical component failure: NS correspond to D2, D3 and C. Between ES and NS, we have defined two significant service degradation levels: minor and major degradation levels. The mapping between the architectural failure modes and the service degradation levels is given in Table 1.
- **User Y** needs the entire capacity of the system and does not accept any service degradation. The entire service is delivered in OK, B and DD. NS gathers all other failure modes.

The states belonging to the same service degradation level are grouped into a class state. These classes are denoted respectively: ES-S, mD-S, MD-S and NS-S.

Service levels	Entire service (ES)	Minor degradation (mD)	Major degradation (MD)	No Service (NS)	Reboot
X	OK, B	D0, DD	D1	D2, D3, C	Reb
Y	OK, B, DD	—	—	D0, D1, D2, D3, C	Reb

Table 1: Mapping between architectural failure modes and service degradation levels

User X corresponds most likely to a potential user of MMS as he takes advantage of the architecture reconfigurations offered by the manufacturer. User Y is a special case, it has been defined to allow comparison between the two types of users without considering complex situations. Note that state group DD is in ES for Y while it is in mD for X.

2.4 Maintenance

Although maintenance management is user-dependent, the system supplier usually proposes a maintenance contract. In order not to suspend system activities after each failure and call for a paying maintenance, two maintenance policies are defined:

- **Immediate maintenance**: for NS-S (as defined by the user's needs). However some time is needed for maintenance arrival. Immediate maintenance is thus performed in two steps: maintenance arrival (during this time, the system is unavailable) and repair.
- **Delayed maintenance**: for the other states with service degradation: the system continues to be used until a period of time in which its activities can be stopped for repair. Different delays may be allocated to the different service levels. Delayed maintenance is also performed in two steps: a delay during which a degraded service is delivered and repair.

The repair is followed by a system reboot.

2.5 Extended architecture

To illustrate the approach, we also consider an extended architecture that features a fifth node used as a spare. The four basic nodes are used in the same manner as in the reference architecture. In case of failure of a first node, the latter is replaced by the spare node. We have thus four systems: the reference architecture (composed of 4 nodes) combined with users X and Y, and the extended architecture (5 nodes), also with users X and Y. For the four systems, we assume immediate maintenance for NS-S and delayed maintenance for the other states. For the sake of conciseness, these systems will be respectively referred to as X4, Y4, X5 and Y5.

The failure modes and service degradation levels defined for the reference architecture apply to the extended architecture as well, since the spare does not contribute to service accomplishment in the absence of a failure.

3. Dependability and performability measures

Dependability measures are user-dependent: they are defined according to the service degradation levels. For user Y, we have considered a conventional availability measure: the system is available in ES-S states and unavailable in NS-S.

While this measure still holds for X, three additional **dependability levels** are defined corresponding to the three service levels:

- LE(t): dependability level associated with the entire service, ES, at time t.
- Lm(t): dependability level associated with the minor service degradation level, mD.
- LM(t): dependability level associated with the major service degradation level, MD..

Let e(t) denote the system state at time t. According to the partitions defined in Section 2.3, we have:

- LE(t) = Prob { e(t) ∈ ES-S}.
- Lm(t) = Prob { e(t) ∈ mD-S}
- LM(t) = Prob { e(t) ∈ MD-S}

For X, the classical system availability and unavailability are respectively defined by:

A(t) = Prob { e(t) ∈ {ES-S, mD-S, MD-S} } = LE(t) + Lm(t) + LM(t)

UA(t) = Prob { e(t) ∈ {NS-S, Reb} } = 1 – A(t)

An additional measure of interest for X and Y is the system unavailability due to system reboot:

$UA_{Reb}(t)$ = Prob { e(t) ∈ Reb}.

The steady state measures are respectively denoted: LE, Lm, LM, A, UA and UA_{Reb}

LE and A will be expressed in terms of probability. Lm, LM, UA and UA_{Reb} will be expressed as the sojourn times per year in the considered service level.

