Reliability Improvement of the Computing System Through Matrix Partitioning

Anju Khandelwal

Department of Mathematics
S.R.M.S. Women's College of Engineering and Technology, Bareilly, India
E-mail : dranjukhandelwal@yahoo.co.in, dranjukhandelwal@rediffmail.com

(Received September 22, 2008)

Abstract: Improving the performance of the computing systems is a major and challenging problem for the researchers in this wide area of research. It is almost impossible that the computing system has to execute only as many tasks as the number of processors available in the system. That means the number of tasks, which are to be executed on the computing systems, shall be the more as compared to the number of processors in the systems. The problem of execution of "m" tasks to "n" processors (m > n) on a computing system is addressed here through an efficient task allocation policy for the task execution on a computing system. The present paper is based on the consideration of Execution Reliability and Communication Reliability of the tasks to the processors. The execution reliabilities of the tasks on different processors have been taken into consideration while preparing the algorithm to such a case. This research paper contains the computational algorithm of the approach and its implementation. This problem is capable to deal all such real life situations, where the tasks are more than the number of processors. Further this approach can be extended to achieve the true optimal results. The developed algorithm is coded in C++ and implemented on the several sets of input data, to test the effectiveness and efficiency of the algorithm.

Keywords: Distributed systems, assignments, execution reliability, communication reliability.

2000 Mathematics Subject Classification No.: **90C08**

1. Introduction

Reliability analysis for any Distributed Computing System is the current necessity and most important area of research. There are number of ways to improve the reliability of the distributed computing system. Optimizing the reliability through task allocation is one of the problems in this category. Some heuristics techniques may also provide some good solutions. Sometimes correctly processing of the task by any processor is on top priority irrespective of time or cost constraints, that is, complete processing of task is necessary to achieve a goal where other constraints are secondary. The literature survey reveals that only few researchers devoted their research work in this area so that this area has to be taken into consideration. Kuo et al[1] presented an annotated overview of system reliability optimization in which he has suggested heuristics, meta-heuristics algorithm, exact method for reliability

redundancy allocation, multi objective optimization and also assignment of interchangeable components in reliability systems. The exact solutions regarding such problem are not necessarily desirable because exact solutions are very difficult and even when they exist there utility is on boundary line. Therefore the majority of research work in this area is devoted to heuristic and meta-heuristic solutions. The other two similar algorithms have been given by Peng, Shin and Abdel[2], Sagar and Sarje[3] and B. Shirazi, M. Wang and G. Pathak[4]. Peng, Shin and Abdel[2] used matrix reduction technique. According to the criteria given therein, a task is selected randomly to startwith and then assigned to a processor.

2. Assumptions

The completion of a program from computational point of view means that all related tasks have got executed and the final output has been generated after integrating the respective outputs of individual programs.

1. The number of tasks to be allotted is more than the number of processors.
2. Some of the tasks have restrictions for being allocated on some particular processor(s) because of the non-availability of desired facilities, which may include access to particular peripheral device, and high-speed arithmetic capability of the processor. The execution reliability of such tasks is taken to be zero, on those processors where these tasks cannot be executed.
3. Whenever two or more tasks are assigned to the same processor, the communication cost between them is assumed to be zero.

3. Problem Statement

Let the given system consists of a set of n processors $P = \{p_1, p_2, ..., p_n\}$, and a set of m tasks $T = \{t_1, t_2, ..., t_m\}$. The present problem is based on the consideration of Execution Reliability and Communication Reliability of the tasks to the processors. The Execution Reliability of individual tasks corresponding to each processor is given in the form of matrix ERM(,) of order $m \times n$ and Communication Reliability is taken in the square matrix CCM(,) of order m respectively. A procedure to assign the tasks to the processors of the Distributed Computing systems is to be designed in such a way that the overall reliability is to be optimizing under the pre-specified constraints and none of the tasks get unexecuted. The reliability functions to measure ER and CR are then formulated. Further function to obtain Ereliability has been derived for the said purpose. The main objective of this problem is to maximize the total program execution reliability by allocating the tasks optimally. Keeping in view we suggested a modified method to assign all the tasks as per the required availability of processors so that none of the tasks get remains unexecuted and the present approach does not require to add dummy processors.

