Computación y Sistemas Vol. 5 No. 3 pp. 192 - 203
© 2002, CIC - IPN. *ISSN 1405-5546* Impreso en México

File Decomposition, Replication and Assignments Problems

Problemas de Decomposición, Replicación y Asignación de Archivos

José Lisandro Aguilar Castro
CEMISID, Departamento de Computación
Facultad de Ingeniería, Universidad de los Andes
Av. Tulio Febres 5101, Mérida, Estado Mérida-Venezuela
e-mail : aguilar@ing.ula.ve

Article received on July 19, 1999; accepted on December 25, 2001

Abstract

File decomposition, replication and assignment problems have been three of the principal research topics in parallel and distributed processing. In this paper, we present these problems and propose a heuristic algorithm for determining effective file decomposition, replication and assignment solutions. At first, a model is developed for decomposing, replicating and allocating files on distributed systems. The model considers storage costs, communication costs, the query rate, updating rates of files, the maximum expected access times to files at each computer, the storage capacity of each computer and the workload imbalance cost. The criterion of optimality is minimal overall operating costs. Because these problems are in general NP-hard, we propose a heuristic algorithm based on genetic algorithm to solve them. Several examples for different distributed systems are given to illustrate our model.

Keywords: File Systems, Distributed Systems, Performance Evaluation

Resumen

Los problemas de descomposición, replica y asignación de archivos, han sido tres de los principales tópicos de investigación en sistemas paralelos y distribuidos. En este trabajo nosotros presentamos estos problemas, y proponemos algoritmos heurísticos para alcanzar una efectiva descomposición, replicación y asignación de archivos. Al inicio es desarrollado un modelo para estos problemas. El modelo considera los costos de almacenamiento, de comunicación, de consulta, de actualización, así como las capacidades de almacenamiento de cada sitio y los costos por desequilibrio de las cargas de trabajo. El criterio de optimización consiste en minimizar los costos operativos. Al ser este problema NP-Completo, nosotros proponemos algoritmos heurísticos basados en los Algoritmos Genéticos para resolverlos. Varios ejemplos de uso de nuestro modelo son presentados para diferentes arquitecturas distribuidas

Palabras claves: Sistemas de Archivos, Sistemas Distribuidos, Evaluación de Rendimiento

1 Introduction

Distributed Computing Systems present many challenges, some of which have been already encountered in centralized computing systems, while others are new and unique to a distributed environment. Technical issues include systems integrity, concurrence control, addressing, naming and security. Other issues involve managerial and design problems, which include decisions on the allocation of files and databases to sites (computers) [Chapman et al., 1993; Chen and Sheu, 1994; Gifford, 1987; Kalns and Ni, 1995; Knobe and Natarajan, 1993; Mahmood et al., 1994; Lee, 1997; Lee, 1995; Yu-Kwong et al., 1996], allocation of tasks [Aguilar, 1998; Aguilar and Kloul, 1997; Aguilar and Gelenbe; 1997; Aguilar and Jimenez, 1997], etc.

In distributed systems, files are accessed either for update or retrieval activities by geographically dispersed users and applications. An important issue in the design/management of the distributed systems is designing the file system. The file system must solve problems such as files replication level (optimal number of copies for each file), files fragmentation level (optimal number of fragments for each file) and the allocation of them to the participating sites to achieve satisfactory systems performance. These problems are quite important because using a naive file system on shared distributed platforms may result in load imbalance, performance degradation, excessive communication overhead, etc.

These problems have been proven to be NP-Complete [Kalns and Ni, 1995; Li and Chen, 1991]. Lee derived efficient dynamic programming algorithms for determining effective data allocation schema to execute a sequence of Do-loops [Lee, 1997; Lee, 1997]. Other previous paralleling compiler research has emphasized allowing programmers to specify the data allocation/decomposition using language extensions, such that compilers can then generate all the communication instruction according to these language extensions (for example, HPF) [Chen and Sheu, 1994; Li and Chen, 1991]. That is, it is also possible to use compiler techniques to automatically determine data allocation/decomposition. [Mahmood et al., 1994], propose an algorithm to solve the adaptive file allocation in distributed systems to minimize the overall communication and storage costs.

Their algorithm makes an one-period look ahead incremental allocation. In [Gifford, 1987] is introduced an algorithm for the maintenance of replicated files. The reliability and performance characteristics of a replicated file are controlled by appropriately choosing r (read quorum of votes to read a file) and w (write quorum of votes to write a file). Other papers, which addressed the problem of determining file allocations include [Chapmand et al., 1993; Kalns and Ni, 1995; Knobe and Natarajan, 1993].

This paper is concerned with modeling file decomposition, replication and assignment problems on distributed system, and with designing an efficient algorithm based on Genetic Algorithm (GA) to solve them. Optimal solutions to these problems are determined on the basis of the query rate, updating rates of files, the workload imbalance cost, storage costs, communication costs, and the system's limitations (the storage capacity of each computer, the maximum expected access times to files at each computer, the interconnection topology, etc.). Our model uses as input a listing of the files in the organization, a listing of sites, the number of transactions (updates and queries) originating from those sites, and the amount of data to transfer for each transaction. Based on this information and the cost structure for each criterion, we formulate our optimization model. We develop an algorithm based on GA to solve it. This algorithm is tested on different distributed/parallel platforms. Our model will allow deciding on: How many replicas of a given file should the system have?, How fragments should be decomposing a file in?, Which sites will be selected for placement of files (fragments or replicas)?, From which file location will the data be retrieved in each transaction?.

The rest of this paper is organized as follows. In section 2, we illustrate file decomposition, allocation and replication problems. In section 3, we analyze different criteria that must be taken into consideration to solve these problems. In section 4, we introduce GA and propose an efficient algorithm for solving these problems based on GA. In section 5, we present experimental studies on different distributed/parallel systems. Finally, some concluding remarks are given in section 6.

2 Problem Definition

In this section, we will present typical problems to solve during the design and management of file system for distributed system. These problems are: the file decomposition problem, the file replication problem and the allocation problem.

2.1 File Replication Problem

The problem is the following: for a given number of files, what is the number of replicas/copies for each one and where they must be assigned to maximize system performance. This problem is typical in systems where geographically dispersed users or applications access files and the communication cost is very high. This problem depends of several criteria: communication cost, storage cost, update cost, number of updates request in the system, etc. Therefore, the goal of file replication is to determine the number of copies of a file to maximize system performances by reducing the cost of managing and the communication cost. These copies must be assigned in the system in a way that optimizes performance. That is, this problem includes a file replica allocation problem. In this case, one of the main problems is the correct operation of the system that requires that consistency be maintained among copies of the same file in different sites [Gifford, 1987].

