Joint Conference of 1996 International Computer Symposium
December 19~21, Kaohsiung, Taiwan, R.O.C.

A Simple File Assignment Method to Maximize the System Reliability in Distributed Computing Systems under Memory Space Constraints

Deng-Jyi Chen, Ching-Yun Lee, Tz-Shin Cheng

Institute of Computer Science and Information Engineering
National Chiao-Tung University, Hsinchu, Taiwan, R.O.C.

Abstract

Distributed computing systems (DCS) have become a major trend in today's computer system design due to their capability for offering high speed and high reliable performance advantages. Reliability is an important issue in DCS design for military application. The distribution of programs and data files are two important factors that affect the program reliability and system reliability. The reliability-oriented file assignment problem is to find a file distribution such that the program reliability or system reliability is maximal. In this paper, we develop a simple heuristic file assignment algorithm which use several simple heuristic assignment rules to achieve reliability-oriented file assignment. The proposed algorithm can obtain the optimal solutions in most cases and reduce computation time significantly.

1. Introduction

A distributed computing system is a collection of processor-memory pairs connected by a communication subnet and logically integrated by a distributed operating system and/or a distributed database system [1,2,3]. Potential advantages of a DCS are significant, including good performance, reliability, resource sharing, and extendibility. Performance enhancement is due to the use of multiple processors and efficient subnets. Reliability improvement is due to the use of redundant techniques in data files, programs, processes, and communication devices. A program in a DCS may require one or more data files located in different computers for execution. The distribution of data files can affect program reliability and overall system reliability. An important problem in DCS design is to find a file distribution such that a certain reliability measure is maximal. Several network reliability measures have been defined and associated evaluation methods have been developed. The concepts of distributed program reliability (DPR) and distributed system reliability (DSR) in [4,5] and the reliability evaluation algorithm in [13,14] are adopted in this paper.

The file assignment problem and related programs such as task assignment and job scheduling have been studied for many years [6,7,8,9]. Since the file assignment problem is NP-complete [10], Wang [11] proposed a reliability-oriented file assignment algorithm to solve the optimal file assignment problem under a memory space constraint. This method is capable of finding the optimal solution, but it is not efficiently. Hwang & Tseng proposed a heuristic task assignment algorithm for the k-DTA problem to maximize reliability of a distributed system under some resource constraints [12]. The k-DTA algorithm assigns fixed number of copies of programs and data files only, so the system utilization is not maximized. In addition, the algorithm is quite complex. This paper presents a simple but effective heuristic algorithm for reliability-oriented file assignment problem. The Simple File Assignment algorithm use a special Minimum Selection Rule and Grouping Rule based upon some heuristics [11,12] to find a file distribution such that the DPR (or DSR) is maximal and the memory limitation constraint of each processing element is satisfied. Numerical results are given through computer simulation to show the solution of our proposed algorithms.

2. Notation, Definitions, and Problem statements

Notation & Acronyms:

G(V,E)	An undirected graph; V, E: set of [vertices, edges/links] in the graph
N_i	node i in V
L_{ij}	link between N_i and N_j
PRG	the set of programs allocated in the network for execution
F_s	the set of files required by PRG
PRG_p	program p in PRG
F_i	file i in F_s

n	number of nodes in G; n= | V |
k	number of files in F_s
l	number of programs in PRG
p(q)	probability that the communication link works (fails)
PA_i	The set of programs allocated on N_i
FA_i	The set of files allocated on N_i
$E(PRG_p)$	event that PRG_p can successfully run and files can be successfully accessed by PRG_p
Pr(E)	probability of event E
ROFA	Reliability-Oriented File Assignment
SFA	Simple File Assignment algorithm

Definitions:

• FST: a file spanning tree that connects the root node (the processing element that runs the program under consideration) to some other nodes such that its vertices hold all the needed files.

• MFST: a minimal FST such that there exists no other FST which is subset of it.