As the system continues operation even in the presence of component failures, with performance degradation, **performability** measures are of prime interest as indicated in [Meyer 1978; Arlat and Laprie 1983; Meyer and Sanders 1993; Tomek, Mainkar et al. 1994]. Performability measures the performance of the system in presence of failures. In our case, we assume that all states in a given service degradation class have the same reward rate. Let r_Z denote the reward rate of the class state z: r_Z is the performance index in the class state Z (these indexes are

appreciated by the user according to his application). The expected reward rate, W(t), at time t is defined by:

$$E[W(t)] = r_{ES\text{-}S}\, LE(t) + r_{mD\text{-}S}\, Lm(t) + r_{MD\text{-}S}\, LM(t) + r_{UA}\, UA(t).$$

The expected reward rate at steady state is:

$$E[W] = r_{ES\text{-}S}\, LE + r_{mD\text{-}S}\, Lm + r_{MD\text{-}S}\, LM + r_{UA}\, UA.$$

The above equation shows that the dependability measures are special cases of the expected reward rate with specific performance indexes (equal to zero or one).

In the rest of the paper, we will evaluate the steady unavailability, UA, the steady unavailability due to system reboot, UA_{Reb}, the dependability levels LE, Lm and LM and the expected reward rate at steady state, E[W]. When necessary, other measures such as the Mean Down Time, MDT (due to NS-S), or the Mean Time To Failure, MTTF, will be evaluated to put emphasis on specific behaviors.

4. Dependability modeling

As the system is modular, with COTS components on which various specific applications can be run, our modeling approach has the same properties. It is modular, it uses generic sub-models and it is based on the separation of architectural concerns (including all basic hardware and software components) from those related to the user's application (i. e., service concerns). Modularity is useful for mastering complexity. The elaboration of generic sub-models is useful for modeling efficiently similar architectures thanks to the reuse of sub-models. The separation of concerns property allows reuse of the part of the model related to the system architecture, when the basic architecture is to be used with different service requirements.

Model construction is based on Generalized Stochastic Petri Nets (GSPN). The GSPN is then transformed into a Stochastic Reward net [Ciardo, Muppala et al. 1989] by assigning reward rates to tangible markings (i. e., the states of the Markov chain associated with the GSPN). The Markov Reward model [Howard 1971] associated with the Stochastic Reward net is then processed to obtain dependability and performance measures.

The GSPN of the components and their structural interactions are established in a first step. Any structured modeling approach can be used. However, to support the mapping between the service degradation levels and the architectural failure modes, specific places corresponding to the various failure modes are created and put into a layer, identified as the failure mode layer. The latter constitutes the interface between the architectural and the service parts of the model. Once the

needs of a user are identified, the mapping between the service degradation levels and the failure modes can be easily achieved. The maintenance is managed according to the user policy in the service model. The interactions between the different parts of the model are depicted in Figure 3[2].

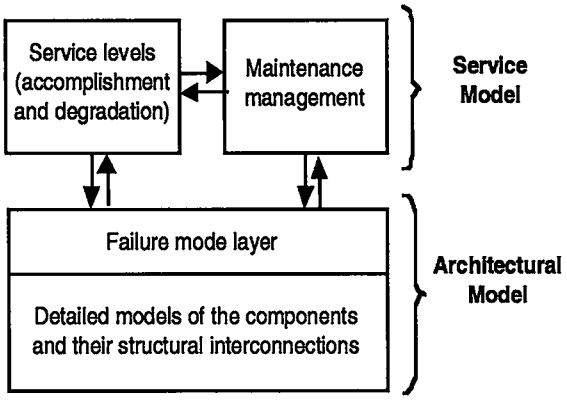


Figure 3: Interactions between the different parts of the model

The architectural model is the same for users X and Y, whereas the service model is different. It can be argued that the model is over-dimensioned for Y as we have considered several failure modes that will be grouped into the same service degradation level. However, as we have built only a small part of the model that is specific to Y, there is no loss of time. The gain is even more substantial for the system manufacturer who will be able to provide to potential system buyers quantified dependability and performability measures, based on the same model (requiring only small adaptations for specific situations).

4.1 System model

In the architectural model, a block model is associated to each component, including controllers, power supply units, fans, busses, diagnostic equipment, etc. When identical components have the same behavior with respect to the evaluated measures, they are modeled by one block. This is for example the case of the four processors and the four memory banks within a node. Considering the model of the reference architecture, given the number of different components, the architectural model is composed of 40 blocks (among which 19 are different). The GSPN of the reference system has 183 places and 376 transitions for X4 (182 places and 351 transitions for Y4). The GSPN for the extended system has 215 places and 580 transitions for X5 (213 places and 643 transitions for Y5). We have first derived the Markov chain for X4 and Y4 with two successive failures before system unavailability (i. e., truncation of order 2) and the Markov chain corresponding to a truncation of order 3. The error related to the dependability measures is less than

---

2 Note that functional interactions between the components may appear; they will be taken into account in a specific part of the service model.