4. Proposed method

Let the given system consists of a set of n processors $P = \{p_1, p_2, ..., p_n\}$, and a set of m tasks $T = \{t_1, t_2, ..., t_m\}$. The present problem is based on the consideration of Execution Reliability and Communication Reliability of the tasks to the processors. The Execution Reliability of individual tasks corresponding to each processor is given in the

form of matrix ERM(,) of order $m \times n$ and Communication Reliability is taken in the square matrix CCM(,) of order m respectively. Since the number of tasks are more than the number of processors, so that we divide the problem into sub balance problems. First of all obtain the product of each row and each column except the position where the reliability is zero (zero reliability should be kept aside with the product of row or column) from the ERM(,) and store the results into Product_Row( ) and Product_Column( ), each of them are one dimensional arrays. Select the first set of tasks, (this set of tasks shall contain only as many tasks as the number of processors) on the basis of maximum reliability against the tasks in the Product_Row() array. Store the result into ERM (, ,) a two dimensional array. Repeat the process until remaining tasks are either less than or equal to the number of processors. If the tasks are equal to the processors then it will become the last sub problem, else to form the last problem we have to delete the column (processor) from ERM(, ,) on the basis of Product_Column() array i.e. this set shall contain only as many processors as number of tasks left, so that we delete the processors having lesser reliability in the Product_Column() array. Convert matrices of each sub problem into execution unreliability matrix namely, EURM(, ,) by subtracting each element of ERM(, ,) from one. For allocation purpose a modified version of row and column assignment method devised by Kumar et al[5] is employed which allocates a task to a processor where it has minimum execution unreliability. The communication unreliability of those tasks, which are allocated on the same processor, becomes one. For each sub problem we calculate the exaction reliability of each processor and store the result in a linear array PER(j) and also communication reliability PCR(j) of the distributed computing system, where j= 1,2,…n. The overall assignment reliability [Ereliability] is expressed as products of the execution reliabilities and communication reliabilities of all the tasks as follows:

$$P\text{ER(j)} = \prod_{i=1}^{n} \{ \sum_{j=1}^{n} er_{ij} \, x_{ij} \} \; ;$$

$$P\text{CR(j)} = \prod_{i=1}^{m} \{ \sum_{j=1}^{n} cr_{ij} \; y_{ij} \}$$

$$\text{Ereliability} = \left[\prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} \; er_{ij} x_{ij} \right\} * \prod_{i=1}^{n} \left\{ \sum_{j=1}^{n} \; cr_{ij} y_{ij} \right\} \right]$$

where

$$y_{ij} = \begin{cases} 1, \text{ if } i^{\text{th}} \text{ task is assigned to } j^{\text{th}} \text{ processor} \\ 0, \text{ otherwise} \end{cases}$$

and

$$y_{ij} = \begin{cases} 1, \text{ if the task assigned to processor i communicate} \\ \qquad\qquad\qquad \text{with the task assigned to processor } j \\ 0, \text{ otherwise} \end{cases}$$

5. Computational Algorithm

The method discussed in the problem is to determine the task allocations in distributed processing environment depend upon the following components:

1. Determine the initial allocation

2. Determine the final allocation
3. Compute the total optimal system reliability.

6. Algorithm

To give an algorithmic representation to the technique mentioned in the previous section, let us consider a system in which a set of m tasks $T = \{t_1, t_2, ..., t_m\}$ is to be executed on a set of n available processors $P = \{p_1, p_2, ..., p_n\}$.

Step-1:

Input: m, n. ERM (,), CRM(,)

Step-2:

Obtain the product of each row of the ERM(,) in such a way that, if any reliability(ies) is (are) zero then keeping it aside along with sum of that row (just to avoid the condition sum ER * 0 = 0). Store the results in one-dimensional array Product _Row (,) of order m.