2.2 File Decomposition Problem

The problem is the following: for a given number of files, what is the optimal number of fragments for each one and where they must be assigned to maximize system performance. This problem is typical in systems where geographically dispersed users or applications access files, but each user or application accesses a different file fragment. The performance of different file decompositions can be measured in terms of retrieval time for different type of queries, the load distribution, and the storage utilization. Therefore, the goal of file decomposition is to divide a file in fragments to maximize system performance by balancing the computational load among the sites and by minimizing remote memory accesses (or communication messages). These fragments must be assigned over the different sites on the system in a way that optimizes performance. That is, this problem includes a file fragment allocation problem. An extension of this problem is the multidimensional decomposition; that is, the attribute space is partitioned and mapped to different physical locations.

2.3 File Assignment Problem

The problem is the following: given a number of sites that process common information files, how can one allocate the files so that the assignment yields minimum overall operating costs subject to constraints [Mahmood et al., 1994; Yu-Kwong et al., 1996]. Therefore, the problem is that of assigning nf files to n sites in a way that optimizes performance. Typically, the problem is then characterized by the following objectives:

- The communication between different sites of the system must be kept to a minimum.
- The load of the different sites must be balanced.
- The total effective execution time of applications must be minimized.

We can adapt this definition for the cases of file fragmentation or file replication problems. For these cases, we search an optimal assigning of the file fragments or file replicas.

3 Definition of the Cost Function

In this section, cost functions are presented to solve these problems. First, we will analyze different criteria, which are used for solving these problems. Then, we present general cost functions using these criteria. The cost functions are based on these criteria, because they correspond to different aspects that must be optimized into a given parallel/distributed system to improve its performance. This is one of the novel aspects presented by

this paper. The following assumptions are used in order to formulate our model:

Assumption 1. Each transaction is routed to the more close location with the file that transaction requires.

Assumption 2. Each transaction can be processed in a single location.

Assumption 3. Updates are made simultaneously on all locations with file replica.

3.1 Files Replication and the Related Cost Functions

For this problem, we define five costs: storage cost, update cost, query cost, load imbalancing cost and cost for non-optimal number of copies. They are defined as:

3.1.1 Storage Cost

This cost depends on the size of the files, and on the cost to store a length unit of a file in a given site.

$$Cs = \sum_{f=1}^{nf}\sum_{k=1}^{n} Q_k L_f X_{fk} \qquad (1)$$

where, nf, number of files.
n, number of sites.
X_{fk}, is a binary variable whose value is 1 if f^{th} file is stored in the k^{th} site and is 0 Otherwise.
L_f, size of the file f.
Q_k, unit storage cost per unit length and unit time at site k.

3.1.2 Update Cost

This cost is often considered to be one of the most important factors that need to be minimized. It depends on the number of updates for each file from each processor (and the quantity of information to update), on the cost to update a length unit of a file in a site, and on the average cost to transfer a length unit of a file from a site to other site.

$$Cu = \sum_{f=1}^{nf}\left(\sum_{k=1}^{n}\left(\sum_{j,j\neq k}^{n}\left(C_{jk} AT'_k X_{fk} Vu_{fj}\right) + AT'_k X_{fk} Vu_{fk}\right)\right) \qquad (2)$$

where, AT'_k, unit update cost per unit length and unit time at site k.

Vu_{fj}, quantity of information to update of the f^{th} file from the j^{th} site per unit time.

C_{jk}, transmission cost from the j^{th} site to the k^{th} site per unit length and unit time.

3.1.3 Query Cost

This cost depends on the number of requests for each file from each site (and the quantity of information to query), on the cost to query a length unit of a file in a site, and on the average cost to transfer a length unit of a file from a site to other site.

$$Cq = \sum_{f=1}^{nf}\left(\sum_{k=1}^{n}\left(MIN_{k\neq j \& j\in I_f}\left(C_{kj} AT_j X_{fj}\right)\left(1-X_{fk}\right)Vq_{fk} + AT_k X_{fk} Vq_{fk}\right)\right) \qquad (3)$$

where, AT_k, unit query cost per unit length and unit time at site k.

Vq_{fj}, quantity of information to request of the f^{th} file from the j^{th} site per unit time.

I_f, set of sites with a copy of the f^{th} file.

3.1.4 Cost of load imbalance

An optimal file replication must assure an equitable distribution of the workload between sites. A possible form for this cost term is

$$Cli = \frac{\sum_{j=1}^{n}\left(CLI_j - \left(\sum_{i=1}^{n}\left(CLI_i / n\right)\right)^2\right)}{n} \qquad (4)$$

where, CLIj, workload in the site j:

$$CLI_j = \sum_{f=1}^{nf}\left(\sum_{k=1}^{n} s\left(AT_j X_{fj}\right)\left(1 - X_{fk}\right)Vq_{fk} + AT_j X_{fj} Vq_{fj} + \sum_{k=1}^{n} AT'_j X_{fj} Vu_{fk}\right)$$

where, s = {1 if $C_{kj}AT_jX_{fj} = MIN(C_{km}AT_mX_{fm})$ for $\forall m=1, n$ & $m\in I_f$ 0 otherwise

3.1.5 Cost for Non-Optimal Number of Copies

This cost penalizes if the system has not the optimal number of copies for each file according to the Eq. 9. It depends on the quantity of information to update and the quantity of information to request.

$$Cno = \sum_{f=1}^{nf}\left|\left(\sum_{k=1}^{n} X_{fk}\right) - r_f\right| \qquad (5)$$

where, r_f, number of copies of the file f according to the Eq. 7.

Finally, we shall express the operating cost (objective function) according to the next equation:

$$CF = Cs + Cu + Cq + Cli + Cno \quad (6)$$

3.1.6 Constraints

The classical constraints are [Kalns et al. 1995; Lee, 1997; Mahmood, 1994]:

a) To store r_f redundant copies of the f^{th} file:

$$r_f = \sum_{k=1}^{n} X_{fk} \quad (7)$$

b) To assure that the storage capacity of each site is not exceeded:

$$\sum_{f=1}^{nf} L_f X_{fk} \leq S_k \quad (8)$$

where, S_k, storage capacity of the site k.

c) To determine the optimal number of copies according to the quantity of information to update and the quantity of information to request

$$r_f \leq \frac{\sum_{j=1}^{n} Vq_{fj}}{\sum_{j=1}^{n} Vu_{fj}} \quad (9)$$

To solve the file replication problem we must minimize the next cost function:

MIN (CF)

Therefore, the objective for adding a copy of a file is to minimize (6) subject to the constraints (7), (8) and (9).