1%. We have thus considered models with a truncation of order 2 to process models with less number of states and hence reduce the execution time mainly for sensitivity analyses where several executions are needed.

To illustrate the approach, we present the GSPN related to the processors within a node. Our objective is to show how the architectural model communicates with service model through the failure mode layer.

4.2 The processor model

The four processors of a node are modeled by the GSPN of Figure 4. The upper part of the figure gives a simplified view of the service model (corresponding to X4): places in the left side represent the four service levels and places in the right side represent the maintenance states: P_{main}: call for delayed maintenance, P_{Reb}: reboot state and P_M: the repairman presence place. The lower part of the figure gives the architectural model with the failure mode layer.

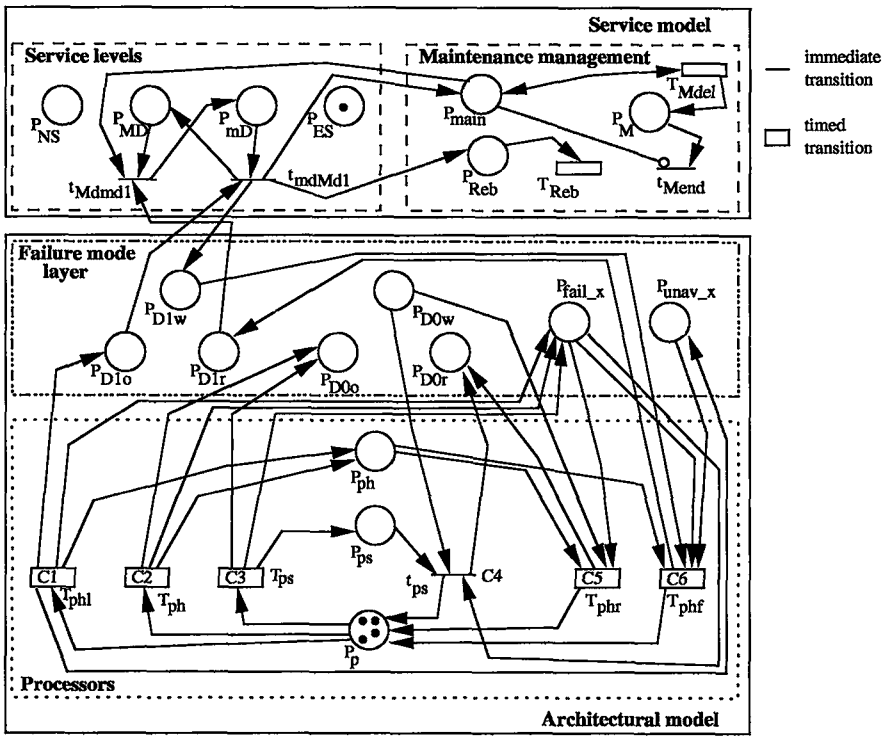
All transitions are not shown for clarity. In particular, in the architectural model, the enabling conditions Ci associated with the transitions are indicated at the bottom of the figure.

In the architectural model, P_p gives the number of operational processors of node x (its initial marking is 4). After a software failure of a processor (transition T_{ps}), the system is rebooted through t_{ps}. A hardware failure of a processor (transition T_{ph}) leads to a minor service degradation level. However the failure of the last processor (transition T_{ph1}) leads to a major service degradation level. P_{ps} represents the number of processors with a software failure waiting for reboot and P_{ph} those with a hardware failure waiting for repair.

The failure mode layer gathers all the architecture's failure information. Figure 4 shows only information used or updated by the processor model. Each failure mode Z has 3 places:

1) P_{Zo}: occurrence of a failure bringing the system into failure mode Z that has to be taken into account in the service model;
2) P_{Zw}: waiting for repair;
3) P_{Zr}: end of processor repair; the service degradation level has to be accordingly updated.