Step-3:

Obtain the product of each column of the ERM (,), in such a way that if any reliability (ies) is (are) zero then keeping it aside along with product of that column (just to avoid the condition ER * 0 = 0). Store the results in one-dimensional array Product _Column (,) of order n.

Step-4:

Partition the execution reliability matrix ERM (,) of order m x n to sub matrices such that the order of these matrices become square i.e. number of row should be equal to number of column. Partitioning to be made as mentioned in the following steps.

Step-4.1:

Select the n tasks on the basis of Product _Row (,) array i.e. select the 'n' tasks corresponding to most maximum product to next maximum product, if there is a tie select arbitrarily. (For the cases in which product is ER * 0 = 0, maximum value depends only on ER and the impact of zero is to be neglected).

Step-4.2:

Store the result in the two dimensional array ERM (, ,) to form the sub matrices of the sub problems.

Step-4.3:

If all the tasks are selected then go to step 4.7 else steps 4.4

Step-4.4:

Repeat the step 4.1 to 4.3 until the number of tasks become less than n.

Step-4.5:

Select the remaining task say r, r < n, select the r processors on the basis of Product_ Column (,) array i.e. the processors corresponding to the most maximum product to next maximum, if there is a tie select

arbitrarily (for the cases in which product is ER * 0 = 0, maximum value depends only on ER and the impact of zero is to be neglected).

Step-4.6:

Store the result in the two dimensional array ERM (, ,), which is a last sub problem.

Step-4.7:

List of all the sub problems formed through Step 4.1 to 4.6 and repeat step 5 to step 13 to solve each of these sub problems.

Step-4.8:

Convert all execution reliability matrices of each sub problem into the execution unreliability matrices by subtracting each entry by one in all execution reliability matrices.

Step-5:

Find the minimum of each row of NEURMI (, ,), and replace it by 0.

Step-6:

Find the minimum of each column of NEURM(, ,), and replace it by 0.

Step-7:

Search for a row in NEURM, which has only one zero and assign the task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position.

Step-8:

Search for a column in NEURM, which has only one zero and assign the task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position.

Step-9:

Check whether nar = n if not then pickup an arbitrary 0 and assign task(s) corresponding to this position. Add one to the counter that is nar = nar + 1 and also store this position, else, Check column (s) position of 0's in unassigned row(s). Check the row(s) for any previous assignment in the corresponding column(s). Find the minimum of the entire elements for the remaining rows and replace it zero.

Step-10:

Evaluate Execution Reliability.

Step-11:

Evaluate Communication Reliability.

Step-12:

Execution Reliability (Ereliability) are thus calculated as:

Ereliability = ER * CR

Step-13:

Stop.

7. Implementations

Example-I

Consider a system consisting of a set $T = \{t_1, t_2, t_3, t_4,\}$ of 4 tasks and a set $P = \{p_1, p_2, p_3\}$ of 3 processors,

Step-1: Input: 4, 3

$$ERM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.997 & 0.996 & 0.994 \\ 0.993 & 0.998 & 0.992 \\ 0.998 & 0.991 & 0.994 \\ 0.997 & 0.993 & 0.998 \end{array};$$

$$CRM(,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \\ 0.000 & 0.994 & 0.996 & 0.995 \\ 0.994 & 0.000 & 0.992 & 0.993 \\ 0.996 & 0.992 & 0.000 & 0.992 \\ 0.995 & 0.993 & 0.992 & 0.000 \end{array}$$

Step-2:

Obtain the product of each row and column of ERM (,), i .e. the products of each row and each column are as:

$$\text{Product\_Row}() = \begin{array}{cccc} t_1 & t_2 & t_3 & t_4 \\ 0.9870539 & 0.9830858 & 0.9830838 & 0.9880409 \end{array}$$

$$\text{Product\_Column}() = \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.9850768 & 0.9781664 & 0.9781714 \end{array}$$

Step-3:

We partitioned the matrix ERM (,) to define the first sub problem ERMI (, ,) by selecting rows corresponding t_1, t_2, t_4 on the basis of Product_Row() array and on the basis of the Product_Column() array, by deleting columns corresponding to p_2, p_3 then after selecting the remaining one task t_3 to form the last sub problem ERMII (, ,), as there was only one task, for which we required only one processor. So that the modified matrices are as;

Sub Problem-I:

$$ERMI(,,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_4 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.997 & 0.996 & 0.994 \\ 0.993 & 0.998 & 0.992 \\ 0.997 & 0.993 & 0.998 \end{array}$$

and Sub Problem-II:

$$ERMII(,,)\begin{matrix} & p_1 \\ t_3 & 0.998 \end{matrix}$$

Obtain the unreliability matrices for ERMI (,,) and ERMII (,,) as,

$$EURMI(,,)=\begin{matrix} & p_1 & p_2 & p_3 \\ t_1 & 0.003 & 0.004 & 0.006 \\ t_2 & 0.007 & 0.002 & 0.008 \\ t_4 & 0.003 & 0.007 & 0.002 \end{matrix}$$

$$EURMII(,,)\begin{matrix} & p_1 \\ t_3 & 0.002 \end{matrix}$$

Step-4 & 5:

On applying modified Hungarian method devised by Kumar et al[5] to assign the tasks, on the basis of min $\{r_i\}$ and min $\{c_j\}$ from unreliability matrices for every i and j. We put $r_{ij} = 0$ and $c_{ij} = 0$, for every *i* and *j*. On applying this to all sub problems, the modified matrices for each sub problem are mentioned below:

$$EURMI(,,)=\begin{matrix} & p_1 & p_2 & p_3 \\ t_1 & 0.000 & 0.004 & 0.006 \\ t_2 & 0.007 & 0.000 & 0.008 \\ t_4 & 0.003 & 0.007 & 0.000 \end{matrix}$$

$$EURMII(,,)\begin{matrix} & p_1 \\ t_3 & 0.000 \end{matrix}$$

Step-6, 7&8:

After implementing assignment process, the solution set of the each sub problem, the allocation is thus obtained.

Solution for the Sub Problem-I:

Tasks	→	Pr*ocessors*	*ER*
t_1	→	p_1	0.997
t_2	→	p_2	0.998
t_4	→	p_3	0.998

Solution for the Sub Problem-II:

Tasks	→	Pr*ocessors*	*ER*
t_3	→	p_1	0.998

Step-9:

After implementing the process, we obtain the following set of complete assignments alongwith execution reliabilities of each processor.

$Tasks$	$\rightarrow$	$Processors$	ER
t_1	$\rightarrow$	p_1	0.997
t_2	$\rightarrow$	p_2	0.998
t_3	$\rightarrow$	p_1	0.998
t_4	$\rightarrow$	p_3	0.998

Step-10:

ER : = 0.99102990

CR : = 0.96645590

Ereliability : = 0.95778660

Step-11:

Stop.

Example-II

Now consider a system which consists of a set T = {t_1, t_2, t_3, t_4, t_5} of 5 tasks and a set P = {p_1, p_2, p_3} of 3 processors, where,

Step-1: Input: 5, 3

$$ERM(,) = \begin{array}{c|ccc} & p_1 & p_2 & p_3 \\ t_1 & 0.997 & 0.996 & 0.994 \\ t_2 & 0.993 & 0.998 & 0.992 \\ t_3 & 0.998 & 0.991 & 0.994 \\ t_4 & 0.997 & 0.993 & 0.998 \\ t_5 & 0.992 & 0.995 & 0.996 \end{array}$$

$$CRM(,) = \begin{array}{c|ccccc} & t_1 & t_2 & t_3 & t_4 & t_5 \\ t_1 & 0.000 & 0.994 & 0.996 & 0.995 & 0.993 \\ t_2 & 0.994 & 0.000 & 0.992 & 0.993 & 0.995 \\ t_3 & 0.996 & 0.992 & 0.000 & 0.992 & 0.996 \\ t_4 & 0.995 & 0.993 & 0.992 & 0.000 & 0.992 \\ t_5 & 0.993 & 0.995 & 0.996 & 0.992 & 0.000 \end{array}$$