3.2 File Decomposition and the Related Cost Functions

For this problem, we define four costs: storage cost, update cost, query cost, and load imbalancing cost. They are defined as:

3.2.1 Storage Cost

This cost depends on the size of the fragment of each file, and on the cost to store a length unit for a file in a given site.

$$Cs = \sum_{k=1}^{nf} \sum_{f=1}^{c_k} \sum_{i=1}^{n} Q_i L'_{fk} Y_{ifk} \quad (10)$$

where, c_k, number of fragments of the file k.

Y_{ifk}, is a binary variable whose value is 1 if f^{th} fragment of the k^{th} file is stored in the i^{th} site and is 0 Otherwise.

L'_{fk}, size of the f^{th} fragment of the k^{th} file.

3.2.2 Update Cost

It depends on the number of updates for each fragment of each file from each processor (and the quantity of information to update), on the cost to update a length unit of a file in a site, and on the average cost to transfer a length unit of a file from a site to other site.

$$Cu = \sum_{k=1}^{nf} \sum_{f=1}^{c_k} \sum_{q=1}^{n} \left(\sum_{i \neq q}^{n} \left(C_{iq} AT'_q Y_{qfk} Vu'_{fki} \right) + AT'_q Y_{qfk} Vu'_{fkq} \right) \quad (11)$$

where, Vu'_{fki}, quantity of information to update of the f^{th} fragment of the k^{th} file from the i^{th} sites per unit time.

3.2.3 Query Cost

This cost depends on the number of requests for each fragment of each file from each site (and the quantity of information to query), on the cost to query a length unit of a file in a site, and on the average cost to transfer a length unit of a file from a site to other site.

$$Cq = \sum_{k=1}^{nf} \sum_{f=1}^{c_k} \sum_{q=1}^{n} \left(\sum_{i,i \neq q}^{n} \left(C_{iq} AT_q Y_{qfk} Vq'_{fki} \right) + AT_q Y_{qfk} Vq'_{fkq} \right) \quad (12)$$

where, Vq'_{fki}, quantity of information to request of the f^{th} fragment of the k^{th} file from the i^{th} site per unit time.

3.2.4 Cost of Load Imbalance

An optimal file decomposition must assure an equitable distribution of the workload between the processors. A possible form for this cost term is

$$Cli = \frac{\sum_{q=1}^{n} \left(CLI_q - \left(\sum_{i=1}^{n} (CLI_i / n) \right)^2 \right)}{n} \quad (13)$$

where, CLIq, workload in the site q:

$$CLI_q = \sum_{k=1}^{nf} \left(\sum_{f=1}^{c_k} \sum_{i=1}^{n} \left(AT'_q Vu'_{fki} + AT_q Vq'_{fki} \right) Y_{qfk} \right)$$

3.2.5 Constraints

The classical constraints are [Kalns et al. 1995; Lee, 1997; Mahmood, 1994]:

b) To assure that the storage capacity of each site is not exceeded

$$\sum_{k=1}^{nf}\sum_{f=1}^{c_k} L'_{fk} Y_{ifk} \leq S_i \qquad (14)$$

d) To assure non-duplication fragments

$$\sum_{i=1}^{n}\sum_{k=1}^{nf}\sum_{f=1}^{c_k} Y_{ifk} = 1 \qquad (15)$$

e) To assure files unit

$$\sum_{i=1}^{n}\sum_{f=1}^{c_k} L'_{fk} Y_{ifk} = L_k \qquad (16)$$

To solve the file decomposition problem we must minimize the next cost function:

$$CF = Cs + Cu + Cq + Cli \qquad (17)$$

Therefore, the objective for decomposing a file is to minimize (17) subject to the constraints (14), (15) and (16).

4 Resolution Method: Genetic Algorithms

GA is an optimization algorithm based on the principles of evolution in biology. A GA follows an "intelligent evolution" process for individuals based on the utilization of evolution operators such as mutation, inversion, selection and crossover [Aguilar and Kloul,1997; Aguilar and Gelenbe, 1997; Goldberg, 1989]. The idea is to find a good local optimum, starting from a set of initial solutions, by applying the evolution operators to successive solutions. The procedure evolves until it remains trapped in a local minimum.

The GA applied in our problem follows the next procedure: we define a space of research of *nf* vectors where everyone represents an individual, and every individual represents a possible solution. Each vector has *nf* elements and the i^{th} element is a vector with the number of copies (or fragments) of the i^{th} file, and each element of the i^{th} vector (file) has a value among *1... n*, according to the site to which the copy of the i^{th} file (or fragment of the i^{th} file) is assigned. Furthermore, we use the cost function (6) (or (17)) to determine the cost of every individual. We begin with an initial population of individuals randomly defined and we choose the individuals with minimal cost for generating new individuals using the genetic operators. Since the population is constant, we substitute the worst individuals of initial solution by the best individuals generated. The procedure stops if we exceed a given number of generations without finding a better solution.

In this method two parameters we studied: the maximum number of generations (NUMGEN), the number of individuals (NUMIND) and the probability (PM) to use the mutation operator after the crossover operator. The first parameters allow to optimize the speed-up of the algorithm to reach an optimal solution. We remark than the quality of the solutions improves more rapidly in the first generations that in the following. Thus, a satisfactory quality is obtained rapidly without to wait that the algorithm converges. In this work, we used the crossover operator and then the mutation operator according to the PM probability. The mutation operator can change the number of copies (or fragments) of a given file, or the site where the copies (or fragments) of a given file are assigned. For the PM parameter, if the probability is large we obtain good results, but a large PM implies an execution time large. The general genetic algorithm is defined as:

Generation of individuals which represent potential solutions:
Repeat until system convergence
  Evaluation of every individual
  Selection of the best individual for reproduction
  Reproduction of the individual using evolutive operators
  Replace the worst old individuals by the new individuals

For the file replication problem, our approach not only generates the near optimal replicas allocation, also generates the optimal number of copies for each file. For the file decomposition problem, our approach not only generates the near optimal fragments allocation, also generates the optimal number of fragments for each file.

5 Results Analysis

We gathered statistic in the following manner: the number of simulations per a given set of parameters (total number of files, total number of processors, quantity information to query, etc.) was either (depending of which occurs first):

- 30 simulations
- The number of simulations required for obtaining a given standard deviation of the cost functions (6) or (17).