Also the interface contains places summarizing the state of each node x: P_{fail_x} (number of failures in x) and P_{unav_x} (when it is marked, node x is unavailable).



Transitio	Rate	Definition
T_{ph}	$M(P_p)*\lambda_{ph}$	Hardware failure ($M(P_p)$: marking of P_p; λ_{ph}: hardware failure rate of a processor).
T_{ps}	$M(P_p)*\lambda_{ps}$	Software failure of a processor.
T_{phl}	λ_{ph}	Hardware failure of the last processor.
t_{ps}	-	The processor becomes available after software failure and reboot.
T_{phr}	μ_{ph}	Processor repair after a hardware failure (the node has not been isolated).
T_{phf}	μ_{ph}	Repair of one processor, reintegration of the node
T_{Mdel}	σ_{Mdel}	Delayed maintenance rate.
T_{Reb}	σ_{Reb}	Reboot rate.
t_{Mend}	-	End of maintenance intervention (there's no more failure in the system).
t_{mdMd1}	-	Change of service degradation level following the failure of the last processor.
t_{Mdmd1}	-	Change of service degradation level after a repair from MD-S.

C1: $M(P_{ph}) = 3$ and $M(P_{NS}) = 0$ and $M(P_{unav\_x}) = 0$

C2: $M(P_{ph}) > 3$ and $M(P_{NS}) = 0$ and $M(P_{unav\_x}) = 0$

C3: $M(P_{NS}) = 0$ and $M(P_{unav\_x}) = 0$

C4: $M(P_{Reb}) = 1$

C5: $M(P_M) = 1$

C6: $M(P_M) = 1$

Figure 4: GSPN of the four processors of a node

The model is explained through an example. Let us assume that the last available processor in node x fails. The firing of T_{phl} removes a token from P_p and puts a token in 1) P_{ph} (there is an additional processor failure in node x), 2) P_{fail_x} (there is an additional failure in node x), 3) P_{unav_x} (node x becomes unavailable) and 4) P_{D1o} (D1 failure mode). The failure is thus recorded in the failure mode layer to update the service model. Assuming that the system was in mD, the token from P_{D1o} enables t_{mdMd1} whose firing changes the current service level from mD to MD and puts a token in P_{D1w} meaning that the failure has been recorded and is waiting for repair. Also it puts a

token in P_{main} for an additional call for maintenance. The presence of a token in P_{main} enables T_{Mdel}, corresponding to the delayed maintenance call. The firing of T_{Mdel} means that a repairman has arrived and the repair is performed. After the repair of a processor of node x, the token is moved from P_{D1w} to P_{D1r} and the service level returns to mD consuming a token from P_{D1r}.

4.3 Model parameters

The nominal values of the model parameters (failure rates, repair time, maintenance delay and arrival times) have been either provided by the system manufacturer or assigned according to our experience. Sensitivity analyses have been performed to evaluate their relative influence on dependability and performability measures. Unless specified, the results of the next section have been obtained using the nominal values.

5. Results

Based on the models of the reference architecture and the extended architecture, as well as on the nominal parameter values, the dependability measures presented in Section 3 have been evaluated for users X and Y and several sensitivity analyses have been carried out. We have selected a subset of results to show the influence of some parameters. We first give some results related to user X, then to Y, before summarizing the results.

5.1 User X

Influence of the maintenance time. The immediate maintenance time is equal to the sum of the maintenance arrival time and the repair time (Cf. 2.4). The nominal value of each time is 2 hours and it is assumed that all components have the same repair time. In order to analyze the influence of the maintenance time on dependability, we carry out a sensitivity analysis with respect to the repair time.

Table 2 shows that the sojourn time in mD-S, Lm, is not sensitive to the repair time, while the sojourn time in MD-S, LM, and the unavailability are affected. To a large extent, the unavailability of the system is due to the reboot time as shown by the last column. The difference between the last two columns gives the unavailability due to NS-S. Reducing the repair time by one hour reduces the sojourn time in NS-S by 21 min per year (from 35 min to 14 min).

