Optimal number of hosts in a distributed system based on cost criteria

M. Xie \(^a\), Y. S. Dai \(^a\), K. L. Poh \(^a\) & C. D. Lai \(^b\)

\(^a\) Department of Industrial and Systems Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119 260
\(^b\) Institute of Information Sciences and Technology, Massey University, New Zealand

To cite this article: M. Xie, Y. S. Dai, K. L. Poh & C. D. Lai (2004): Optimal number of hosts in a distributed system based on cost criteria, International Journal of Systems Science, 35:6, 343-353

To link to this article: http://dx.doi.org/10.1080/00207720410001716228
Optimal number of hosts in a distributed system based on cost criteria

M. XIE†*, Y. S. DAI†, K. L. POH† and C. D. LAI‡

Redundant or distributed systems are increasingly used in system design so that the required reliability and availability can be easily achieved. However, such an approach requires additional resources that can be very costly. Hence, how to design and test such a system in the most cost-effective way is of concern to the developers. A general cost model and a solution algorithm are presented for the determination of the optimal number of hosts and optimal system debugging time that minimize the total cost while achieving a certain performance objective. During testing, software faults are corrected and the reliability shows an increasing trend, and hence system reliability increases. A general system model is constructed based on a Markov process with software reliability and availability obtained from software reliability growth models. The optimization problem is formulated based on the cost criteria and the solution procedure is described. An application example is presented.

1. Introduction

One of the important goals in system design is to achieve some kind of redundancy or fault tolerance so that the reliability and availability of the system is improved. There are many studies on fault tolerance techniques. Ammar et al. (2000) presented a comparative survey of fault tolerance techniques as it arises in hardware systems and software systems. Kuo and Prasad (2000) summarized an overview of the methods for solving various reliability optimization problems. These studies show that fault tolerance techniques can increase system reliability significantly, but at the same time, it is very expensive. Dai (2003) studied distributed systems from a service reliability point of view.

One type of fault tolerance techniques is copying the same version of software on multiple hosts (e.g. Laprie and Kanoun 1992, Pham 1992, Mitra et al. 2002, Philippi 2003). Many systems are developed in multiple host environments (Kreimer 2002). For example, a telephone switching system contains several servers to provide the service. The software on these servers is identical and the type of the servers can also be same.

The number of redundant hosts will have significant influence on the cost and system availability, because it can be very costly to prepare the hosts but hosts are able to improve system availability easily.

System availability is a major performance measure for complex systems. Earlier research on system redundancy has been concerned with system availability calculation and redundancy optimization assuming a fixed reliability of the components. On the other hand, during the testing stage, software parts are usually debugged in a sense that faults detected are corrected, so that the reliability of the system is increased. The additional variable here is the time we need to test the system. If it is tested for too long, the release is delayed while a short test duration may not achieve the required reliability level of the system.

The number of redundant hosts also affects the test time duration and it is usually an important system design parameter. During the test phase, the redundant hosts run together and if any one fails in software parts, the debugging process starts to remove the faults. After debugging, the improved new version of software is copied to other hosts. Obviously, the more hosts, the more likely the faults in the software are detected.
and removed. Thus, the testing process is accelerated and the testing time can be reduced.

However, installing redundant hosts is very expensive although it can improve the system availability and accelerate the test process. Thus, how to optimize the number of hosts is an important objective, and this will be studied in this paper.

The release time is another important decision variable together with the number of hosts that will also be considered. There are usually two different stages before and after the release time. Before the release, we have a testing stage and the operational stage comes after the release. This paper will consider different types of cost for both stages in order to minimize the total expected cost.

Optimization models have been studied in computing and software systems. Okumoto and Goel (1980) were the first to discuss the software optimal release policy from the cost–benefit viewpoint. Sheu (1992) considered periodic testing policy for a computer system that fails when the total number of hidden faults exceeds a threshold level. Recently, Berman and Kumar (1999) considered some optimization models for recovery block design. Jung and Choi (1999) studied some models for modular systems. Lee (2003) discussed the application of heterogeneous systems in reliability context. Imaizumi et al. (2000) derived the mean time and the expected cost until system failure and discussed an optimal number that minimizes the expected cost for a system with \( N \) microprocessor units. Nevertheless, these and other related models usually do not consider the combined software and hardware.