The standard values considered are the following:

Number of files (nf): 5, 15, 30, 50, 100
Number of sites (n): 5, 10, 15, 20
Maximal number of fragments per file (c_f): 3, 10, 15
Maximal size of the files (L_f): between 1 and 10
Quantity of information to update from a file/fragment (Vu_{fj}/Vu'_{flj}): between 3 and 7
Quantity of information to query from a file/fragment (Vq_{fj}/Vq'_{flj}): 7*((Vu_{fj} or Vu'_{flj})
Unit Transmission cost (C_{kj}): 1 if k and j are connected directly, between 2 and 5 otherwise
Unit Storage cost (Q_k): between 5 and 7
Maximal storage capacity per site (S_k): between 30 and 100
Unit Query cost (AT_k): between 10 and 50
Unit Update cost (AT'_k): AT_k*5

$$\forall f=1, nf; \quad \forall k,j=1, n; \quad \forall l=1, c_f;$$

For the case of the file replication problem, the cost functions studied here are the following:

$$F1 = C_s + C_u + C_q + C_{li}$$

$F2 = C_u + C_q$
$F3 = C_s + C_u + C_q + C_{li} + C_{no}$
$F4 = C_u + C_q + C_{li}$

For the case of the file decomposition problem, the cost functions studied here are the following:

$F1 = C_s + C_u + C_q + C_{li}$
$F2 = C_s + C_u + C_q$
$F3 = C_u + C_q$
$F4 = C_u + C_{il}$
$F5 = C_u + C_q + C_{il}$

In the next part, the value of these cost functions and the execution time to obtain a near optimal solution using our approach are presented for different distributed systems. Due to space limitations, the figures presented in this section were chosen because they are representative of the phenomenon studied.

The GA parameter values are the following:

NUMGEN: 20, 50, 100.
PM: 0.2, 0.5, 0.8, 1.
NUMIND: 10, 20, 50.

For each case, we have tested the different parameter combinations of the GA and selected the best one according to the best individual given for each combination (see [Aguilar, 2001] for more details). The crossover operator is used all time, and the mutation operator according to the PM probability.

5.1 Homogeneous Systems with Full Connection

In this case, we consider a distributed system architecture which consisting of a collection of n homogeneous sites with distributed memory. The sites are fully interconnected connected with a reliable high-speed network. For this architecture, $AT_k = AT'_k = Q_k = C_{kj} = 1$, $\forall$. k,j=1, n.

5.1.1 File Replication

In the figure 1, F2 and F4 give the best results for systems with more than 15 files, and their execution times are similar. F1 and F3 have large execution times because they evaluate more criteria. F4 explicitly minimizes total communication with a load balancing. This is the reason of the best results with it. In the figure 2, F2 gives the best results. That is, the main objective to minimize is the load balancing because this platform is homogenous.

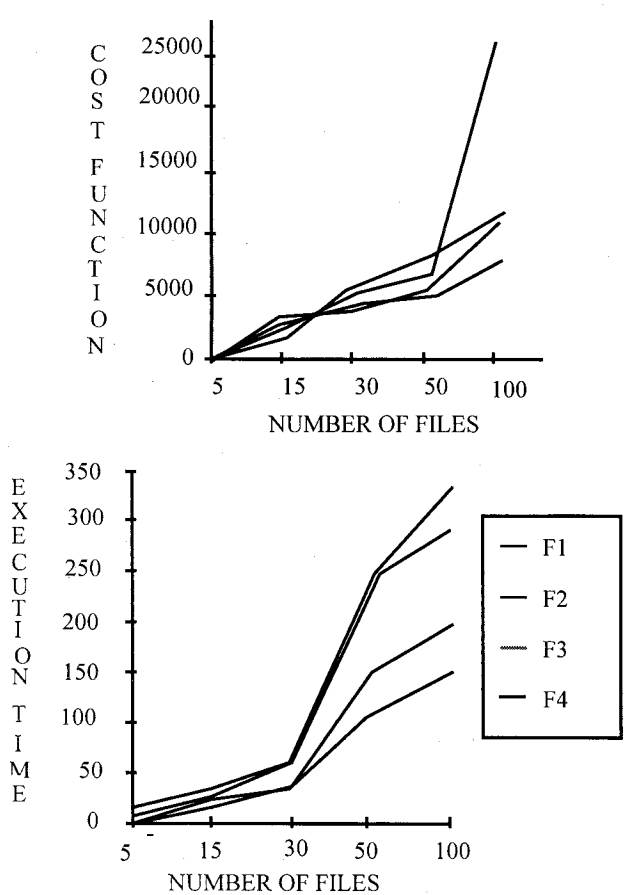


Figure 1. Results of the simulation for n= 10

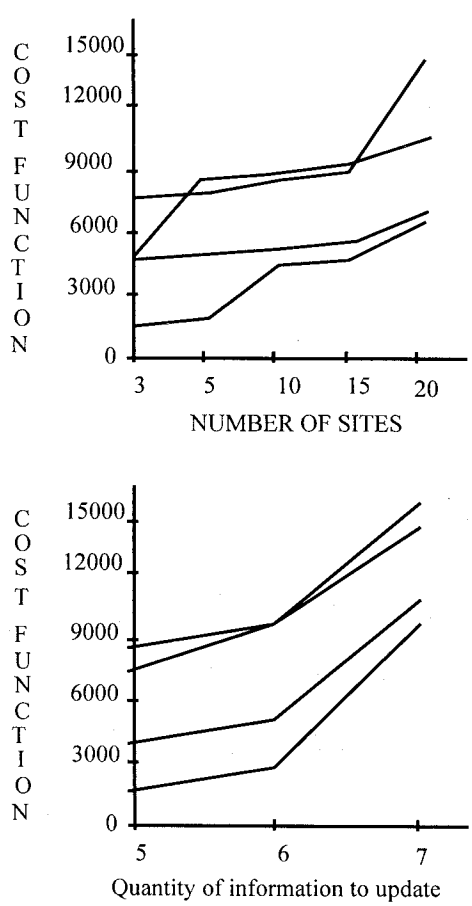


Figure 2. Results of the simulation for nf= 100

5.1.2 File Decomposition

For this type case, (Figures 3, 4, 5), the cost function and execution times are very large because we evaluate more criteria. F1 and F5, need a very large time to reach suboptimal solutions. For systems of little size, the difference between the functions is not important for different numbers of files. Otherwise, F2 and F3 are more interesting, because they give the best results in a reasonable execution time. These cost functions minimize the communication cost. Load balancing introduces a large value in the cost functions. Figure 5 shows the influence of the quantity of information to update in the communication cost. For F1, F4 and F5, this criterion has a large influence on the results. The rate of increase of the cost functions and of the execution times seems to be exponential. According to that, for a system with more than 100 files (100+k, for a small value of k), the cost functions F1, F4 and F5 will be very high and the execution times impossible. In general, F2 and F3 could have the same problem but for a large value of k.