Repair time	LE (probability)	Lm	LM	UA	UA_{Reb}
2 h	0.98913	47h40	46h13	1h20	0h45
1 h	0.98926	47h42	45h23	0h58	0h44

Table 2: X4 dependability measures in hours per year (and probability) according to the repair time

Using the nominal values of the parameters, the mean time to failure (MTTF) is 42033 hours (4.8 years) and the mean downtime (MDT) due to NS-S is 2 h 43. The sojourn time in NS-S is 34 min per year (= 2 h 43 min / 4.8 years); which is in accordance with the 35 min obtained from Table 2.

Influence of the reboot time. The nominal reboot time is 20 min. Table 2 suggests that on average there are two system reboots per year. Table 3 confirms this result for different values of the reboot time. As mentioned in Section 2.1, it is assumed that all system reconfigurations force a reboot. In addition, the re-insertion of the off-line repaired component(s) necessitates a system reboot. The results of Tables 2 and 3 argue in favor of performing a reconfiguration without a reboot whenever possible or with fast reboot (lasting less time).

Also Table 3 shows that the 35 min of unavailability due to NS-S are independent from the reboot time (it is given by the difference between UA_{Reb} and UA).

Reboot time	LE (probability)	Lm	LM	UA	UA_{Reb}
10 min	0.98916	47h41	46h18	0h57	0h23
20 min	0.98913	47h40	46h13	1h20	0h45
30 min	0.98910	47h41	46h09	1h41	1h07

Table 3: X4 dependability measures according to the reboot time

Influence of the maintenance delay: Immediate maintenance is performed only when the system is in NS-S. When the system is in a degraded service state, maintenance is delayed. The average nominal maintenance delay is one week for both minor and major service degradation states. Table 4 shows that if the maintenance delay is two weeks for mD-S, the time spent in mD-S, Lm, is almost multiplied by two. On the other hand, when the delay is reduced to two days for MD-S, the time spent in MD-S, LM, is significantly reduced. In both cases, unavailability is not affected.

Delay in mD-S / MD-S	LE (probability)	Lm	LM	UA
1 week / 1 week	0.98913	47h40	46h13	1h20
2 weeks / 1 week	0.98378	94h33	46h15	1h19
1 week / 2 days	0.99274	47h51	14h25	1h19

Table 4: X4 dependability measures according to maintenance delay in mD-S and MD-S

Another possibility could be to perform the delayed maintenance in a more regular manner: i. e., to make periodic maintenance even without any error reported. However, periodic maintenance could be more expensive than on-request maintenance if the time between two maintenance interventions (i. e., its period) is too short (to improve system availability). A tradeoff has to be made: the periodicity can be optimized through the evaluation of the number of visits to the delayed maintenance states. The models we have developed can be modified to model periodic

maintenance: this modification affects only the maintenance part of the model and its interactions with the failure mode layer.

Influence of the hardware failure rate (λph). The nominal hardware failure rate of a processor is 10^{-6}/h. Table 5 shows the sensitivity of system dependability to this failure rate. If this rate is one order of magnitude lower, the unavailability is reduced by 8 min per year, whereas if the failure rate is one order of magnitude higher, it is increased by 1 h 09 min per year. The most influenced level corresponds to the minor service degradation level. This is due to the presence of 16 processors in the system whose first three successive failures lead to minor service degradation.

λph	LE (probability)	Lm	LM	UA
1.0E-07	0.99145	28h02	45h40	1h12
1.0E-06	0.98913	47h40	46h13	1h20
1.0E-05	0.96602	245h20	49h52	2h29

Table 5: X4 dependability measures according to λph

Influence of the remote cache controller (RCC) failure rate. The nominal failure rate of an RCC is 10^{-7}/h. Table 6 shows that system dependability is affected by the value of this failure rate. Indeed, a failure rate of 10^{-5}/h increases the system unavailability by 14 min per year and doubles the sojourn time in LM (corresponding to the major service degradation level, MD-S). The most influenced level is LM; this is due to the presence of 4 RCCs whose failures lead to MD-S.

λRCC	LE (probability)	Lm	LM	UA
1.0E-07	0.98913	47h40	46h13	1h20
1.0E-06	0.98855	47h42	51h13	1h21
1.0E-05	0.98282	47h47	101h10	1h34

Table 6: X4 dependability measures according to λRCC

Influence of the spare node. Table 7 gives the dependability measure for the extended architecture, system X5. These results are to be compared with those of Table 2. It can be seen that the system unavailability is unaffected but the time spent in MD-S, LM, is considerably reduced. The "Spare use" column indicates the time during which the spare is used. For the nominal values, this time is 46 h 25 per year: it is distributed among mD-S and ES-S. Note that the major service degradation level, LM, corresponds here to the loss of 2 nodes before maintenance achievement. A sojourn time in MD-S of 14 min per year shows that the double failure is unlikely.