This paper first analyses the reliability of combined software and hardware systems in a distributed or redundant environment, and then presents an optimization model to determine the number of redundant hosts and release time based on the cost criterion. The paper is organized as follows. In Section 2, a general model is presented for the distributed system, and the cost factors in the development and maintenance process are discussed. The analysis of system availability with software reliability growth model is also introduced. Section 3 presents the optimization procedure for the optimal number of redundant hosts and the optimal release time. In Section 4, an example is given to illustrate the application of this optimization model to a system having both software and hardware failures. Section 5 concludes this paper.

1.1. Notation

- \( \lambda_h \) hardware failure rate of an exponential distribution,
- \( \lambda_s(t) \) software failure rate at time \( t \),
- \( \mu_s \) software repair rate,
- \( \mu_h \) hardware repair rate,
- \( t_r \) release time,
- \( T_d \) deadline for release,
- \( T_e \) ending time of contracted maintenance after release,
- \( K_f \) number of faults remaining in the software at time \( t \),
- \( \lambda \) software failure rate per fault,
- \( N \) number of hosts,
- \( C_d(N) \) cost of hosts,
- \( B(t_r) \) reward (benefit) function for system to be released ahead of deadline,
- \( C_s(N, t_r) \) risk cost function for unavailable system,
- \( C_s(R) \) software development cost to achieve reliability level \( R \),
- \( C(N, t_r) \) expected total cost with \( N \) hosts and release time \( t_r \),
- \( A_3(t) \) availability function at time \( t \),
- \( R(t) \) software reliability at the release time point,
- \( A^* \) required minimum system availability when the system is released,
- \( \text{HDSHS} \) homogeneous distributed software/hardware system,
- \( \text{OpTr}(N) \) optimal release time under the condition that the number of hosts is \( N \),
- \( \text{minC}(N) \) minimum expected total cost under the condition that the number of hosts is \( N \).

2. System and cost model for \( N \) hosts

2.1. Model for \( N \) hosts system

Hosts are used to increase the system availability for critical operation in the system. In this paper, a Markov model is used to describe the failure and repair process of the systems with \( N \) hosts. Since computing systems do not age and the requests usually follow a Poisson distribution, Markov models associated with exponential distribution that are commonly used and can be justified here.

The physical system is assumed to contain \( N \) software subsystems (SW1–SWN) running on \( N \) hosts (HW1–HWN) (figure 1). Lai et al. (2002) studied a type of widely implemented system with \( N \) homogeneous hot hosts, called HDSHS (Homogeneous Distributed Software/Hardware System). That is, identical copies of application software run on the same type of computers. This system may be implemented to provide services for uncorrelated random requests of customers. Such a system has been recently studied, from the functional point of view, by various researchers (e.g. Fragniere et al. 2000, Guo and Liu 2000, Yamada et al. 2000, Das and Woodside 2001, Dini 2001). Note that this
type of system design provides some kind of redundancy and hence fault-tolerance in the system.

For example, a telephone switching system contains several servers to provide the telephone service. The software on these servers is identical and the type of the servers can also be same. These servers are working simultaneously to receive uncorrelated random calls. After a failure occurs in a server, the maintenance personnel begin to repair and debug it. The system is available unless all the servers fail at the same time (here failure could be a physical breakdown of the system or just being busy). Similarly, there are many counters in a bank and each counter has a workstation in which the identical software is running. The bank is available to provide service unless no workstation can work. Thus, in this paper, the optimal number of homogeneous redundant hosts for this kind of HDSHS is studied.

Assume that the system has two states, up (working) and down (malfunctioning). When the host is at working state, it has a certain probability to fail. When the host is at malfunctioning state, there are maintenance personnel to repair it and restore the system function. We generally assume that the repair of the hardware simply restores the system while the repair of software is a debugging effort leading to software improvement. Hence, our emphasis here is on the software development aspect and its impact on such a distributed system.

The status of the \( N \) hosts distributed system, in terms of the number of functioning hosts, can be described by a Markov transition graph, as in figure 2.

In figure 2, state \( n \) is the state of the system when there are \( n \) hosts in working state and \( N-n \) hosts in malfunctioning state \( (n=0,1,\ldots,N) \). \( Pr_F(n+1) \) is the probability that any one of \( (n+1) \) working hosts will fail in the next state, and \( Pr_R(n+1) \) represents the probability that any one of \( N-n \) malfunctioning hosts is repaired in the next transition. If there is no host in the working state, the system is unavailable which is represented by the state 0. Also, \( P_d(t) \) denotes the probability that, at time \( t \), the system stays at state \( n \). The system availability can then be expressed as \( A_d(t)=1-P_d(t) \).