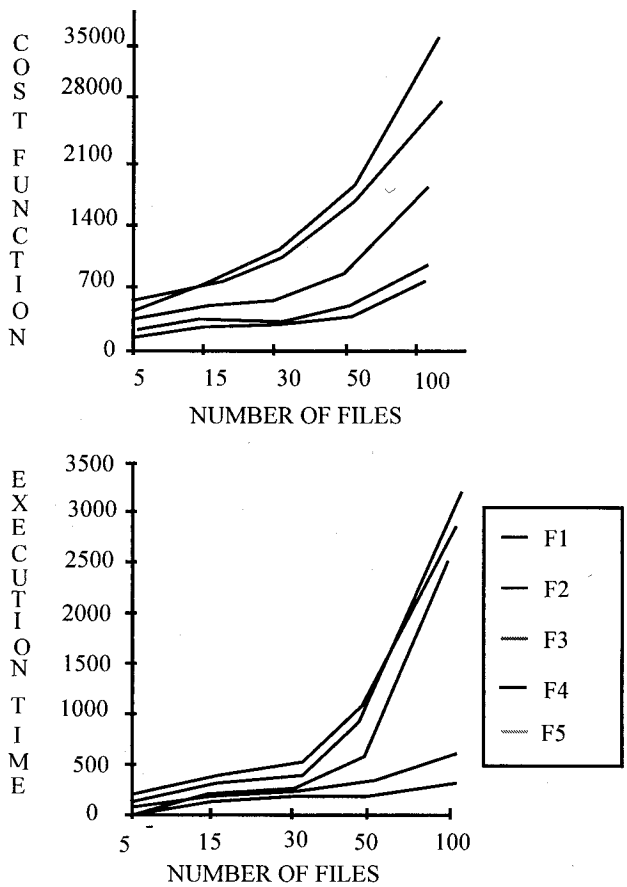


Figure 3. Results of the simulation for n= 5, c_f= 3

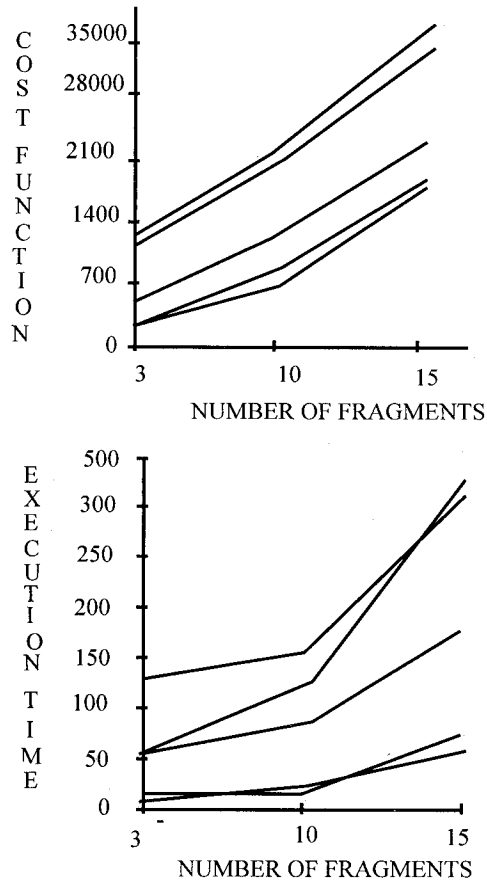


Figure 4. Results of the simulation for nf= 15, n= 10

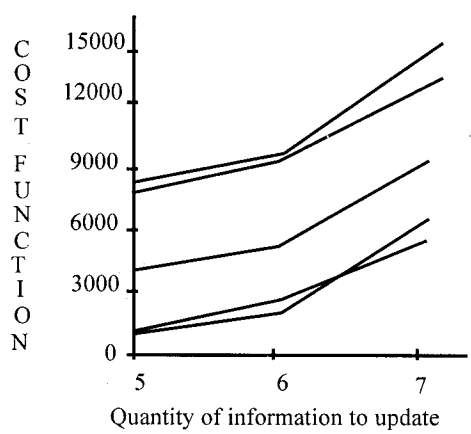


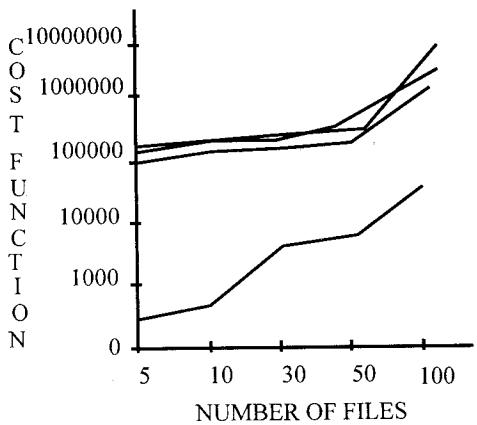
Figure 5. Results of the simulation for nf= 15, n= 10, c_f= 3

5.2 Homogeneous System without Full Connection

In this case, we consider a distributed system architecture which consisting of a collection of n homogeneous sites with distributed memory, but the sites are not fully interconnected. For this architecture, $AT_k = AT'_k = Q_k = 1$, $\forall$ k=1, n.

5.2.1 File Replication

Figure 6 shows the cost function reached and their execution times. F2 gives the best results, because F2 minimize the communication cost, the more important criterion in this platform. The cost function and execution times are similar between F3, F1 and F4. In general, F4 does not give better results because load balancing cost increase cost value.



