An Economic Contribution to Solve Load Distribution Problems

Joel Silva Vicente

Departamento de Informática. Universidade do Minho
4710-057 Braga, Portugal
joelvic@hotmail.com

Abstract. Due to the similarity found between an economy and a distributed computer system, microeconomics concepts and algorithms were introduced to bring a new approach to the traditional workload distributing that is inefficient in modern complex distributed systems. This new approach defines a distributed computer system as an economy where competition set prices for the resources in the system. Jobs compete for these resources by issuing bids, and the resource allocation decisions are made through auctions held by processors. In this way, competition and a price system limits the complexity of the allocation resources process.

1 Introduction

To achieve higher performances, the processors, the associated memory devices and inter-processor communication bandwidth must be efficiently allocated to the units of work the systems services.

This document introduces a new approach to design resource allocation algorithms for distributed computer systems. This new approach presents and uses concepts from microeconomics studies [1]. These concepts offer us a set of tools - for limiting the complexity of computer systems by decentralizing the control of resources - and a set of mathematical models that can yield several new insights into resource sharing problems.

There is also a similarity between multiple computer processor systems and human economies that we cannot deny. In an economy we immediately identify two types of agents: suppliers and consumers. A computer system has also suppliers, which are the processors and consumers, which are the units being performed (processes, jobs, transactions). Suppliers have scarce resources, as processors, memory and communication bandwidth, that consumers are interested in acquire.

In an economy, suppliers and consumers are independent economic agents that attempt to selfishly optimize their satisfaction. In this document processors and jobs will have the same behaviour and an effective global allocation of resources is achieved indirectly through selfish competition in an attempt to obtain the scarce resources. This selfish behaviour and competition of agents will be coordinated by the introduction of money and pricing, as in a normal human economy [1]. Each job is endowed with money that it uses to purchase required resources and each processor owns a set of resources, and charges consumers for the use of it resources. The price a producer charges for a resource is determined by its supply and demand of the agents for the resource. The price system ensures that a feasible allocation of resources is achieved. In conclusion, we assume that all interactions between agents are carried out through competition and a well structured system price [2].

Competition has many benefits in distributed systems and this approach effectively performs load distribution. The results are even better than the ones obtained with traditional non-economic load distribution algorithms.

Four main sections perform the structure of this document. Section 2 describes the model of the distributed computer system. In section 3, it is described the economy in detail and defined the behavioural rules of each type of agent. Section 4 contains an evaluation of our approach as applied to load distribution. Conclusions are presented in section 5.

2 Models and Reasons for Using it

2.1 Models

Consider the distributed system with N processors denoted $P_1,\ldots, Pn$. Each processor Pi has an associated processing speed parameter r_i, used to define the service time of a job on this processor [3]. If a job requires μ CPU seconds on a processor with $r_i = 1$, it will require μ / r_i on processor Pi.

These processors are connected by a point-to-point network defined by an edge set:

$$E= \{ (i, j, d): Pi, Pj \text{ are processors, } d>0 \}$$

An edge $e= (i, j, d) \varepsilon E$ represents a bidirectional communication link connecting processors Pi and Pj. The link has a delay of d seconds per byte.

Considerations to this model:

1. Jobs are submitted to all processors.
2. The resource demanded by a job is CPU time.
3. Communication bandwidth may be used by a job to migrate from one processor to another when looking for CPU time.
4. A job must always leave the system on the processor at which it was submitted.
5. The three parameters presented below are known.

There are three parameters associated with the job when it enters into the system:

1. μ – The CPU time of the job
2. ReqBc – The number of bytes need to describe the job. ReqBc must be sent when migrating to another processor;
3. RspBc – The number of bytes needed to describe the result of executing the job.

2.2 The Reasons for Using the Model

Literature has been presenting many load distributing algorithms that substantially improve the performance of simple models. However, these models face a major problem [3]. In unifying the allocation of processors and communication resources the traditional load distributing algorithms are underestimating the profound effect of the communication resources in the success of load distribution. In our economy the auctions and bids that decide CPU allocations determine the allocation of communication resources [4].

We immediately conclude that there are improvements that we can achieve with our economic approach. As we will demonstrate, our economic algorithms obtain better performances than non-economic load distribution algorithms, decrease the complexity and improve algorithmic structure.