Step-2:

Obtain the product of each row and column of ERM (,), i .e.

$$\text{Product\_Row} = \begin{array}{ccccc} t_1 & t_2 & t_3 & t_4 & t_5 \\ 0.9870539 & 0.9830858 & 0.9830838 & 0.9880409 & 0.9830918 \end{array}$$

$$\text{Product\_Column} = \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.9771962 & 0.9732756 & 0.9742587 \end{array}$$

Step-3:

We partitioned the matrix ERM (,) to define the first sub problem ERMI (,) by selecting rows corresponding to t_1, t_4, t_5 and second sub problem ERMII (,) by selecting rows corresponding to the tasks t_2, t_3 and by deleting columns corresponding to p_2. So that the modified matrices for each sub problems are as;

Sub Problem-I:

$$ERMI(,,) = \begin{array}{c} \\ t_1 \\ t_4 \\ t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.997 & 0.996 & 0.994 \\ 0.997 & 0.993 & 0.998 \\ 0.992 & 0.995 & 0.996 \end{array}$$

and Sub Problem-II:

$$ERMII(,,) = \begin{array}{c} \\ t_2 \\ t_3 \end{array} \begin{array}{cc} p_1 & p_3 \\ 0.993 & 0.992 \\ 0.998 & 0.994 \end{array}$$

Obtain the unreliability matrices for the sub problems ERMI(,) and ERMII(,) as,

$$EURMI(,,) = \begin{array}{c} \\ t_1 \\ t_4 \\ t_5 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.003 & 0.004 & 0.006 \\ 0.003 & 0.007 & 0.002 \\ 0.008 & 0.005 & 0.004 \end{array}$$

$$EURMII(,,) = \begin{array}{c} \\ t_2 \\ t_3 \end{array} \begin{array}{cc} p_1 & p_3 \\ 0.007 & 0.008 \\ 0.002 & 0.006 \end{array}$$

Step-4 & 5:

On applying modified Hungarian method devised by Kumar et al[5] to assign the tasks, on the basis of min $\{r_i\}$ and min $\{c_j\}$ from unreliability matrices for every i and j. We put $r_{ij} = 0$ and $c_{ij} = 0$, for every i and j. On applying this to all sub problems, the modified matrices for each sub problem are mentioned below:

$$EURMI(,,) = \begin{array}{c} \\ t_1 \\ t_2 \\ t_4 \end{array} \begin{array}{ccc} p_1 & p_2 & p_3 \\ 0.000 & 0.000 & 0.006 \\ 0.003 & 0.007 & 0.000 \\ 0.008 & 0.005 & 0.000 \end{array}$$

$$EURMII(,,) = \begin{array}{c} \\ t_2 \\ t_3 \end{array} \begin{array}{cc} p_1 & p_3 \\ 0.000 & 0.008 \\ 0.000 & 0.000 \end{array}$$

Step-6, 7&8:

After implementing assignment process, the allocation is thus obtained.

Tasks	→	Pr*ocessors*	*ER*
t_1	→	p_2	0.996
t_2	→	p_1	0.993
t_3	→	p_3	0.994
t_4	→	p_1	0.997
t_5	→	p_3	0.996

Step-9:

ER : = 0.9762239

CR : = 0.9501149

Ereliability : = 0.9275249

Step-10:

Stop.