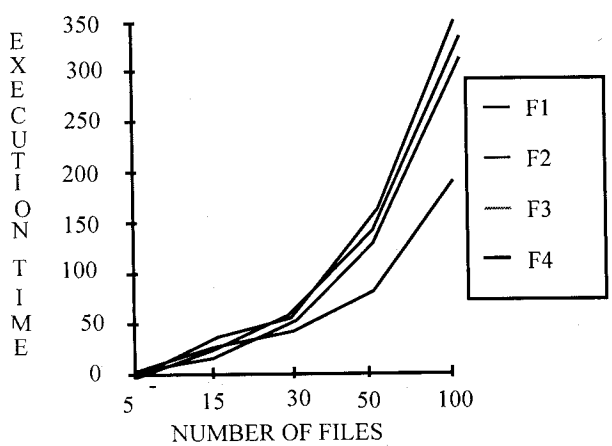
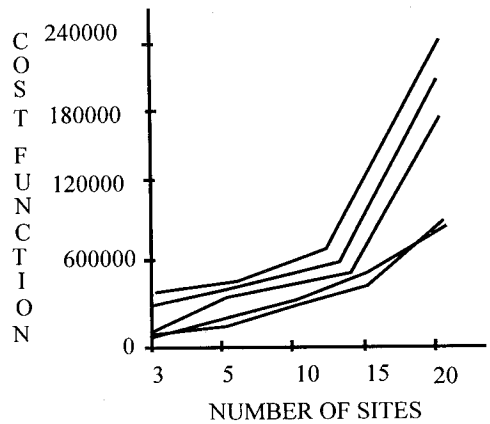


Figure 6. Results of the simulation for n= 20

5.2.2 File Decomposition

The results clearly show that F2 and F3 give the best results in all cases, with a littler execution time. This performance improvement is particularly substantial when the system is large. For large systems, load balancing cost is large.



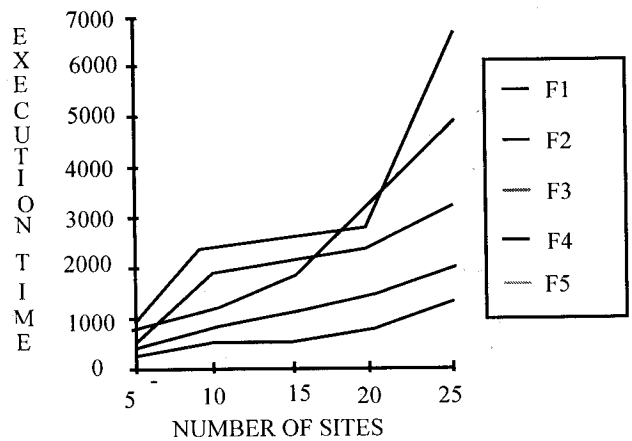


Figure 7. Results of the simulation for nf= 100, c_f= 3

5.3 Heterogeneous System with Full Connection

In this case, we consider a distributed system architecture which consisting of a collection of n heterogeneous sites with distributed memory. The sites are fully interconnected connected with a reliable high-speed network. For this architecture, $C_{kj} = 1$, $\forall$ k,j=1, n.

5.3.1 File Replication

In this case, F2 gives the best results. There is a large difference between this cost and F1, F3 and F4. That is due to the large value of the load balancing cost.

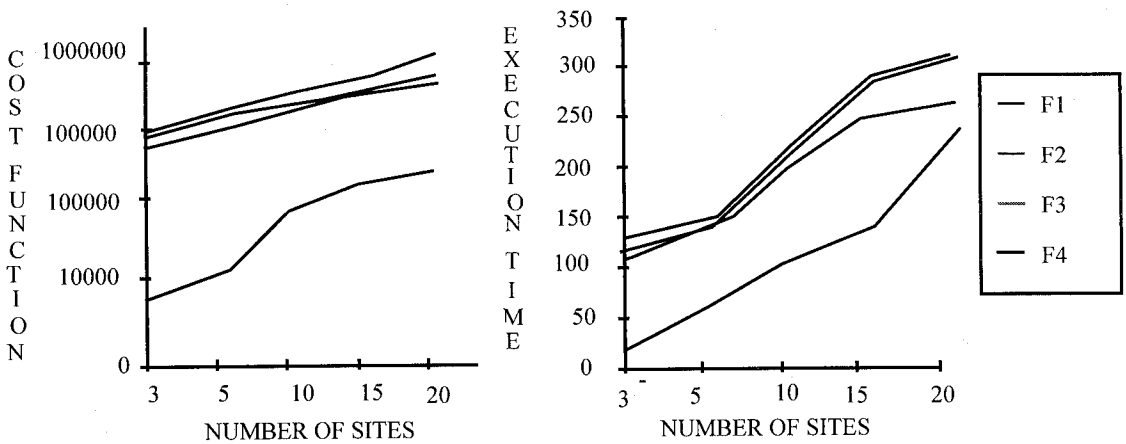


Figure 8. Results of the simulation for nf= 100

5.3.2 File Decomposition

In this case, F2 and F3 give the best results. The combination $C_u + C_q + C_{li}$ gives a large cost function. Storage cost is little.

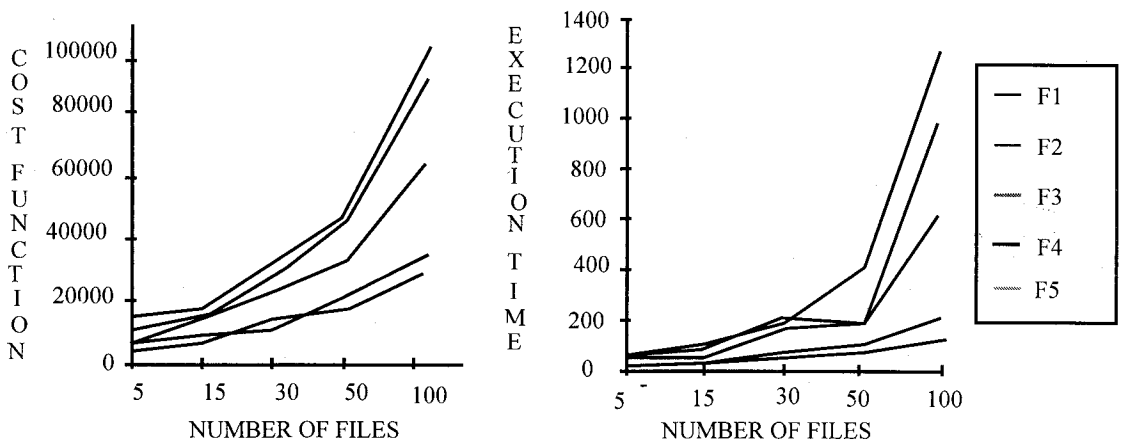


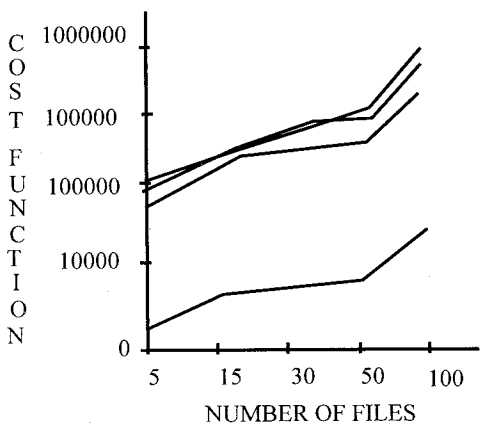
Figure 9. Results of the simulation for n= 20, c_f= 3

5.4 Heterogeneous System without Full Connection

In this case, we consider a distributed system architecture which consisting of a collection of *n* heterogeneous sites with distributed memory, but the sites are not fully interconnected.