• DPR: The probability that a given program can be run successfully and will be able to access all the files it requires from remote sites in spite of faults occurring among the processing elements and communication links. The MFSTs connect the root node (the PE that runs the program under consideration) to other nodes such that these nodes hold all the files needed for the program under execution. The DPR and DSR can then be determined by computing the probability that at least one of these MFSTs is working. Thus the distributed program reliability for a given program j can be defined as the probability that at least one MFST of a given program j is operational [4,5]. This can be written as

$$\text{DPR} = \Pr\ (\bigcup_{j=1}^{n_{mfst}} MFST_j)$$

where n_{mfst} is the number of MFSTs that run the given program.

• DSR: The probability that all the programs in the system can be run successfully. The DPR measures the reliability of a particular distributed program. For the entire DCS to be operational, several such programs or a given set of distributed programs must be operational. A system level reliability measure for l distributed programs to be operational is defined in [15] as

$$\text{DSR} = \Pr\ (\bigcap_{i=1}^{l} PRG_i)$$

• MFFC: An MFFC is a maximal feasible file combination such that there exists no other feasible file combination which is a superset of it. If a combination $X_i^{(t)} = (x_{i1}^{(t)}, x_{i2}^{(t)}, \ldots, x_{ik}^{(t)})$ is said to be a feasible file combination of node N_i, then

$$\sum_{j=1}^{k} s_j x_{ij} \le C_i$$

where C_i = the available memory space of node i (N_i)

s_j = size of file j (F_j)

x_{ij} = the indicator of file assignment; $x_{ij} = 1$ if F_j is assigned to N_i, otherwise $x_{ij} = 0$

• Node Environment Weight Heuristics: Node X_1 is more reliable than node X_2 if and only if the degree of X_1 is higher than that of X_2 [12]. It is because the node with higher degree is more likely to have more paths to the destination nodes than those with lower degrees. The node environment weight represents the composite reliability for the nodes and links surrounding a node.

ENV_WEIGHT(N_i) ≡

$$R(N_i) \cdot \sum_{N_j \in ADJ(N_i)} R(N_j) \cdot R(L_{ij})$$

where $ADJ(N_i)$ = set of nodes which are adjacent to N_i

$R(N_i) = \Pr\{N_i \text{ is operational}\}$

$R(L_{ij}) = \Pr\{L_{ij} \text{ is operational}\}$

• Program Weakness Heuristics: Program P_1 is weaker than program P_2 if and only if the minimum number of nodes required to assign P_1 and its associated files are greater than those required for P_2 [12]. We simply use the total memory of P_i and its associated files to approximate the number of nodes required. The weakness decision function is:

$$\text{WEAKNESS}(P_i) \equiv \text{SIZE}(P_i) + \sum_{F_j \in FN_i} SIZE(F_j)$$

Problem statements:

The reliability-oriented file assignment problem can be mathematically stated as follows:

Problem - Maximizing DSR subject to memory space constraint

$$\text{Maximize DSR} = \Pr\ [\bigcap_{i=1}^{l} E(PRG_i)],\quad \text{subject to}$$

$$\sum_{j=1}^{l} SIZE(P)_j X_{ij} + \sum_{j=1}^{k} SIZE(F_j) Y_{ij} \le C_i,\ \forall\ N_i,\ i = 1,2,\ldots,n$$

$$\sum_{i=1}^{n} X_{ij} \ge 1 \qquad j = 1,2,\ldots,l$$

$$\sum_{i=1}^{n} Y_{ij} \ge 1 \qquad j = 1,2,\ldots,k$$

$X_{ij} = 1$ if P_j is allocated on N_i, otherwise $X_{ij} = 0$

$Y_{ij} = 1$ if F_j is allocated on N_i, otherwise $Y_{ij} = 0$

For the reliability-oriented file assignment problem, we give:

a) network topology

b) files required by each program for execution
c) the size of each program and data-file
d) the available memory space of each processing element (computer node)
e) the reliability of each node and communication link

This paper is concerned with the file assignment to maximize the system reliability of the distributed computing system under memory space constraints.

3. The Simple File Assignment algorithm

This section presents a Simple File Assignment (SFA) algorithm which use the MFFC concept, program weakness and node environment weight heuristic measures. We also simplify the assignment method by introducing two new concepts called the Minimum Selection Rule and Grouping Rule. The combined effort makes our new algorithm simple and efficient.

