

Reliability Optimization for Software Systems With Multiple Applications

Naruemon Wattanapongsakorn

Department of Computer Engineering, King Mongkut's University of Technology, Thonburi, Bangkok, Thailand
naruemon@cpe.eng.kmutt.ac.th

Abstract

This paper presents a model for optimizing both software and hardware reliability for embedded systems under cost and multiple reliability constraints. Previously, most optimization models have been developed for hardware-only or software-only systems by assuming the hardware, if any, has perfect reliability. In addition they assume that failures for each hardware or software component is statistically independent. In other words, none of the existing optimization models was developed for hardware & software (called embedded) systems with failure dependencies. Our model minimizes system cost under multiple reliability constraints. We use our model for an embedded system with multiple applications sharing multiple resources. This is the first time that optimization of this kind of model is performed on this type of system.

I. INTRODUCTION

For mission critical systems, redundant techniques are commonly applied for high reliability. For embedded systems consisting of software and hardware components, redundancy can be achieved by applying extra copies of these components (in parallel) to handle the system work loads. Various system redundant architectures have been discussed such as N-Version Programming (NVP), N-Self Checking Programming (NSCP) and Recovery Blocks (RB). The architectures result from different ways of integrating software and hardware redundancy, together with some decision algorithms such as voting, acceptance tests or comparison.

In the system that we consider, there are a specified number of subsystems in series. For each subsystem, there are several hardware and software choices to be made. The system is designed using commercial off the shelf (COTS) components, each with known cost and reliability.

We consider that all hardware redundancy is active, and each component is statistically independent from each other. However, software components/versions have related faults among all software versions (P_{all}), due to the faults in the specification, related faults between any two software versions (Prv), and faults from the software voting algorithm (Pd). These types of related faults exist even when each software version is independently developed [2]. Each hardware and software unit is considered non-repairable, and has 2 states: functional or failed, with known reliability (deterministic), that is with either a constant or a time-dependent failure rate.

For each subsystem, a choice of component redundancy, RB/1/1, can be obtained. We choose the

RB architecture choice, because it has potential to give better reliability results compared with other basic architecture choices such as NVP and NSCP. In each subsystem i , software and hardware component failures each has different effects on one another and on the overall system. Related faults (Pall, Prv, and Pd) are included in the system reliability analysis.

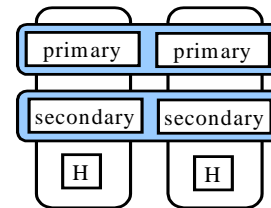


Figure 1. Recovery Block (RB): RB/1/1 Architecture

The diagram of Recovery Block (RB): RB/1/1 Architecture was discussed in [1, 2]. It is depicted in Figure 1 for a single software subsystem. A series system may consist of several or many RB/1/1 subsystems. This RB/1/1 model consists of an adjudication module, two software versions and two hardware components. For two hardware components, each is running two independent versions (primary and secondary). The primary version is active until it fails, and the secondary version is the backup spare. The system fails when both software versions fail, or both hardware components are not functional.

The probability that an unacceptable result occurs during a single task iteration, $1 - R_i(t)$, based on [2], is represented by

$$Prv_{12} + Qrv_{12} Pd + Qrv_{12} Qd P_{all} + Qrv_{12} Qd Q_{all} Ph_1 Ph_2 + Qrv_{12} Qd Q_{all} (1 - Ph_1 Ph_2) P_{v1} P_{v2} \quad (\text{eq. 1})$$

Notation

N	Number of subsystem in the system
m_i	Number of hardware choices available for subsystem i
p_i	Number of software versions available for subsystem i
R	Estimated system reliability
R_i	Estimated reliability of the subsystem i
Rhw_{ij}	Reliability of hardware component j at subsystem i
Rsw_{ij}	Reliability of software component j at subsystem i
Chw_{ij}	Cost of using hardware component j at subsystem i
Csw_{ij}	Cost of developing software version j at subsystem i
Cost	Affordable Cost
P_x	Probability that event X occurs
Q_x	Probability that event X does not occur; $Q_x = 1 - P_x$
P_{v_i}	Probability of failure of software version i
Prv_{ij}	Probability of failure from related fault between two software versions, i and j
P_{all}	Probability of failure from related fault among all software versions, due to faults in specification
P_d	Probability of failure of decider or voter
Ph_i	Probability of failure of hardware component i . If only one hardware is applied, $Ph_i = Ph$
U_j	The utilization function for subsystem j
x_i	The number of applications running on subsystem i .
ConstApp $_j$	Reliability constraint of application j