3 The Economy

The economy that allocates resources in distributed system described in the previous section is completely defined by the actions of two agents. We consider jobs and processors the two agents [3].

The jobs have the following main characteristics:

1. Enter the system at all processors.
2. Each receives an initial allocation of money.
3. If job J requires μj CPU seconds, it must purchase $\mu j / r_i$ seconds on some processor Pi.
4. Are allowed to migrate through the system in search of CPU service but they must pay each time they cross a link.
5. They must return to the processor at which they entered.

The processor's main characteristics are:

1. Sell CPU time and communication bandwidth to jobs.
2. Establish the prices for their local resources.
3. Their goal is to maximize revenue. It does not attempt to distribute loads or minimize response times.

To make intelligent purchasing decisions, the jobs must have some information about the prices of remote resources. So each processor is allowed to advertise on the Bulletin Boards maintained on adjacent processors. BB entries at a processor *Pi* are only defined for *Pi* and its neighbours.

3.1 Jobs

With the amount of money endowed when entering the system and the information about the prices listed in the Bulletin Boards maintained by the processors, jobs will attempt to purchase resources and be serviced. This economic operation will have three phases:

1. Jobs will compute a budget set.
2. Jobs will compute a preference relation.
3. Jobs will compute a bid.

These three phases are described below.

Compute a Budget Set. Consulting the price information in the local bulletin information job J will define a budget set that is composed by the IDs of the processors at which J believes that it can afford service given its resource necessities and remaining funds. A budget set element is an ordered pair (K, Ck). K represents the processor where J believes that it can afford service, in this case processor Pk. The estimated cost for this budget set is Ck that is composed of three sub-costs:

1. The cost of buying μj / rk CPU seconds at Pk.
2. The cost to cross the link to get to Pk. This cost does not exist if J is already at Pk.
3. The cost to get from Pk to the processor at which J entered the system.

There is also a rule that jobs have to obey that is related with the third sub-cost. Each time a job J crosses a link between two processors *Pi* and *Pj*, it must keep some money to cross from Pj and *Pi* on the return trip to its origin after getting CPU service.

Computing Preferences. After computing his budget set a job must decide which ordered pair is best. The job will use a preference relation that will determine the best ordered pair

that fits his needs. For the solution to the load distribution problem it is considered three preference relations:

1. Price – the job wants to be serviced as cheaply as possible.
2. Service Time (ST) – a job prefers the ordered pair of the budget set at which it can be most quickly serviced.
3. Service Time vs. Price (STVP) – in this preference function a job considers both service time and price. A job will also place a relative preference of service time over a cost at a processor. This is given by the equation below (A is a weight giving relative importance to ST):

$$\min[\ Ck + A * STk\ ]$$

Bid Generation. At this moment job J has already defined its budget set and its preference relation. It is comfortable to estipulate where it wants to be serviced. The job J will submit bids for the resource it needs. When bidding for resources, jobs that are interested in remote CPU capacities must bid first for the links that will take them to their resource destination.

To calculate the price bid for the desired resource, job will consider the current price in the bulletin board plus a fraction of its surplus:

$$€\ \text{resource's price} * \text{resource} + \text{surplus} * Sj$$

Its surplus is determined by the difference between the job's wealth (mj) and the cost of the optimal element of the budget set (Ck):

$$mj - Ck$$

A job J's bidding rule has two parameters. The first is Sj, which determine J's use of surplus funds. The second parameter is a feedback factor δj. Each time j wins an auction it sets Sj = Sj – δj. When loses it updates Sj = Sj + δj. The use of feedback δj implements a simple learning mechanism. The goal is to learn the minimum use of surplus funds needed to win an auction. Using this approach, in periods when competition is fierce, Sj will increase and improve a job's chances of getting service. In times when there is little competition for resources, Sj will decrease and money will be saved.

3.2 Processors

The processors perform three functions:

1. Auction Resources.
2. Update Prices.
3. Advertise.

In the next section will be described and explained these functions.

Auction Resources. Since several jobs demonstrate their will for a resource the processor will have to hold an auction to determine which job gets the resource. In this economy were implemented two auction models: Sealed Bid Auction and Hybrid Auction.

Under a sealed bid auction model all bids for a resource are opened and the job that bid the highest unit price (total amount of money bid divided by the quantity demanded) wins.