Emphasis in this paper is on the changing software reliability, and hence system availability, caused by the debugging of the software system. That is, the reliability of the software component is changing because detected faults are removed in system testing. This will be discussed in Section 2.3 after the cost model is introduced.

2.2. Cost model

To illustrate the relationships among the decisions and costs, an influence diagram that provides simple graphical representations of decision situations is shown in figure 3. Different decision elements are shown in the influence diagram as of different shapes (e.g. Clemen 1995, pp. 50–65).

The number of hosts \( N \) and the release time \( t_r \) are two decision variables that are required to desire at the design phase. The number of hosts will affect the optimal decision of the release time. Both the number of hosts and release time will affect the system availability, which is the probability for the system to be available. These three factors determine the development cost. The number of hosts also decides the cost of hosts.
The release time determines the rewards or penalty depending on whether the release is before or after the deadline. If the system is unavailable after release, a cost is incurred, probably according to the contract. Hence, the cost of hosts, the development cost, reward and penalty should be considered together when deriving the total mean cost.

As described in figure 3, the total average cost is aggregated by the cost of hosts, penalty cost for unavailable system, reward for releasing ahead of deadline, and development cost. That is, the expected total cost, \( C(N, t_r) \), when the number of hosts is \( N \) and the release time is \( t_r \) is given by:

\[
C(N, t_r) = C_h(N) + C_r(N, t_r) + C_t(R(t_r)) - B(t_r),
\]

(1)

and each of the cost components will be described in the following.

2.2.1. Cost of hosts. The cost function for a multi-version fault-tolerant system can be described as a linear function to the number of versions. This is used in Laprie et al. (1990) originated from Boehm (1981), and we have:

\[
C_h(N) = a_1 N + b_1,
\]

(2)

where \( N \) is the number of hosts, \( b_1 \) is a constant and \( a_1 \) is defined as the expected cost per host. Here we have assumed the hosts used in the system are of the same type.

2.2.2. Reward for early release. In our model, it is assumed that there is a deadline for release. This is the case when the penalty cost for delay is very high and the system has to be released even if it has not reached a desirable level; this may result in higher maintenance and other costs. On the other hand, because of the competitive market place, there is a reward for releasing the system earlier. It is assumed that a constant \( b_2 \) is rewarded if the system can be released in time, no matter how early the release time is (\( b_2 \) can be set to 0) and \( a_2 \) is the expected reward per unit time before the deadline. Thus, the reward function of the release time can be expressed as:

\[
B(t_r) = a_2(T_d - t_r) + b_2, \quad t_r \leq T_d,
\]

(3)

where \( T_d \) is the deadline for release, \( t_r \) is the release time so that \( (T_d - t_r) \) is the time ahead of the schedule.

2.2.3. Risk cost for system being unavailable. On the other hand, after the system is released, there is a risk for it to be unavailable, and there are contractual consequences. This cost factor is generated by the unavailable system after releasing. If the system is not available after the release, it cannot achieve certain function successfully, so it will lead to additional cost, termed risk cost as in Zhang and Pham (1998). Here
we assume the risk cost for unavailable system is a function of system availability and release time:

\[ C_i(t_r) = a_3 \int_{t_r}^{T_e} [1 - A_N(t)] dt, \]  

(4)

where \( t_r \) is the release time, \( T_e \) is the ending time for contracted maintenance after release, \( A_N(t) \) is the availability function at time \( t \) between \( t_r \) and \( T_e \), and \( a_3 \) is the risk cost per unit time when the system is not available. In the above equation, \( 1 - A_N(t) \) is the probability for the system to be unavailable at time \( t \). Hence, the integral \( \int_{t_r}^{T_e} [1 - A_N(t)] dt \) is the expected time for the system to be unavailable from \( t_r \) to \( T_e \), and then \( a_3 \int_{t_r}^{T_e} [1 - A_N(t)] dt \) expresses the expected risk cost.