3.1 Minimum Selection Rule (MSR)

From the definition of the MFFC, we know that the MFFCs give us the best reliability for each node. For a file assignment problem with size of each program and files given, there are several MFFC combinations for a single node with limited storage capacity. The basic idea for the SFA algorithm is using MSR to pick up just one or two good MFFCs for each node. The MSR algorithm is simple with reducing computation time very much. This heuristic reduction makes original intractable problems solvable and the reliability deviation from the optimal solution is also tolerable by simulation results.

Example 1: Suppose we know that programs P_1, P_2 need file $\{F_1, F_2\}$ and $\{F_1, F_3\}$ to execute successfully. The size of node N_1 is 3 and the sizes of program P_1, P_2 and file F_1, F_2, F_3 are 1, 2, 1, 2 and 3, respectively (i.e., $SIZE(P_i) = i$ and $SIZE(F_i) = i$). We can compute all the MFFCs for node N_1 and show the result in table 1.

Table 1: All the MFFCs for node N_1 of Example 1

Node N_1 (MFFCs)	P_1	P_2
$a_1=\{P_1,P_2\}$	$F_1+F_2=3$	$F_1+F_3=4$
$b_1=\{P_1,F_1\}$	$F_2=2$	$P_2+F_3=5$
$c_1=\{P_2,F_1\}$	$P_1+F_2=3$	***$F_3=3$***
$d_1=\{P_1,F_2\}$	***$F_1=1$***	$P_2+F_1+F_3=6$
$e_1=\{F_1,F_2\}$	***$P_1=1$***	$P_2+F_3=5$
$f_1=\{F_3\}$	$P_1+F_1+F_2=4$	***$P_2+F_1=3$***

Assume we wish to favor program P_1 for node N_1 (it is decided by program weakness and node environment weight heuristics), we sum up the sizes of all the lacking components of set $\{P_1, F_1, F_2\}$ for each MFFC. We do this because $\{P_1, F_1, F_2\}$ is the set of complete requirement for successful execution of program P_1. If we assign MFFC $a_1=\{P_1, P_2\}$ to this DCS, the other nodes of this DCS must supply the remaining F_1 and F_2 for P_1 to run successfully. It goes without saying that $d_1=\{P_1, F_2\}$ is more favorable than $a_1=\{P_1, P_2\}$ for P_1 because d_1 is more self-sufficient (i.e., d_1 lacks F_1 only, while a_1 lacks both F_1 and F_2).

From table 1, we can see (in bold & italic face) MFFC $d_1=\{P_1, F_2\}$ and $e_1=\{F_1, F_2\}$ are indistinguishable to Minimum Selection Rule because the weight of their lacking component is identical. However, MSR has successfully reduced the number of possible MFFCs of node N_1 from 6 to 2, a three-fold speed up just for a single node N_1. The combined effort of MSR for every node is very significant.

3.2 Grouping Rule (GR)

Before we go into the details about the Grouping Rule, let's observe some optimal file assignment cases.

Example 2: In Figure 1, the optimal solutions are the sets whose union of node N_2's MFFC and node N_3's MFFC is the full set $\{P_1, P_2, F_1, F_2, F_3\}$ of the DCS. The node N_1's MFFC is irrelevant, however.

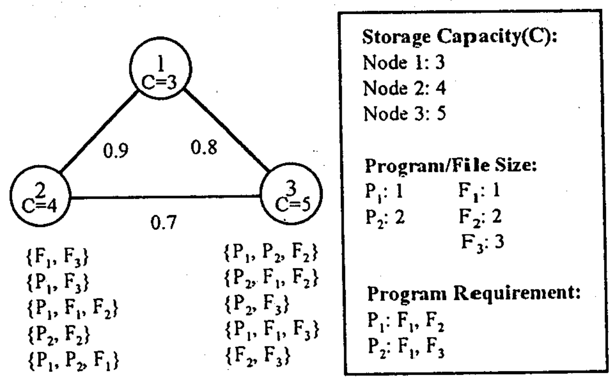


Figure 1. The optimal file assignments of Example 2

The definition of the Full Set is: *a set which contains all the programs and their required data files in the DCS.* Node 2 and node 3 in the example 2 obviously contain a single full set $\{P_1, P_2, F_1, F_2, F_3\}$. Full set is the basic requirement of the DSR. Let's recall the definition of the minimum file spanning tree. The MFST for all programs must contain all the elements in the full set. If not, at least one of the programs will fail so the DSR value is zero. In addition, denote the set of MFSTs for an assignment ASS(S,G) of a dependent set by MFST(S,G). If there exists another assignment ASS(S-{v},G) where v is a terminal node of some MFST in MFST(S,G), then DSR(S,G) < DSR(S-{v},G). This theorem is obvious from the definition of DSR.