Arrival / repair times	LE (probability)	Lm	LM	UA	UA_{Reb}	Spare use
2 h / 2 h	0.99426	48h46	0h14	1h19	0h45	46h25
1 h / 2 h (or 2 h / 1h)	0.99433	48h28	0h14	0h58	0h44	45h35

Table 7: X5 dependability measures

Expected reward rate. Let us assume that the performance index of a given state class represents the percentage of system processing capacity in this class (an index of 0.6 for example, mean that the processing capacity is 60 %). We have thus $r_{ES\text{-}S}$ =1 and r_{UA} = 0. In this context, Table 8 gives the expected reward rate at steady state with various values of the performance indexes r_{mD} and r_{MD}. Obviously, X5 has higher expected reward rate than X4. Note that X5 is less sensitive to the performance index associated to MD-S because of the reduced sojourn time in MD-S. The last line gives the system availability

Performance indexes	E[W] for X4	E[W] for X5
r_{mD} = 0.8 ; r_{MD} = 0.6	0.996650	0.998726
r_{mD} = 0.8 ; r_{MD} = 0.7	0.997178	0.998728
r_{mD} = 0.9 ; r_{MD} = 0.7	0.997722	0.999285
r_{mD} = 0.9 ; r_{MD} = 0.8	0.998250	0.999288
r_{mD} = 1 ; r_{MD} = 1 **(A)**	A = 0.999849	A = 0.999850

Table 8: Expected reward rate at steady state, E[W]

5.2 User Y

For Y, the service is either entire or nil. The availability results according to the repair time are summarized in table 9. Reducing the repair time improves the availability of the system. This is not surprising since most of the failures lead to immediate maintenance. Note that, as opposed to what was observed for X4, system unavailability is mainly due to NS-S (difference between the last two columns): 3 h 25 compared to 24 min of unavailability due to system reboot.

Repair time	A (probability)	UA	UA_{Reb}
2 h	0.99957	3h49	0h24
1 h	0.99970	2h37	0h24

Table 9: Y4 availability according to the repair time

Considering the nominal values, the MTTF is 7175 h and the MDT due to NS-S is 3 h 09 min. This means that immediate maintenance is called on average a little bit more than once a year and the reboot time of 24 min corresponds to system reboot after maintenance: the system does not exercise reboots in ES-S as all failures are tolerated without system reconfiguration. Recall that for X4, the immediate maintenance is called on average once every 4.8 years, but the system is rebooted on average twice a year.

The maintenance delay rate does not influence the availability of the system: the maintenance is delayed only for benign failures that do not affect service delivery. Sensitivity analyses with respect to hardware processor and to the RCC failures are similar to those obtained for X4.

Influence of the spare node: The presence of a spare node directly impacts the availability of the system since the first node failure can be tolerated. Table 10 shows that unavailability is divided by

3 for Y5 compared to Y4 (Table 7). Moreover, it can be seen that the unavailability is of the same order of magnitude as for X4 (these values are 1h20 and 58 min from Table 2).

Considering the nominal values of the parameters, the time during which the spare is used is 93h 08 min. This time is almost equal to the sum of Lm and LM of Table 2, line 1, that is 93 h 53 min. The difference (45 min) corresponds to the time spent in states without an external disk (failure mode DD defined in Section 2; also Cf. Table 1).

Repair time	A (probability)	UA	UA_{Reb}	Spare use
2 h	0.99986	1h15	0h41	93h08
1 h	0.99990	0h55	0h40	92h17

Table 10: Y5 dependability

5.3 Tradeoff dependability - performance

Table 11 reports, from the previous tables, the expected reward rate and availability of the four systems for the nominal values of the parameters (for X, the expected reward rate is equal to availability). It shows that the reference architecture provides better availability for user X and better performance for Y: there is thus a tradeoff between system performance and availability. On the other hand, the extended architecture provides better availability and better performance for Y. However, the difference between X and Y availability is not significant. We should be careful about the latter result as it is obtained for the nominal values of the parameters. Additional sensitivity analyses are under investigation to evaluate the impact of other important parameters. In particular, we are considering the influence of coverage factor.