2.2.4. Development cost. Since our focus is on system integration testing with the emphasis on software testing and debugging, software development cost includes the cost incurred in testing and debugging to improve the software reliability. The development cost function for a single software module proposed in Kumar and Malik (1991) is:

\[ C_i(R_i) = H_i \times \exp(B_i R_i - D_i), \]  

(5)

where \( H_i, B_i \) and \( D_i \) are constants and \( R_i \) is the individual module software reliability achieved at the end of testing. These parameters are explained in Kumar and Malik (1991). Briefly speaking, the cost is exponentially increasing with respect to the improved reliability of a single module. This means that the higher reliability of the single module is, the more increasing rate of the cost \( B_i \) affects the slope of the curve. A high value of \( B_i \) corresponds to a curve that rises very steeply as reliability approaches one, while a lower value corresponds to a curve that rises gradually. Furthermore, \( H_i \) and \( D_i \) control the scale of the curve together through \( H_i/e^{D_i} \). Note that a HDSHS has \( N \) homogeneous hosts running an identical version of the software.

2.3. Software reliability and system availability

An important problem here is to obtain the system availability function for the risk cost. The availability of the system is affected by the reliability of both software and hardware components. Although the hardware reliability can be assumed to be constant, since after the detection of a software failure corrective action will normally be taken to remove the fault, the software reliability will subsequently increase over time. A time-dependent software reliability model will need to be used. There are many models that can be used to model the reliability of software (Xie 1991, Lyu 1996, Pham 2000). For any software reliability growth model, a reliability function during the testing can be obtained in a straightforward manner.

The system availability model for a HDSHS can be obtained without complication. The state transition graph is the same as figure 4. The corresponding Kolomogorov’s differential equation (e.g. Lai et al. 2002) for the probability that the system is in the state \((i,j)\) at time \( t \) is, for \( i, j \neq 0, N; i + j \leq N \):

\[ P_{i,j}^{'}(t) = \mu_{h} P_{i+1,j}(t) + (N-i-j+1)\lambda_{h} P_{i,j-1}(t) \]
\[ + (N-i-j+1)\lambda_{s} P_{i,j-1}(t) + \mu_{s} P_{i,j+1}(t) \]
\[ - \lambda_{s} P_{i,j}(t), \]  

(6)

\[ \text{State(i,j): i hw down, j sw (on different hosts) down} \]

Figure 4. State transition graph for combined software/hardware failures.
where
\[ x_{i,j} = \mu_s + (N - i - j)\lambda_h + (N - i - j)\lambda_s + \mu_h. \]  
(7)

It is assumed that all \( N \) hosts work properly at the initial time so the initial conditions are
\[ P_{0,0}(0) = 1 \quad \text{and} \quad P_{i,j}(0) = 0, \quad \text{for} \ i, j \neq 0. \]  
(8)

The boundary conditions are:
\[ P_{0,0}'(t) = \mu_h P_{1,0}(t) + \mu_s P_{0,1}(t) - N(\lambda_s + \lambda_h)P_{0,0}(t) \]
\[ P_{0,j}'(t) = \mu_h P_{1,j}(t) + (N - j + 1)\lambda_s P_{0,j-1}(t) + \mu_s P_{0,j+1}(t) - (\mu_h + \mu_s + (N - j)(\lambda_h + \lambda_s))P_{0,j}(t) \]
\[ \text{for} \ j = 1, 2, \ldots, N - 1; \]
\[ P_{i,0}'(t) = \mu_h P_{i+1,0}(t) + (N - i + 1)\lambda_h P_{i-1,0}(t) + \mu_s P_{i,1}(t) - (\mu_h + \mu_s + (N - i)(\lambda_h + \lambda_s))P_{i,0}(t) \]
\[ \times P_{i-1,0}(t) \quad \text{for} \ i = 1, 2, \ldots, N - 1; \]
\[ P_{N,0}'(t) = \lambda_h P_{N-1,0}(t) - \mu_s P_{N,0}(t) \]
and
\[ P_{0,N}(t) = \lambda_s P_{0,N-1}(t) - \mu_h P_{0,N}(t). \]

The system availability for the \( N \)-host based system can be calculated by:
\[ A_N(t) = \sum_{i+j<i} P_{i,j}(t). \]  
(9)

To solve the differential equations, we need to know the number of remaining software faults \( K \) at time \( t \). It is assumed that there is only one version of software to be executed in \( N \) hosts. Each copy of software suffers a constant failure rate \( \lambda_s(t) = K \lambda \) of an exponential distribution (e.g. Jelinski and Moranda 1972).