Let's examine the result of example 2. The optimal assignment is the spanning of all programs and their files on the node 2 and node 3 in the full set. From previous theorem we know this assignment is better than other assignments which full set must span all the three nodes. But how do we make a choice between two or more file assignments with the same maximum number of full sets? Figure 2 is an optimal file assignment to this DCS and Figure 3 is a poor one. They differ only in the file assignments. This DCS has several groups of nodes which contain the full set $\{P_1, P_2, P_3, F_1, F_2, F_3, F_4\}$.

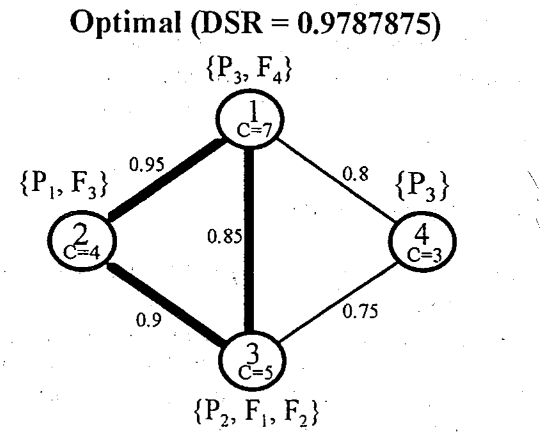


Figure 2. The optimal file assignment of Example 4

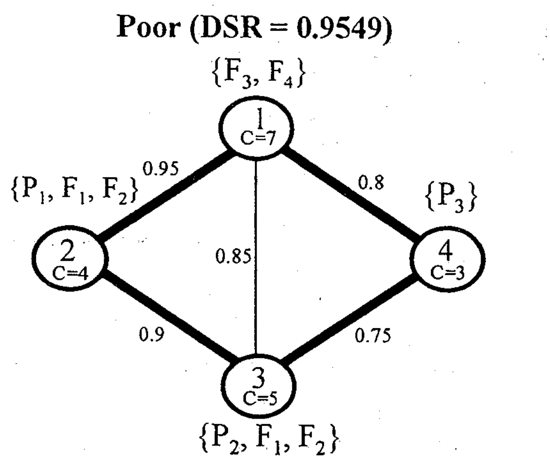


Figure 3. The poor file assignment of Example 4

Figure 2 and 3 is an example to show how file assignments affect the distributed system reliability. First we know that the file assignment which contains more groups of nodes with full set is more reliable. To prove this theorem, let's recall how distributed system reliability is computed. The DSR is the probability of the union of the number of MFSTs that run the given program. The MFST spans one of the full sets we just discovered. If there are more full sets in the DCS, the number of MFSTs is also larger, hence the DSR should be greater because of $P(A \cup B) \geq P(A)$ by the set theory. These are some hints we got from the previous observations: a) A file assignment which has greatest number of groups with full set is our preferred choice, and b) If there are two or more file assignments which contains the same number of groups with full set, the one which groups with full set reside on less number of nodes and their center on stronger nodes is our preferred choice. For each heuristic guess, we will select the best one by following criteria:

- The number of full set is maximum.
- These full sets occupy minimum number of nodes.
- These full sets which have stronger nodes.

The Grouping Rule provides good estimates to those MSR selections and gives us a quite precise method to pick up near-optimum file assignments.

3.3 The Simple_File_Assignment algorithm

The Simple File Assignment algorithm consists of five steps:

Step1: Initialization

Step2: Determine the MFFCs for each node

We try every possible combinations of program and file distributions, check this combination's storage capacity and compare it with all MFFCs in the *'found"* list. If the new found MFFC is over-sized, ignore it. If it is a superset of previously found MFFCs, delete those MFFCs in the *'found'* list.