The Hybrid Auction is a mix between an English Auction and a Dutch Auction. The Hybrid Auction tries to find the highest price that can be charged for a resource by increasing the asking price when a bid is submitted (English Auction), and decreasing the asking price where no bid is submitted (Dutch Auction).

Update Prices. When a processor sells a resource to a job, the corresponding entry in the bulletin board is set to the unit price of the sale.

The main idea of this pricing policy is that the price charged by a processor for a resource is the maximum price that some job is willing to pay.

Advertise. Based on a local price change, a processor may send an advertisement message to its neighbours.

This policy maximizes the amount of information that is available to the jobs.

4 Performance Experiments

The load distribution problem is to design algorithms that minimize the mean job waiting time by migrating the jobs to balance the workloads of the processors nodes. Each job independently computes the best place (node) to be served based on its preferences and wealth, and the resources prices. The main goal of each processor (node) in the economy is to maximize revenue.

To estimate the quantitative and qualitative behaviour of the load distribution economy it was implemented the behavioural rules for the processors and jobs on Columbia University's Network Simulation Testbed (NEST) [3].

The distributed system simulated was a nine processor mesh (Figure 1) with the following characteristics:

1. The processing rate of each processor is 1.0.
2. Each communication link has a capacity of 1 Mibt/s.
3. Jobs arrive at all processors in the system.
4. The jobs have a mean service time of 30 ms.
5. The arrival process was assumed to be with the same rate at all processors.
6. The ReqBc and RspBc parameters are the same for all jobs and fixed at 1000 bytes.
7. Each job that enters the system has € 5.00, without considering its CPU demand.

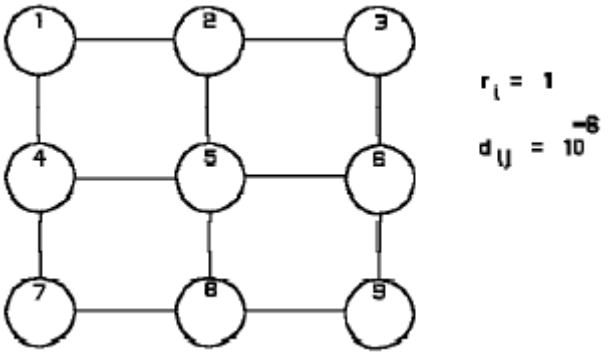

Fig. 1. A simple distributed system. From [3], with kind permission of the author.

It is easy to observe that this system described is very homogeneous. Load distribution algorithms are least effective in completely homogeneous systems. This system provides a stern test of the effectiveness of applying economic concepts to the load distribution problem.

The main performance metric studied was the mean job waiting time (response time – CPU service demand). Figure 2 plots the mean job waiting time versus utilization under the sealed bid auction model for each of the three job preference relations. The difference for Figure 3 is the use of the hybrid auction model. In both figures, the waiting times are

compared to the case of no load distribution (an M/M/1 queuing system using shortest-job-first scheduling (SJF)) and the case of theoretically optimal load distribution (an M/M/9 system using SJF scheduling). The M/M/9 is showed for comparison purpose only.

The main conclusions that are offered by these figures are easy to identify: with economic algorithms it is achieved an effective overall allocation of resources. All six economies are substantially better than the no load distribution case. Another conclusion from the figures is the little difference between the performance levels obtained by the three preference relations. However is showed that under the sealed bid model, Service Time is marginally better and in the Hybrid model Price is the best.

However the way in which the economies balance the loads is remarkably different has it is showed by figures 4 and 5 that represents the mean job migration distance for the three preferences relations in each auction model defined. The migration distance is the number of links a job crosses before receiving CPU service on some processor. So a conclusion is obtained: jobs with response time preference migrate less compared to the jobs that have price preferences.