In the present paper, the JM model is used because of its Markovian interpretation. In general, the reliability function is given by:
\[ R(x|\lambda_s(t)) = \exp(-\lambda_s(t)x). \]  
(10)

Usually the parameters in the intensity function have to be obtained from some initial testing or other earlier information.

According to the JM model, the probability of software having \( n \) remaining faults at time \( t \) is:
\[ P(n) = \binom{K_0}{n} e^{-\lambda t} (1 - e^{-\lambda t})^{K_0-n} \quad \text{for} \ 0 \leq n \leq K_0. \]  
(11)

where \( K_0 \) is the number of initial remaining faults and \( \lambda \) is a constant.

Thus, the expected number of remaining faults at time \( t \) is
\[ K_t = \sum_{n=0}^{K_0} P(n)n. \]  
(12)

By the above equations, the system availability for a different number of hosts, \( N \), can be calculated numerically.

As an example, with \( K_0 = 32 \) and \( \lambda = 0.006 \), the software reliability function for \( x = 10 \) is used to calculate the software development cost. Assuming \( \lambda_h = 0.01 \), \( \mu_h = 0.1 \) and \( \mu_s = 0.13 \), the system availability for a different number of hosts is shown in figure 5.

Note that when the number of hosts increases, the system availability increases at the same time point. At the early stage from 0 to about 25 hours, the system availability decreases, which means the software failure rate is much more than the repair rate initially. After that period, the system availability increases with the testing and debugging process going on. The system availability function can be used in the optimization model that will be described in the following.

3. Optimization model and algorithm

3.1. Optimization model

Our general model is based on the cost criteria, and the decision variables are the number of hosts and the release time. The objective in this optimization model is to minimize the expected total cost. There are three types of constraints in this decision problem. First, the customers may require a minimal system availability \( A^* \) after the release. Second, there is a deadline for the system to be released so the release time should be earlier than that. Finally, the customers may limit the maximum number of hosts \( N^* \) due to their budget and other physical restrictions.

The optimization model is constructed as follows.

Decision variables: number of hosts \( N \) and the release time \( t_r \).

Objective function:
\[ \min \left\{ C(N, t_r) \right\} \]  
(13)

Constraints:
\[ A_N(t_r) \geq A^* \geq 0 \]
\[ 0 \leq t_r \leq T_d \]  
(14)
\[ N = 1, 2, 3, \ldots, N^* \]
where \( A^* \) is the required system availability after the release, \( T_d \) is the deadline for release and \( N^* \) is the maximum number of hosts allowed because of any physical constraints such as space limits. If there is no such constraint, we can assume a very large value of \( N^* \) in this model. On the other hand, usually only a small number of hosts will be practical.

In the objective function, when the number of hosts \( N \) increases, the release time \( t_r \) can be reduced to maintain the same system availability because of the redundancy. The software reliability at release time point \( R(t_r) \) may also be decreased at the same time. The system availability may also increase with additional hosts. Thus, the cost of hosts \( C_h(N) \) and the reward for releasing ahead of deadline \( B(t_r) \), should increase, while the risk cost for unavailable system \( C_r(N, t_r) \), and the software development cost \( C_t(R(t_r)) \), should decrease.

### 3.2. Solution procedures

For the general optimization model, the solution procedures can be described as follows.

**Step 1.** Obtain the system availability function of the distributed system with \( N \) hosts.

**Step 2.** Obtain the cost function and the expected total cost.

**Step 3.** Let \( N \) take each integer value from 1 to \( N^* \) to obtain the expected total cost and save the results as \( C(1, t_r) \) to \( C(N^*, t_r) \), which must satisfy the constraints (14).

**Step 4.** For each expected total cost in \( C(1, t_r) \) to \( C(N^*, t_r) \), compute the optimal release time, and save the results as OpTr(1) to OpTr(\( N^* \)), so that we can get the minimal expected total cost and save them in \( \text{min}C(1) \) to \( \text{min}C(\( N^* \)) \) that \( \text{min}C(n) = C(n, \text{OpTr}(n)) \) \( (n = 1, 2, \ldots, N^*) \).

**Step 5.** Compare the minimal total mean cost from \( \text{min}C(1) \) to \( \text{min}C(\( N^* \)) \) in order to select the optimum number of hosts \( \text{OpN} = \text{min}(\text{min}C(n)) \) \( (n = 1, 2, \ldots, N^*) \) and output the results.