Step3: Determine the program weakness and node environment weight measures

For program weakness, we add the size of program and all its needed files and does a sort to determine the order. For environment weight, we just add all the link reliabilities for each node under the simplifying assumption of perfect node and does a sort to determine the order.

Step4: Use Minimum Selection Rule to choose good MFFCs

We generate the heuristic Minimum Selection Table and assign weak program to strong node.

Step5: Use Grouping Rule to determine the near-optimum file assignment

We check all our file assignments in Step 4 and pick up the one which has maximum number of full sets. If there are two or more assignments which have maximum number of full sets, we check if the new one occupy less nodes. If two or more assignments occupy the same number of nodes, we check which assignments own stronger nodes.

3.4 The Complete SFA algorithm

```
Algorithm Simple_File_Assignment;
begin
  Step1: Initialization
  Step2: Determine the MFFCs for each node
    for n = 1 to Number_of_Nodes do
      MFFC[n,i] = 0 /* clear i'th MFFC for Node n */
      NC = 0   /* Node Capacity */
      for i = 2^(F+P) downto 1 do   /* i is a FFC's bit
            representation */
        add File/Program Size to NC
      if NC < (This Node's Maximum Capacity) then
        MFFC[n,nm] = FFC   /* add this FFC to the
            list of MFFCs */
        for k = 1 to nm do   /* nm: the number of the
            list of MFFCs */
          if FFC is a superset of MFFC[n,k] delete
              MFFC[n,k]
  Step3: Determine the program weakness and node
        environment weight measures
    for i = 1 to Number_of_Programs do
      for j = 1 to Number_of_Files do
        if (program i needs file j) then weak[i] = weak[i]
            + FileSize[j]
    sort the weak[i] array
    for i = 1 to Number_of_Nodes do
      for j = 1 to Number_of_Edges do
        if (node i has edge j) then weight[i] = weight[i]
            + Link_Rel[j]
    sort the weight[i] array
  Step4: Use Minimum Selection Rule to choose good
        MFFCs
    for i = 1 to Number_of_Nodes do
      for j = 1 to Number_of_Programs do
        for k = 1 to Number_of_MFFCs[i] do
          m = (program j and files needed) - MFFC[i,j]
              /* m: remaining prg/file*/
          if m < min_m then m = min_m   /* pick up
              the minimum */
    for i = 1 to Number_of_Nodes do
      do the assignment (one-by-one)
  Step5: Use Grouping Rule to determine the near-
        optimum file assignment
    for cnt = 1 to Number_of_Minimum_Selections do
      for i = 1 TO 2^Number_of_Nodes do   /* i : Set of Nodes
            bit representation */
        if (set i = FullSet) then FS = FS + 1   /* i :
            Number of FullSets */
      if FS > MaxFS then MaxFS = FS   /* i : find the
          maximum # of FullSets */
      if FS = MaxFS then Check_Stronger_Nodes
end
```

3.5 Time complexity analysis

The time complexity of the SFA algorithm is analyzed and summarized in table 2.

Table 2: The time complexities for SFA algorithm

Step	Complexity
1	$O(1)$
2	$O(n \cdot 2^{p+f})$
3	$O(p \cdot f) + O(p \log p) + O(n \cdot e) + O(n \log n)$
4	$O(n \cdot p \cdot m) + O(n)$
5	$O(s \cdot 2^n)$
Total	$O(n \cdot 2^{p+f} + n \log n + n \cdot p \cdot m + s \cdot 2^n)$

where (n, p, f) = number of (nodes, programs, files),
e = number of edges of the node,
m = number of MFFCs of the node,
s = number of minimum selections.

From table 2, we therefore conclude that the complexity of SFA algorithm is bound by step 2 or step 5.

4. Illustrative examples & Simulation results

Example 3: For a simple distributed computing system with parameters is shown in Figure 4, we apply our Simple File Assignment algorithm to it to show how it works.