8. Conclusions

The algorithm presented in this paper is capable of maximizing the overall reliability of Distributed Computing System through task allocation. In distributed computing system, tasks are allocated in such a way that their individual reliability of processing is optimized as well as the overall reliability is improved. In this approach not only that the loads of each processor get equally balanced and none of the tasks gets unprocessed so that requirement for adding dummy processor becomes meaningless in our approach. Our approach provides an optimal solution for assigning a set of "m" tasks of a program to a set of "n" processors, where m > n in a computer communication systems that maximizes the overall reliability of the system and the load of all allocated tasks on all the processors equally balanced. For the example-I, the execution reliability on different processors has been obtained. Also the communication reliability of the computer communication system is mentioned in the following table:

Processors of the Distributed Computing Systems	p_1	p_2	p_3
Execution reliability of each processor PER (,)	0.995006	0.998000	0.998000
Communication reliability of DCS	0.96645590		
Total reliability of DCS [Ereliability(,)]	0.95778660		

The final results of example-I are as:

Tasks	→	Pr*ocessors*	*ER*	*CR*	*Ereliability*
$t_1 * t_3$	→	p_1			
t_2	→	p_2	0.99102990	0.96645590	0.95778660
t_4	→	p_3			

For the example-II, the execution reliability on different processors has been obtained. Also the communication reliability of the distributed computing system is mentioned in the following table:

Processors of the Distributed Computing Systems	p_1	p_2	p_3
Execution reliability of each processor PER (,)	0.990024	0.996000	0.990021
Communication reliability of DCS	0.9501149		
Total reliability of DCS [Ereliability(,)]	0.9275249		

The final results of Example-II are as:

Tasks	$\rightarrow$	Pr*ocessors*	*ER*	*CR*	*Ereliability*
t_1	$\rightarrow$	p_2			
$t_3 * t_5$	$\rightarrow$	p_3	0.9762239	0.9501149	0.9275249
$t_2 * t_4$	$\rightarrow$	p_1			

The method is presented in computational algorithmic form and implemented on the several sets of input data to test the performance and effectiveness of the algorithm. The problem discussed in this paper, would be useful to the network system designer working in the field of distributed computing systems. The developed method is programmed in C++ and implemented the several sets of input data to test the effectiveness and efficiency of the algorithm. It is recorded that the method is suitable for arbitrary number of processors with the random program structure.

The present algorithm mentioned in this paper provides the optimal solution for improving the reliability. This problem deals with the performance on the bases of maximizing reliability. However the future scopes for the further researcher is that some model (s) can be developed which shall be capable to improve the performance of the distributed computing systems.

References

1. Way Kuo and V. Rajendra Prasad, An Annotated Overview of System – Reliability Optimization, *IEEE Transactions on Reliability*, **49-2** (2000) 176-187.
2. Dar-Tezen Peng, K. G. Shin and Zoher T. F. Abdel, Assignment Scheduling Communication Periodic Tasks in Distributed Real Time System, *IEEE Transactions on Software Engineering*, SE-**13** (1997) 745-757.
3. B. Shirazi, M. Wang and G. Pathak, Analysis and Evaluation of Heuristic methods for Static Task Scheduling, *Journal on Parallel and Distributed Computing*, **10** (1990) 222-223.
4. G. Sagar and A. K. Sarje, Task Allocation Model for Distributed System, *International Journal of Systems Sciences*, **22-9** (1991) 1671-1678.
5. V. Kumar, M. P. Singh and P. K. Yadav, A Fast Algorithm for Allocating Task in Distributed Processing System, *Proceedings of the 30th Annual Convention of Computer Society of India, Hydrabad (AP), India during Nov.* 8-11, 1995, 347-358.

6. Cheol - Hoon Lee, Kang G. Shin, "Optimal Task Assignment in Homogeneous Networks," *IEEE Transactions on Parallel and Distributed Systems*, **8-2** (1997) 119-129.
7. A.O. Charles Elegbede, Chengbin Chu, Kando H. Adjallah and Faronk Yalaoui, Reliability Allocation through Cost Minimization, *IEEE Transactions on Reliability*, **52-1**, (2003) 106-111.
8. Avanish Kumar, Optimizing for the Dynamic Task Allocation, *Proceedings of the III Conference of the International Academy of the Physical Sciences held at Allahabad*, 1999, 281-294.