The above procedure can be easily performed in software such as Matlab. The number of calculations in the above procedures is of the same order as \( N^* \). It can be noted that usually the number of hosts will not be very large. An example is presented below.

### 4. Example of an application

A numerical example is presented to illustrate procedures and the solution algorithm. This application example is based on a case of telephone switching system development. The problem is first described and necessary data are presented, followed with the analysis and discussions.

#### 4.1. Description of the case

Company X was awarded a contract to develop the system for a customer. After the development, testing...
and debugging is carried out, especially on the software systems. In this case, the hardware hosts are brought from external suppliers, but the software is developed in house and tested with the system. The management is concerned about how many hosts are needed and also when the system can be released so that the total cost is minimized. For illustrative purpose, the following input values are used:

1. Customer requires the system availability to be more than 0.88 when it is released.
2. Deadline for releasing the system is about 800 hours from now on.
3. By contract, the company will be penalized for an unavailable system about $8000 per hour during the first 300 hours after release.
4. Each host cost about $17,600 and the other fee for all the hosts is about $1293, such as installation fee software copyright fee, etc.
5. Maximum number of hosts is five.
6. If the company can release the system earlier than the deadline, there is a constant reward of $2123.7 and a cumulative reward of $31.5 per hour less than the deadline.

Again, assume that each piece of hardware suffers a constant failure rate \( \lambda_h \) of an exponential distribution. There is only one version of software to be executed in \( N \) hosts and each copy of software suffers a constant failure rate \( \lambda_s(t) = K_t \lambda \) following the JM model. Here \( K_t \) is the number of faults remaining in the software at time \( t \) and \( \lambda \) is a constant. It is also assumed that a maintenance personnel exists and the repair time is exponentially distributed with parameter \( \mu_s \) for software and \( \mu_h \) for hardware that can be estimated using the historical data from previous projects and adjusted in the initial period during the test phase.

4.2. Cost analysis and results

Based on the conditions and the assumptions given in the previous section, the values of the parameters in equations (1–3) can be obtained as \( a_1 = 17,600 \), \( b_1 = 1293 \), \( a_2 = 31.5 \), \( b_2 = 2123.7 \), \( a_3 = 8000 \), \( T_d = 800 \) hours, and \( T_e = 800 + 300 = 1100 \) hours. The parameters for software development cost (4) are assumed as \( H = 10,232 \), \( B = 16 \), \( D = 14 \), which can be estimated from historical data and experiments. The optimization problem can be solved with the required system availability when releasing, \( A^* \), of 0.88 and the maximum number of hosts, \( N^* \), equals to 5.

As can be seen in figure 5, a different number of hosts leads to a different system availability function. To meet the first constraint in (14), the time for them to reach the required system availability 0.88 is different.

### Table 1. Minimal time for the system reliability reaching 0.88

<table>
<thead>
<tr>
<th>( N )</th>
<th>( 1 )</th>
<th>( 2 )</th>
<th>( 3 )</th>
<th>( 4 )</th>
<th>( 5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimal time</td>
<td>631</td>
<td>251</td>
<td>183</td>
<td>152</td>
<td>136</td>
</tr>
</tbody>
</table>

Table 1 presents the minimal release time for the case of different number of hosts.

With the values of parameters given above, we can also obtain each cost factor. The cost function of the hosts can be obtained as \( C_h(N) = 17,600N + 1293 \). The reward function for releasing the system earlier than the deadline can be expressed as \( B(t_r) = 31.5(800 - t_r) + 2123.7 \). The risk cost function for unavailable system is \( C_f(N, t_r) = 8000 \int_{t_r}^{1100} [1 - A_N(t)] dt \) and the software development cost is \( C_s(R(t_r)) = 10,232 e^{16R(t_r)-14} \).

With these cost functions, the total mean cost can be calculated as

\[
C(N, t_r) = C_h(N) + Pe(N, t_r) + C_s(R(t_r)) \\
+ C_m(\mu_s, N) + C_m(\mu_h, N) - B(t_r) \\
= 17,600N + 1293 + 8000 \int_{t_r}^{1100} [1 - A_N(t)] dt \\
+ 10,232 e^{16R(t_r)-14} - [31.5(800 - t_r) + 2123.7] \\
= 17,600N + 31.5t_r + 8000 \int_{t_r}^{1100} [1 - A_N(t)] dt \\
+ 10,233 e^{16R(t_r)-14} - 26030.7.
\]

Finally, the expected total cost as a function of release time for a different number of hosts are shown in figure 6 and the overall results are shown in table 2. The curve TMC(2, \( t_r \)) is different from other curves in figure 6, because it has not decreased to the minimum cost before the deadline of 800 hours. Since the cost of TMC(1, \( t_r \)) is too large to appear in the range of figure 6, its minimal cost result is given in table 2.