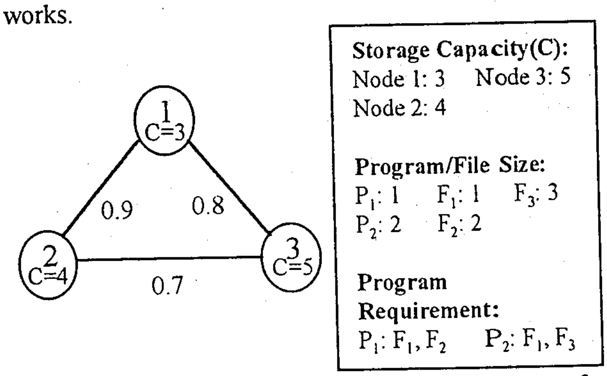


Figure 4. Network topology and data requirement of Example 3.

First, We try every possible combinations of program and file distributions to determine the MFFCs for each node. The left column of table 3 lists all the MFFCs of each node. Then, we compute the program weakness and the node environment weight. The program weakness measure is the sum of the size of the program itself and all required files for successful execution, so $WEAKNESS(P_1) = SIZE(P_1) + SIZE(F_1) + SIZE(F_2) = 1 + 1 + 2 = 4$ and $WEAKNESS(P_2) = SIZE(P_2) + SIZE(F_1) + SIZE(F_3) = 2 + 1 + 3 = 6$. Apparently program P_2 is weaker than program P_1 because P_2 relies on files of bigger size, which is less likely to be assigned because of node capacity limits. The node environment weight measure is the sum of the link reliabilities surrounding that node for the node perfect case here, so $ENV\_WEIGHT(N_1) = 0.9 + 0.8 = 1.7$, $ENV\_WEIGHT(N_2) = 0.9 + 0.7 = 1.6$ and $ENV\_WEIGHT(N_3) = 0.8 + 0.7 = 1.5$. The order of node environment weight measure is $N_1 > N_2 > N_3$.

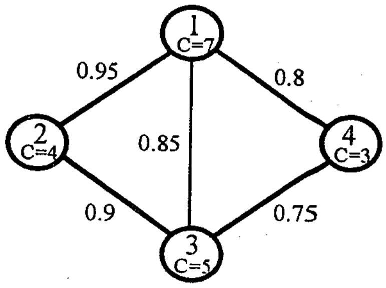
Next, we use MSR to choose good MFFCs. Based on the heuristic which put the weaker program to stronger nodes, we assign P_2 to N_1, P_1 to N_2 and P_2 to N_3. When the favorable order of assignments are determined, table 3 shows how the MFFC of each node is selected: Node N_1 is determined to be favored to Program P_2, so we examine the table 3 to pick up the good MFFCs by the Minimum Selection Rule. Apparently $c_1=\{P_2, F_1\}$ and $f_1=\{F_3\}$ is our preferred choices. Continued this way we got $c_2=\{P_1, F_1, F_2\}$ is the preferred MFFC for node N_2 and $e_3=\{P_2, F_3\}$ is the choice for node N_3. From table 3 we can see there are 6x5x7=210 possible MFFC combinations for this case, and the Minimum Selection Rule effectively cut the choice from 210 to 2.

Table 3: The MSR applied to Example 3

Node N_1 (MFFCs)	P_1	P_2
$a_1=\{P_1,P_2\}$	$F_1+F_2=3$	$F_1+F_3=4$
$b_1=\{P_1,F_1\}$	$F_2=2$	$P_2+F_3=5$
$c_1=\{P_2,F_1\}$	$P_1+F_2=3$	*$F_3=3$*
$d_1=\{P_1,F_2\}$	*$F_1=1$*	$P_2+F_1+F_3=6$
$e_1=\{F_1,F_2\}$	*$P_1=1$*	$P_2+F_3=5$
$f_1=\{F_3\}$	$P_1+F_1+F_2=4$	*$P_2+F_1=3$*
Node N_2 (MFFCs)	**P_1**	**P_2**
$a_2=\{P_1,P_2,F_1\}$	$F_2=2$	$F_3=3$
$b_2=\{P_2,F_2\}$	$P_1+F_1=2$	$F_1+F_3=4$
$c_2=\{P_1,F_1,,F_2\}$	*0*	$P_2+F_3=5$
$d_2=\{P_1,F_3\}$	$F_1+F_2=3$	$P_2+F_1=3$
$e_2=\{F_1,F_3\}$	$P_1+F_2=3$	*$P_2=2$*
Node N_3 (MFFCs)	**P_1**	**P_2**
$a_3=\{P_1,P_2,F_1\}$	$F_2=2$	$F_3=3$
$b_3=\{P_1,P_2,F_2\}$	$F_1=1$	$F_1+F_3=4$
$c_3=\{P_1,F_1,F_2\}$	*0*	$P_2+F_3=5$
$d_3=\{P_2,F_1,F_2\}$	$P_1=1$	$F_3=3$
$e_3=\{P_2,F_3\}$	$P_1+F_1+F_2=4$	*$F_1=1$*
$f_3=\{P_1,F_1,F_3\}$	$F_2=2$	$P_2=2$
$g_3=\{F_2,F_3\}$	$P_1+F_1=2$	$P_2+F_1=3$