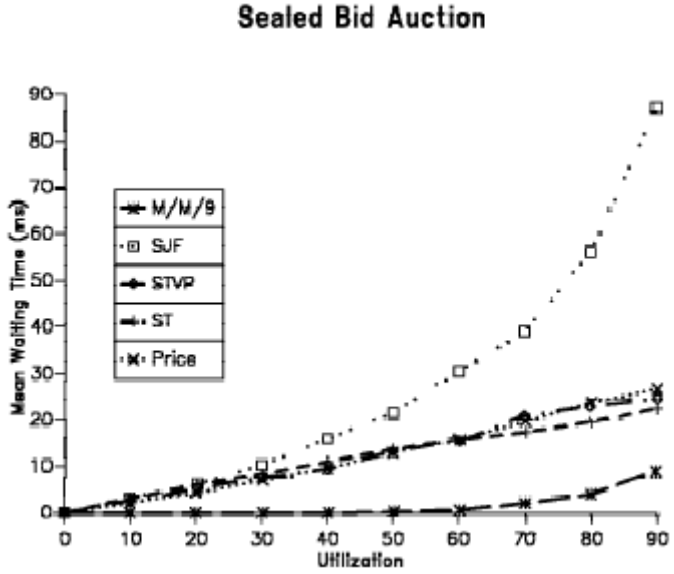

Fig. 2. Job Waiting Times. From [3], with kind permission of the author.

The reason for this behaviour can be found on this explanation: in a mesh topology, each processor has at least two neighbours and when a job is searching for a cheap processor for its CPU demands it will attempt to migrate until it find him. This phenomenon causes the high migration rates in the system when Price preference relation is assumed by the jobs, as in this case. Due to this fact the system does not become unstable but this high migration rate will increase the price for communication bandwidth which will turn it less attractive. So the shorter jobs, that have a smaller CPU demand and have been allocated the same € 5.00 as longer jobs, can spend more for communication links and will outbid the longer jobs. This situation will move the relatively wealthy short jobs to the cheapest (least loaded) processor.

When the Service Time is used, a job wants to migrate only if the local processor is not in its budget. There is no reason to migrate since all processors have the same processing rate and there is a delay associated with migrating. The jobs forced to migrate are considered poor jobs, it means they have large CPU demands and are who can best afford the overhead incurred by migration.

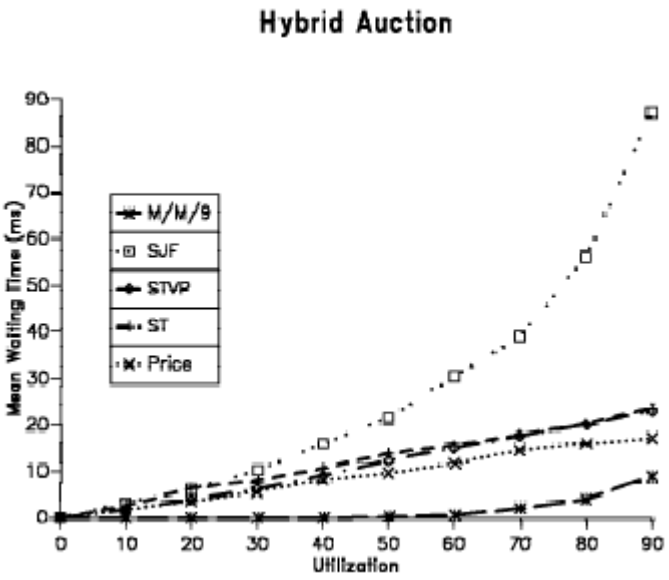

Fig. 3. Job Waiting Times. From [3], with kind permission of the author.

In the STVP preference, the behaviour of the economy can be controlled by the choosing the constants that define the STVP preference. This span the spectrum of possible load distribution strategies, and provide flexibility that allows the implementer of the load distribution algorithms to choose a particular point in the spectrum by setting two parameters. In this way when communication (migration) is very fast, Price is the best preference since it comes closer to the optimal M/M/N system. If however communication is relatively slow, Service is better preference due to its low migration rates.

Choosing between Sealed bid an Hybrid auctions is difficult. But when the mean job service time is small, Sealed bid auctions are better due to their lower overhead. When the mean service is large, then the Hybrid auctions become more attractive since this model does a better job of allocating resources to richer jobs.

As it was showed these economies achieve a global allocation of resources and substantially improve performance over systems in which load distribution is not used. However is essential to demonstrate how well these load distribution algorithms compare with traditional algorithms to solve the load distribution problem.