From table 2, the global minimum cost is 104,580 (units) with the number of hosts \( N = 4 \) and the optimum release time \( t_r = 261.7 \) (hours). The optimum results indicate that there should be four hosts and the system is tested for 261.7 hours. After the testing, the system can be released.

4.3. Sensitivity analysis

Since the objective function of our optimal problem is to minimize the total expected cost, the parameters of cost affect the optimal solutions significantly. A sensitivity analysis has been carried out. Here we outline the general results. The cost for each host may be influenced by the price fluctuation of server market. The reward for early release may be affected by the
changes of product market. The risk cost for unavailable system depends on the penalty determined by the customers.

The cost function for the number of hosts is expressed by equation (1). Figure 7 shows that the optimal number of hosts decreases as the function of the cost of each host and it decreases quickly at the beginning.

In general, if the reward per unit of time \( a_2 \) increases, release time \( t_r \) should be decreased. To reduce the release time while satisfying the availability requirement, the system should add more hosts. Thus, improving \( a_2 \) makes the optimal number of hosts increase.

Generally, if the risk cost parameter \( a_3 \) increases, the system availability should be improved so as to reduce the risk cost. To improve the system availability, it ought to add more hosts. Thus, improving \( a_3 \) makes the optimal number of hosts increase.

Similar studies can be carried out for other parameters. On the other hand, the optimal solution for the number of hosts is very robust to the cost parameters. That is, within a certain range of value, the number of hosts will remain the same because of the discrete nature.

5. Discussions

In this research, an optimization model is presented for a decision problem considering the number of hosts and release time in a multihosts distributed system design environment. This is an important question in system analysis and planning as the cost is usually a major concern. In our model, both software and hardware failures are taken into consideration. The optimization model is based on the general cost criteria and the objective function is to minimize the expected total cost. The main emphasis is on the use of software reliability growth model in this type of analysis, which leads time-dependent component reliability. A solution procedure is presented and an example is given to show how the model is used in an application example.

In this research, specific cost functions are used, although the approach is a general one. A possible further research is to refine the cost functions by incorporating other cost factors or using other cost models. It is expected that different applications call for the use of different cost models. Our cost functions are appropriate at least for the application example presented.

It is also possible to use other software reliability growth models for the modelling of software failure process. Note that the JM model used is very similar, from a statistical point view, to the non-homogeneous Poisson process model of Goel–Okumoto. However, other models could be more appropriate (Ansell 1999, Pham 2001). On the other hand, since models are an approximation of reality and here the common problem is to obtain an accurate parameter estimates, unless it is justifiable and needed, simple models will be sufficient.

Another possibility for further research is that we could, using the data collected during the testing period, dynamically adjust the values of parameters and the optimal solutions. This may lead to a revised decision and hence it may cause other problems such

![Figure 6. Total mean cost versus release time for a different number of hosts.](image)

<table>
<thead>
<tr>
<th>( N )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{minC}(N) )</td>
<td>326970</td>
<td>153060</td>
<td>116690</td>
<td>104580</td>
<td>110880</td>
</tr>
<tr>
<td>( \text{OpTr}(N) )</td>
<td>800</td>
<td>800</td>
<td>324.2</td>
<td>261.7</td>
<td>232</td>
</tr>
</tbody>
</table>

Table 2. Numerical values of the minimum cost for different \( N \)
The optimal number of redundant hosts

Figure 7. Optimal number of hosts versus the cost of each host, $a_1$.

as how to reduce or add hosts. It is possible to revise the release time dynamically through the data collected during the testing period and this poses another interesting problem for future study.

Acknowledgement

The authors thank the anonymous referees for some insightful comments on an early version of the paper.

Part of this paper is adapted with permission from a recent monograph (Computing Systems Reliability, Kluwer Academic/Plenum Publishers). Our research is supported by a research grant (R-266-000-020-112) at the National University of Singapore.

References