Lastly, we apply the Grouping Rule to select our answer from two candidates. Both candidates have the full set $\{N_2, N_3\}$ so GR isn't useful in this case. However, it is irrelevant because both file assignments happened to deliver the optimal solution R=0.916.

Example 4:



Storage Capacity:	Program/File Size:		Program Req.:
Node 1: 7	P_1: 1	F_1: 1	P_1: F_1, F_2, F_3
Node 2: 4	P_2: 2	F_2: 2	P_2: F_1, F_2, F_4
Node 3: 5	P_3: 3	F_3: 3	P_3: F_2, F_3, F_4
Node 4: 3		F_4: 4	

Figure 5. Network topology and data requirement of Example 4

The measure results of program weakness and node environment weight of Example 4 are listed below:

$ENV\_WEIGHT(N_1) = 2.6$ $\quad WEAKNESS(P_1) = 7$
$ENV\_WEIGHT(N_2) = 1.85$ $\quad WEAKNESS(P_2) = 9$
$ENV\_WEIGHT(N_3) = 2.5$ $\quad WEAKNESS(P_3) = 12$
$ENV\_WEIGHT(N_4) = 1.55$

Follow the same heuristic, we put the weaker program to stronger nodes. When the favorable order of assignments are determined, we use the MSR to pick up the good MFFCs. Then we apply the Grouping Rule to select our answer from the candidates. Table 4 lists all the 5 possible heuristic guesses out of 24 in the Example 4. By applying our Grouping Rule, we know the second and the fifth ones (in bold face) are our preferred choices because they span only three nodes N_1, N_2 and N_3. The last column of table 4 is the computed DSR values of each file assignment, which reflect the fact that our Grouping Rule is really quite effective. Table 5 shows the simulation results of Example 4.

Table 4: The file assignments and full set groups of Example 4

	Node 1	Node 2	Node 3	Node 4	Full Set	DSR
1	$\{F_3, F_4\}$	$\{P_1, F_1, F_2\}$	$\{P_2, F_1, F_2\}$	$\{P_3\}$	N_1,N_2,N_3,N_4	0.9549
2	**$\{P_3, F_4\}$**	**$\{P_1, F_3\}$**	**$\{P_2, F_1, F_2\}$**	**$\{P_3\}$**	**N_1,N_2,N_3**	**0.9787875**
3	$\{F_3, F_4\}$	$\{P_1, F_3\}$	$\{P_2, F_1, F_2\}$	$\{P_3\}$	N_1,N_2,N_3,N_4	0.9549
4	$\{P_3, F_4\}$	$\{P_1, F_1, F_2\}$	$\{P_2, F_1, F_2\}$	$\{F_3\}$	N_1,N_2,N_3,N_4	0.9549
5	**$\{P_3, F_4\}$**	**$\{P_1, F_3\}$**	**$\{P_2, F_1, F_2\}$**	**$\{F_3\}$**	**N_1,N_2,N_3**	**0.9787875**

Table 5: Simulation results of Example 4

Number of MFFCs		10,752 = (16x8x12x7)	
Possible Heuristic Solutions		24	
Heuristic Guesses		5	
Speed Up Ratio		448 = (10752/24)	
DSR_{max}	0.9787875	DSR_{min}	0.9549
DSR_{SFA}	0.9787875	E_r	0%

where DSR_{max} = maximum DSR of random assignment
DSR_{min} = minimum DSR of random assignment
DSR_{SFA} = solution of SFA algorithm

$$E_r = \text{Relative error} = \frac{DSR_{max} - DSR_{SFA}}{DSR_{max} - DSR_{min}}$$