Measure	User	Reference architecture (4 nodes)	Extended Architecture (5 nodes)
Expected reward rate	X	0.99825	0.99929
	Y	0.99957	0.99986
Availability	X	0.99985	0.99985
	Y	0.99957	0.99986

Table 11: Comparison of the reference and extended architectures

5.4 Summary of results

The main results presented in this paper can be summarized as follows:

For user X:

- system unavailability is mainly due to the reboot time;
- the no service states are reached on average once each 4.8 years but the system is rebooted twice a year;

- the maintenance delay affects only the sojourn time in states with minor and major service degradation; while the repair time affects system unavailability;
- the addition of a spare does not affect system unavailability but reduces the sojourn time in states with major service degradation.

For user Y:

- system unavailability is mainly due to the maintenance time;
- the no service states are reached on average once a year and the system is rebooted once a year (following system maintenance);
- the addition of a spare considerably reduces system unavailability.

The results revealed the existence of a tradeoff between system availability and system performance. Sensitivity analyses showed the influence of the failures rates of the processors and the remote cache controllers on system dependability. For example selecting a processor with a failure rate one order of magnitude higher to the nominal value assumed by the manufacturer will increase the system unavailability by 50 %.

6. Conclusions

This paper was devoted to the dependability evaluation of a multipurpose, multiprocessor system, MMS, under investigation by a system manufacturer. We have presented a reference architecture and an extended architecture and compared their dependability measures considering two examples of users with different service requirements. Modeling is based on GSPNs to which reward rates are added to evaluate performability. The modeling approach is modular, as many of other published approaches (see e. g., [Meyer and Sanders 1993; Kanoun, Borrel et al. 1996; Nelli, Bondavalli et al. 1996; Fota, Kâaniche et al. 1998]). The originality of our approach is the separation between the architectural and the service concerns. This separation of concerns is very important in particular as we are considering a system manufacturer perspective, in which the user's needs are explicitly accounted for. In a typical user's perspective, as the user is interested in comparing possible architectural solutions for the same service purposes, there is no need to consider explicitly service concerns and usually emphasis is put on architectural concerns. Note that, even though several publications have been devoted to multiprocessors systems (see e. g., [Marsan, Balbo et al. 1984; Muppala, Sathaye et al. 1992], none of them addressed explicitly the manufacturer and the user concerns at the same time. Moreover none considered in detail the system architecture with all components.

The results presented for MMS can be classed into two categories: those supporting the manufacturer choices and those that will support the potential user choices. Of course, these results

are not independent and have to be used together. Proper design choices by the manufacturer will — hopefully — be of great benefits for the user.

From the manufacturer perspective, the results are mainly related to:

- the selection of the processors and of the remote cache controllers, RCC, according to the impact of their failure rates on dependability measures (and their cost most probably);
- the decision concerning the reboot policy: reboot after system reconfiguration or not, reboot with or without on-line tests;
- the provision of a spare node. With respect to this point, a tradeoff should be made between the dependability improvement and the additional difficulty for developing the underlying mechanisms for the insertion of the spare into the system.

From the user perspective, the results concern:

- the selection of the maintenance policy: delayed or immediate maintenance, tradeoff between the maintenance guaranteed time and the cost of the maintenance contract (to agree on with the system provider);
- the selection between the reference architecture and the extended one, and more generally between all available solutions.

Another important point of interest concerns the exploitation by the user of the various degradation possibilities offered by the architecture. According to the service expected by the application, the user has to make choices concerning service degradation levels he can accept and the tradeoff between performance and availability. This choice may affect the architecture of the applicative software architecture.

The work is under progress. More specifically, additional performability measures are under investigation, particularly to study some specific points such as the dependability improvement induced by distributed, shared memories. Also, other extended architectures are under consideration. As the architectural model is modular, and the component models are generic, their modeling is based on extensive reuse of the GSPN developed in this paper.