5.4.1 File Replication

In this case, we obtain the same results than the previous platforms. F2 gives the best results: C_{no} is not large and C_{li} gives the largest values.



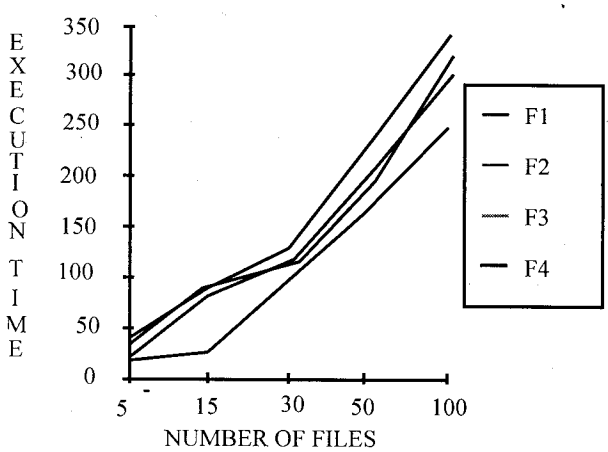
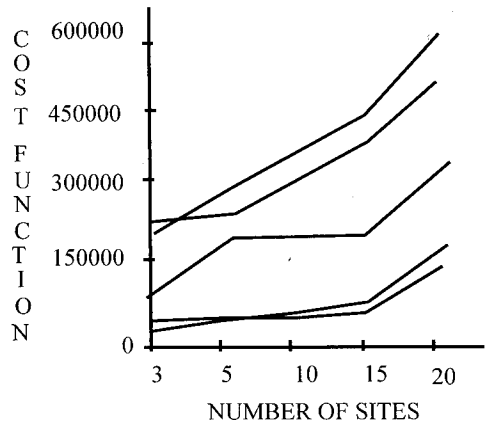


Figure 10. Results of the simulation for n= 20

5.4.2 File Decomposition

In this case, F2 and F3 give the best results again.



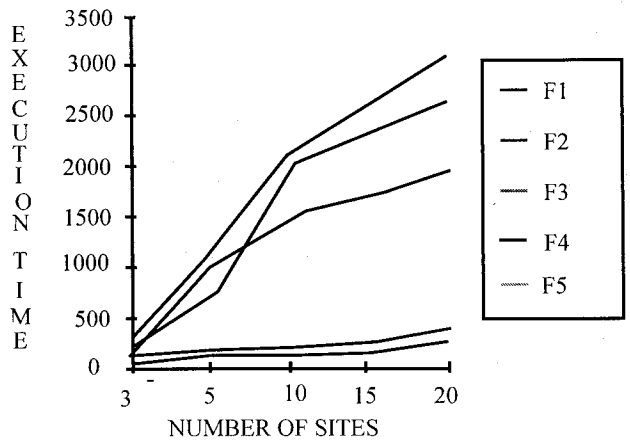


Figure 11. Results of the simulation for nf= 100, c_f= 3

5.5 Comparison with Other Works

We have tested our approach on the same architecture where [Liskov et al., 1998; Bartal 1998] tested their heuristics. To evaluate our algorithm we use the same criteria as them. These results are shown in the next sections.

5.5.1 File Replication

We have compared the performance of our approach with that of a replicated NFS server (called Harp System) that uses UNIX files directly [Liskov et al., 1998]. All our experiments were run in a system containing four servers. We have used the Andrew benchmark. Andrew benchmark runs a fixed set of operations intended to be a representative sample of the kind of actions an average user of the Andrew file system might performance. The table 1 shows the performance results (cost function value). As can be seen, Harp system has a worse performance than our approach when the load query is more than 20 seconds because the interference at the servers degraded its performance. For the other case, the Harp System has a

lower cost because it has a good performance over systems with a light load.

Load Query/second	Liskov et al. 1998 (Harp System)	Our approach
20	22	24
60	36	35
100	64	61

Table 1. Comparison of the results

5.5.2 File Decomposition

In this part, we compare our approach with [Bartal et al., 1998]. Their paper deals with the file allocation concerning the optimization of communication costs to access data in a distributed environment. They develop two distributed file allocation algorithms assuming existence of global information about the state of the network. A first algorithm is the BFS/BSP algorithm, and the other one is an agglomeration clustering algorithm. We have used a Nhfsstone benchmark to compare them. Nhfsstone benchmark is a standard synthetic benchmark designed to measure the performance of NFS file servers. It simulates multi-clients NFS file servers workload in terms of the mix of NFS operations and the rate at which the NFS clients make request to the servers. The table 2 shows the results. In this table, the first number is the value of the cost function and the second number is the average CPU time in seconds of each approach.

Number of sites	Number of query	BFS/BSP	Agglomeration Clustering	Our approach
5	5	60 (128)	66 (12)	64 (50)
7	7	56 (256)	62 (28)	60 (112)
10	10	50 (923)	59 (45)	56 (213)
12	10	44 (2312)	57 (112)	51 (523)
15	10	40 (4121)	55 (231)	45 (1443)

Table 2. Comparison of the results

In this case, our approach obtains good results with a low execution time. We have proposed an algorithm that is between BFS/BSP (which produce the lowest costs) and the agglomeration clustering approach (which produces the lowest CPU times). A tradeoff between the different criteria is necessary to improve our results.

Conclusion

The file decomposition and replication problems can be formulated using a cost function and solved using intelligent techniques. These problems include a fragments/copies assignment problem. In this work, we propose different cost functions for these problems and a heuristic algorithm based on Genetic Algorithms. We test the performance of our algorithm on different distributed architectures according to these cost functions.