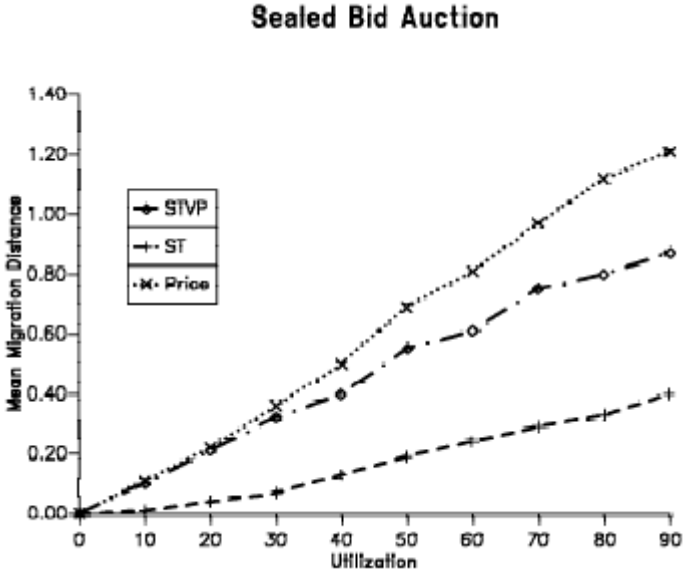

Fig.4. Job Migration Distance. From [3], with kind permission of the author.

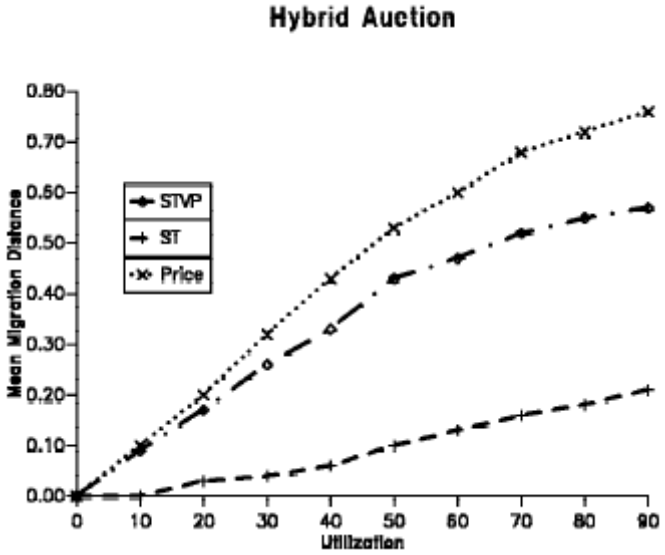

Fig. 5. Job Migration Distance. From [3], with kind permission of the author.

4.1 A Traditional Algorithm

The traditional algorithm chosen to compare and demonstrate the effectiveness achieved by the economic load distribution algorithms was the Livny's HOP 1 algorithm [5]. There are some reasons for this choice:

1. It is one of the few algorithms that has been designed for point-to-point topologies.
2. The complexity of this algorithm is roughly equivalent to the complexity of load distribution economies.
3. Livny's algorithm is extremely general.

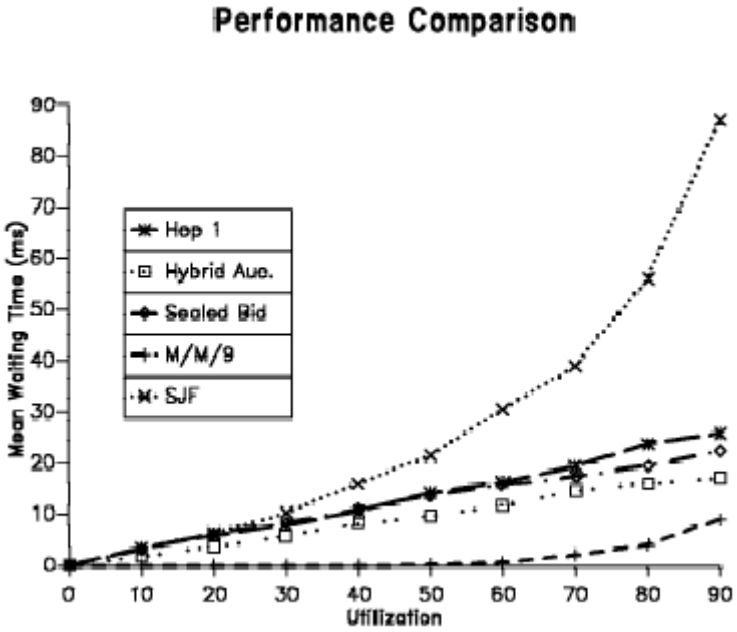

Fig. 7. Economic Vs. Non-Economic Algorithms. From [3], with kind permission of the author.