II. OPTIMIZATION MODEL

$$\begin{aligned}
 \text{Min Cost} &= \sum_{i=1}^n \sum_{j=1}^{m_i} X_{ij} C_{ij} + \sum_{i=1}^n \sum_{k=1}^{p_i} Y_{ik} C_{ik} \\
 \text{Subject to} & \left(\sum_{j=1}^{m_i} X_{ij} = 1 \text{ and } \sum_{k=1}^{p_i} Y_{ik} = 1 \right) \text{ or } \left(\sum_{j=1}^{m_i} X_{ij} = 2 \text{ and } \sum_{k=1}^{p_i} Y_{ik} = 2 \right) \\
 & X_{ij} = 0, 1, 2 \quad Y_{ik} = 0, 1 \\
 \text{where} \quad R_{\text{System}} &= \prod_{i=1}^n R_i \\
 R_i &= \sum_{j=1}^{m_i} \sum_{k=1}^{p_i} X_{ij} Y_{ik} R_{hw_{ij}} R_{sw_{ik}}, \text{ if } \sum_{j=1}^{m_i} X_{ij} = 1, \sum_{k=1}^{p_i} Y_{ik} = 1 \\
 R_i &= Eq.(3), \text{ if } \sum_{j=1}^{m_i} X_{ij} = 2, \sum_{k=1}^{p_i} Y_{ik} = 2 \\
 \text{For each application } j: R_j &= \prod_{i=1}^{a_j} (U_i R_i) \geq \text{ConstApp}_j \\
 U_i &= (0.99)^{\lfloor x_i/2 \rfloor}
 \end{aligned}$$

From the optimization model above, the total system cost is the summation of all the selected software and hardware components. For each subsystem, many choices of software versions and hardware components are available. Each of the versions or components has a reliability and cost. This model allows a choice of component redundant architecture RB/1/1 for each subsystem. For a subsystem without redundancy, a single software version and a hardware component is selected, while with RB/1/1 redundancy, two different software versions and two hardware components with the same type and reliability are obtained. The constraints for this model are reliability requirements for some or all applications running on a common system platform. Reliability of each subsystem is calculated using Equation (eq. 1) if the RB/1/1 redundancy is obtained. Otherwise, it is multiplication of the reliability of the selected software version and the hardware component. To make the problem more realistic, utilization functions are considered such that a subsystem reliability is degraded when it is loaded by multiple applications.

III. AN EXAMPLE SYSTEM

We can perform system cost optimization with reliability constraints for multiple applications sharing the same hardware and software resources. For an example, we use the real-time system shown in Figure 2. The system contains three network subsystems; Bessel, Bohr and Fourier, and five subsystems; Galileo, Faraday, Halley, Kirchoff, and Ohm. Six applications assigned to run on this system are applications A, B, C, D, E, and F. As a result, many hardware resources have to be shared among the applications.

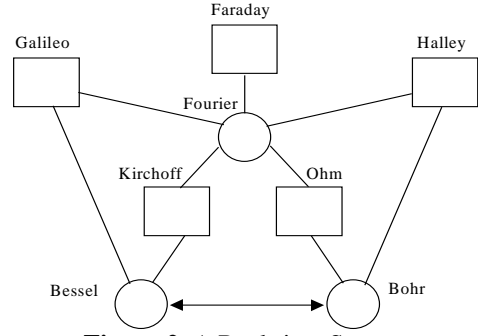


Figure 2. A Real-time System

Since each application consists of its subsystems connected in series, the system reliability for each application can be obtained easily, provided that each subsystem reliability is known. Then, the reliability of the total system consisting of all applications can be calculated, where all the utilized subsystems are considered. For instance, subsystems Faraday and Bohr are not used for any application, they are not counted in the total system reliability. The reliability of the system is therefore calculated as:

$$R_{\text{system}} = U_G R_{\text{Galileo}} \times U_H R_{\text{Halley}} \times U_K R_{\text{Kirchoff}} \times U_O R_{\text{Ohm}} \times U_F R_{\text{Fourier}} \times U_B R_{\text{Bessel}}$$

Using the optimization model, we can perform system cost optimization with multiple reliability constraints. For example, we may consider different cases of reliability constraints, such as

Case 1: Application A, and B each requires 90 percent reliability

Case 2: Applications A and D each requires 95 percent reliability, applications B, C, and E each requires 92 percent reliability, and application F requires 90 percent reliability.

Implementing the optimization model, we can find the optimization results for each case, and also perform a comparison of the results to see how the constraints affect the system cost, system reliability, and the reliability of other applications on the same system.

IV. FINAL REMARKS

In this abstract, we present an optimization model for software systems loading with multiple applications. We consider related faults or failure dependencies in software components, and multiple application constraints. This model builds on our earlier work [3].

References

1. J.-C. Laprie, J. Arlat, C. Beounes, and K. Kanoun, Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures, IEEE Computer, July 1990, pp 39-51.
2. M. R. Lyu, Editor in Chief, Handbook of Software Reliability Engineering, IEEE Computer Society Press, McGraw-Hill, 1996.
3. N. Wattanapongsakorn, S. P. Levitan, "Reliability Optimization Models for Fault-Tolerant Distributed Systems", Proc. Ann. Reliability & Maintainability Symp., Jan 2001, pp. 193-199.