The model introduced in this paper provides a common denominator for analysis and comparison on various proposed distributed system configurations, a tool to study the sensitivity of various parameters and constraints, and a method for evaluating and designing the growth potential of distributed systems. However, some related problems still requires further studies, such as file reliability, privacy, etc. All these problems are important consideration for optimal file management in distributed systems. Next work must consider these aspects. For the file replication problem, our approach not only generates the near optimal replicas allocation, also generates the optimal number of copies for each file. For the file decomposition problem, our approach not only generates the near optimal fragments allocation, also generates the optimal number of fragments for each file.

Experiments show that the quality of the solutions obtained by each cost function is similar independent of the characteristics of the platform or applications considered. In our study, F2 for the file replication problem and F3 for the file decomposition problem appear to give the best results, with a substantially shorter execution time. The cost value and execution time for F1 for the file replication problem, and F1 and F5 for the file decomposition problem are very large. The reason is that these cost functions use a large number of criteria. Normally, C_{li} has a large value and C_{no} has a little value in all cases. In general, it is necessary to determine the better combination of criteria to reach a suboptimal solution that improves the performance of the system and applications. In addition, we have shown our approach can be used in real system with a good performance because our approach considers several criteria at the same time. We can define the set of criteria to be used in a given situation. Next work will study the fragments replication problem and a new equation to calculate the optimal number of copies for each file based on predictions about the quantity of information to update or request. According to our models (see Eq. (6) and (17)), we consider different criteria, which can be conflictive

between them. A tradeoff between the different criteria is necessary to improve our results. Other works will consider the use of evolutionary multiobjective approaches to deal this problem [Aguilar et al., 1999].

Acknowledgments

This work was partially supported by CONICIT grant S1-95000884, CDCHT-ULA grant I-620-98-02-AA and CeCalCULA (High Performance Computing Center of Venezuela).

References

Aguilar, J. and Miranda, P., "Approaches based on Genetic Algorithms for Multiobjetive Optimization Problems", *Proceeding of the Genetic and Evolutionary Computation Conference*, IEEE Press, pp. 3-10, 1999.

Aguilar, J., "Optimizing the speed-up of Parallel Programs and Load Balancing in Distributed Systems", *International Journal of Computers and Applications*, Vol. 8, N. 8, pp. 117-122, 1998.

Aguilar, J. and L Kloul, "Estudio del Problema de Asignación de Tareas en los Sistemas Distribuidos: funciones de costo y métodos de resolución", *Revista Técnica de Ingeniería de la Universidad del Zulia*, Vol. 20, No. 3, pp. 203-213, 1997.

Aguilar, J. and E. Gelenbe, "Task Assignment and Transaction Clustering Heuristics for Distributed Systems", *Information Sciences: Informatics and Computer Science*, Vol. 97, No. 1, pp. 199-219, 1997.

Aguilar, J. and T. Jimenez, "A Processor Management System for PVM", *Lecture Notes in Computer Science*, Springer-Verlag, Vol. 1300, pp. 158-161, 1997.

Aguilar, J., "Un Estudio Exhaustivo del problema de Descomposición, Replicación y Asignación de Archivos, en los Sistemas Distribuidos usando Algoritmos Genéticos", *Jornadas Nacionales de Computación Paralela y Distribuida*, pp. 21-43, 2001.

Bartal Y., A. Fiat and Y. Rabani, "Competitive Algorithms for Distributed Data Management", ACM Symposium on Theory of Computing, pp. 34-39, 1998.

Chapman, B., P. Mehrotra, H. Moritsch and H. Zima, "Dynamic Data Distribution in Vienna Fortran", In *Proc. Supercomputing 1993,* pp. 284-293.

Chen, T. and J. Sheu, "Communication-Free Data Allocartion Techniques for parallelizing Compilers on Multicomputers", *IEEE Transaction on Parallel and Distributed Systems,* Vol. 5, N. 9, pp 924-938, 1994.

Gifford, D., "Weighted Voting for Replicated Data", *Journal of ACM,* Vol. 6, pp. 150-162, 1987.

Goldberg, D., "Genetics Algorithms in Search, Optimization and Machine Learning", *Addison-Wesley,* Reading, Massachusetts, 1989.

Kalns, E. and L. Ni, "Processor Mapping Techniques Toward Efficient Data Redistribution", *IEEE Transaction on Parallel and Distributed Systems*, Vol. 6, N. 12, pp. 1214-1222, 1995.

Knobe, M. and V. Natarajan, "Automatic Data Allocation to Minimize Communication on SIMD Machines", *Journal Supercomputing,* Vol. 7, pp. 387-415, 1993.

Yu-Kwong K., K. Kamalakar, I. Ahmad and N. Moon Pun, "Design and Evaluation of Data Allocation Algorithms for Distributed Multimedia Database Systems", *IEEE Journal of Selected Areas in Communications*, pp. 342-351, 1996.

Mahmood, A., H. Khan and H. Fatmi, "Adaptive File Allocation in Distributed Information Systems", *Informatica,* Vol. 18, pp. 37-46, 1994.

Lee, P., "Techniques for Compiling Programs on Distributed Memory Multicomputers", *Parallel Computing,* Vol. 21, N 12, pp. 1895-1923, 1995.

Lee, P., "Efficient Algorithms for Data Distribution on Distributed Memory Parallel Computers", *IEEE Transaction on Parallel and Distributed Systems,* Vol. 8, N. 8, pp 825-839, 1997.

Li, J. and M. Chen, "The Data Alignment Phase in Compiling Programs for Distributed-Memory Machines", *Journal Parallel and Distributed Computing,* Vol. 13, pp. 213-221, 1991.

Liskov B., S. Ghemawat, R. Gruber, P. Johnson and M. Williams, "Replication in the Harp File System", *Proc. Thirteenth ACM Symposium on Operating System Principles*, pp. 226-238, 1998.

Jose Lisandro Aguilar Castro *received the B. S. degree in System Engineering in 1987 from the Universidad de los Andes-Merida-Venezuela, the M. Sc. degree in Computer Sciences in 1991 from the Universite Paul Sabatier-Toulouse-France, and the Ph. D degree in Computer Sciences in 1995 from the Universite Rene Descartes-Paris-France. He was a Postdoctoral Research Fellow in the Department of Computer Sciences at the University of Houston among 1999-2000. He is an Associate Professor in the Department of Computer Science at the Universidad de los Andes and researcher of the High Performance Computing Center of Venezuela (CeCalCULA) and of the Distributed System and Microelectronic Center (CEMISID). Currently, he is the head of the Science and Technology Bureau of the Merida State, Venezuela.*