References

[Arlat and Laprie 1983] J. Arlat and J.-C. Laprie, "Performance-Related Dependability Evaluation of Supercomputer Systems", *Proc. Proc. 13th Int. Symp. on Fault-Tolerant Computing (FTCS-13)*, Milano, Italy, IEEE Computer Society Press, pp. 276-283, 1983.

[Ciardo, Muppala *et al.* 1989] G. Ciardo, J. Muppala and K.S. Trivedi, "SPNP: Stochastic Petri Net Package", *Proc. 3rd Int. Workshop on Petri Nets and Performance Models*, Los Alamitos, CA, USA, pp. 142-151, 1989.

[Fota, Kâaniche *et al.* 1998] N. Fota, M. Kâaniche and K. Kanoun, "Dependability Evaluation of an Air Traffic Control System", *Proc. 3rd IEEE Int. Computer Performance & Dependability Symposium (IPDS)*, Durham, NC, pp. 206-215, 1998.

[Howard 1971] R.A. Howard, *Dynamic Probabilistic Systems*, New York, J. Wiley and Sons, 1971.

[Kanoun, Borrel *et al.* 1996] K. Kanoun, M. Borrel, T. Moreteveille and A. Peytavin, "Modeling the Dependability of CAUTRA, a Subset of the French Air Traffic Control System", *Proc. 26th IEEE Int. Symp. Fault-Tolerant Computing (FTCS-26)*, Sendai, Japan, pp. 106-115, 1996.

[Kloppenburg 1987] T. Kloppenburg, "LOGISIRE — A Safe Computer System for Process-Automation", *Proc. 3rd Intern. GI/IGT/GMA Conf.*, Bremerhaven, Germany, Springer-Verlag, 1987.

[Marsan, Balbo *et al.* 1984] A. Marsan, G. Balbo and G. Conte, "A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems", *ACM Transactions on Computers,* 2 (2), pp. 93-122, 1984.

[Meyer 1978] J.F. Meyer, "On Evaluating the Performability of Degradable Computing Systems", *Proc. 8th IEEE Int. Symp. Fault-Tolerant Computing (FTCS-8)*, Toulouse, France, pp. 43-52, 1978.

[Meyer and Sanders 1993] J.F. Meyer and W.H. Sanders, "Specification and Construction of Performability Models", *Proc. Int. Workshop on Performability Modeling of Computer and Communication Systems*, Mont Saint Michel, France, pp. 1-32, 1993.

[Muppala, Sathaye *et al.* 1992] J.K. Muppala, A. Sathaye, R. Howe, C and K.S. Trivedi, "Dependability Modeling of a Heterogeneous VAX-cluster System Using Stochastic Reward Nets", *Hardware and Software Fault Tolerance in Parallel Computing Systems*, Ed. D.R. Avresky, pp. 33-59, 1992.

[Nelli, Bondavalli *et al.* 1996] M. Nelli, A. Bondavalli and L. Simoncini, "Dependability Modeling and Analysis of Complex Control Systems: An Application to Railway Interlocking", *Proc. 2nd European Dependable Computing Conf.*, Taormina, Italy, Springer-Verlag, 1996.

[Poueyo and Dalzon 1998] J. Poueyo and J.P. Dalzon, "Mastering Safety of Opel and Distributed Systems in Compliance with IEC 61508", *Proc. IFAC 2nd Symp. on Information Control Manufacturing (INCOM'98)*, Nancy-Metz, France, pp. 511-516, 1998.

[Protic, Tomasevic *et al.* 1998] J. Protic, M. Tomasevic and V. Milutinovic, Eds. *Distributed Shared Memory — Concepts and Systems*, IEEE Computer Society, 1998.

[Tomek, Mainkar *et al.* 1994] L.A. Tomek, V. Mainkar, R.M. Geist and K.S. Trivedi, "Reliability Modeling of Life-Critical, Real-Time Systems", *Proceeding of the IEEE, Special Issue on Real-Time Systems,* 82 (1) , pp. 108-121, 1994.

[Triconex 1996] Triconex, TRICON Technical Product Guide, Version 9 Systems, Irvine, Triconnex Corporation, , 1996.

[Wirthumer 1989] G. Wirthumer, "Votrics — Fault Tolerance Realized in Software", *Proc. 8th IFAC Int. Conference on Computer Safety, Reliability and Security (SAFECOMP'89)*, Vienna, Austria, pp. 135-140, 1989.