5. Summary and Conclusion

Distributed computing system has become a major trend in today's computer system design for its high fault-tolerance, potential for parallel processing and better reliability performance.One of the distributed computing system important characteristics is that it offers redundant copies of software to improve system's reliability. To effectively distribute these redundant copies of software to appropriate nodes is the basic consideration for file assignment problem. The problem has been proved to be an NP-problem. Traditional solution techniques such as back-tracking algorithm and mathematical programming can give the optimal solution, but they have to pay very high computation price as well as high storage problem. To effectively reduce problem space is an important research subject.

In this paper, we develop a heuristic algorithm called Simple File Assignment algorithm for the reliability-oriented file assignment problem to reduce the problem space. Based on the numerical simulation results, our simple file assignment algorithm obtain the exact solution in most cases with much improved computation time. Examples are given to illustrate the applicability of our approach. Simulation results are analyzed to justify the applicability and efficiency of our approach.

References

[1] P. Enslow, "What is a distributed data processing system," Computer, vol. 11, Jan. 1978.

[2] D. A. Rennels, "Distributed fault-tolerant computer systems," Computer, vol. 13, pp. 55-65, Mar. 1980.

[3] D. W. Davies, E. Holler and E. D. Jensen, S. R. Kimbleton, B. W. Lampson, G. Lelann, K. J. Thurber, and R.W. Watson, "Distributed systems architecture and implementation," in Lecture Notes in Computer Science, vol. 105, Berlin, Germany : Springer-Verlag, 1981.

[4] V. K. Kumar, S. Hariri, C. S. Raghavendra, "Distributed program reliability analysis", IEEE Trans. on Software Engineering, vol. SE-12, 1986 Jan., pp 42-50.

[5] C. S. Raghavendra, V. K. Kumar, S. Hariri, "Reliability analysis in distributed systems", IEEE Trans. Computer, vol C-37, Mar. 1988, pp. 352-358.

[6] W. W. Chu, L. J. Holloway, M. T. Lan, K.Efe, "Task allocation in distributed data processing", IEEE Computer Magazine, Nov. 1980, pp. 57-69.

[7] V. Rajendra Prasad, Y. P. Aneja, K. P. K. Nair, "A Heuristic Approach to Optimal Assignment of Components to a Parallel-Series Network", IEEE Trans. Reliability, vol. 40, no. 5, Dec. 1991.

[8] C. V. Ramamoorthy, "The Isomorphism of Simple File Allocation", IEEE Trans. Computer, vol. C-32, Mar. 1983.

[9] Wesley W. Chu, "Optimal File Allocation in a Multiple Computer Systems", IEEE Trans. Computer, vol. C-18, no. 10, Oct. 1969.

[10] K. P. Eswaran, "Placement of records in a file and file allocation in a computer network", Information Processing 74, IFIPS. New York: North Holland, 1974.

[11] C. P. Wang, "On the study of file assignment in distributed system", NCTU Tech. Rep., 1990.

[12] G. J. Hwang, S. S. Tseng, "A Heuristic Task Assignment Algorithm to Maximize Reliability of a Distributed System", IEEE Trans. on Reliability, vol. 42, no. 3, Sep. 1993, pp. 408-415.

[13] M. S. Lin and D. J. Chen, "New Algorithm for the Reliability Analysis of Distributed Systems ", Journal of Information Science and Engineering 8(3) 1992.

[14] D. J. Chen and T. H. Huang, "Reliability Analysis of Distributed Systems Based on a Fast Reliability Algorithm", IEEE Trans. on Parallel and Distributed Systems vol. 3, no. 2, pp. 139-154, Mar. 1992.

[15] A. Kumar, S. Rai and D. P. Agrawal, " On Computer Communication Network Reliability Under Program Execution Constraints", IEEE Journal on Selected Areas in Communication, vol. 6, no. 8, pp. 1393-1399, Oct. 1998.