One of the characteristics of this traditional algorithm is the fact of consider the CPU service times of the jobs unknown. It was considered that for a fair comparison the HOP 1 algorithm should be modified and should use the available service times information.

The sum of the CPU service demands of the jobs in *Pi* represents the load of processor *Pi.* The load of *Pi* also includes the jobs currently in *Pi* and the jobs that *Pj* decided to send to *Pi*. *Pi* broadcasts all load changes to its neighbours.

Pi considers migrating jobs if the number of queued is greater than some threshold T. When this situation occurs *Pi* chooses the neighbouring processor that is least loaded but if this processor is more heavily loaded than *Pi* the HOP 1 algorithm terminates.

After choosing the target processor, *Pi* must decide what the local job to migrate is. It was implemented two job selection policies: First Fit and Best Fit. *Pi* also uses a SJF scheduling algorithm to allocate the CPU; the jobs are queued from smallest CPU demand to largest CPU demand. Using the first fit policy *Pi* will send to target processor the smallest job. Without reversing the load imbalance the largest job will be sent to the target processor under the best fit policy. *Pi* executes the HOP 1 algorithm repeatedly until no productive migrations are possible.

After some HOP 1 algorithm simulations it was determined that a threshold of T=1 and best fit policy were the factors to achieve the best performance and in figure 6 is represented the mean transaction waiting time of HOP 1 algorithm as a function of system utilization. In this figure are also represent another curves: a situation with no load balancing and optimal load balancing (M/M/9); a Sealed Bid-Service Time economy and a Hybrid Auction–Price economy are represented too. This way figure 6 represents a performance comparison between economic and non-economic algorithms. And the conclusions are easy to get: an economy in which hybrid auctions are used by the processors and the jobs use the price preference achieves better performance than HOP 1 algorithm for all utilizations. An economy in which sealed bid auctions are used by the processor and jobs use the service time preference achieve marginally better performance at higher utilizations but however is as good as HOP 1 algorithm at low utilizations.

5 Conclusions

This communication analysed a resource distribution problem in a distributed computer system. It was suggested that the distributed system could be structured as a society with competing microeconomic agents and concepts from microeconomics were also introduced: competition, independency, prices, agents, the laws of supply and demand and selfish behaviour.

In this system a globally effective allocation of the resources is indirectly achieved through the selfish optimization and competitive behaviour of the economic agents that indirectly and without knowing obey the laws of supply and demand.

To validate the results, economies that perform load distribution were implemented and tested. Jobs and processors are economic agents that interact in order to satisfy their necessities: jobs rate possible allocations of resources and submit bids to obtain the resources needed. Processors use auction models to allocate the resources. As well as in a economy, processors, which we defined as suppliers, are concerned with maximizing revenue and do not attempt to balance loads or explicitly improve performance.

All of the economies, that apply economic algorithms, substantially improve mean job waiting time when compared to representative non-economic load distribution algorithm. Another contribution of the economic algorithms is related to the possibility to limit complexity by converting a complex global allocation problem into a set of smaller,

independent problems and also bring stability. There is a possibility to easily adapt to the ratio of communication to computation speeds.

The results presented in this document suggest several possible avenues for further research.

References

[1] Frank, R.: Microeconomia e Comportamento. McGrawHill (1994)

[2] Hildenbrand W., Kirman A.P.: Introduction to Equilibrium Analysis. Amsterdam – North-Holland Publishing Co. (1976)

[3] Ferguson, D.: The Application of Microeconomics to the Design of Resource Allocation and Control Algorithms, PhD Thesis, Columbia University, New York (1990).

[4] Livny M., Melman M.: Load Balancing in Homogeneous Distributed Systems. Proceedings of the ACM Computer Networking Performance Symposium (1982).

[5] Ferguson, D.: Dynamic Distributed Load Sharing Algorithms – A survey. Computer Science Department. Columbia University (1985).

[6] Ferguson, D. Yemini, Y., Nikolau, C.: Microeconomics Models for Resource Sharing in Multicomputer, IBM T.J. Watson Research Centre (1987).

[7] Wang Y., Morris R.J.: Load Sharing in Distributed Systems. IEEE Trans. Computers, Vol. C-34, N. 3 (1985)