

A DISTRIBUTED STRATEGY FOR
RESOURCE ALLOCATION

Ann Kratzer

Technical Report PCS-TR86-118

A Distributed Strategy for Resource Allocation

Ann Kratzer

Dartmouth College

0) Introduction

Research and development in the area of computer networking and communication protocols have made possible the commercial production of distributed resource sharing systems. Such systems are comprised of some number of autonomous host computers that can communicate reliably with one another using a computer communications network. Users of the system can access resources, both hard and soft, irrespective of where objects may be physically located in the system.

A number of new issues arise in the management of such a system that are the responsibility of the (now distributed) operating system. Among these issues is the resource allocation problem. Generally, some of the system's resources can be selectively assigned to any site in the system. The objective in selecting a site for a particular resource unit can be either to improve the service provided to the users or to optimize the utilization of computing resources. Regardless of the kind of resource involved or the performance measure of interest, the goal is always to optimize some cost that measures the 'goodness' of the allocation of resources.

In this paper, we present a decentralized algorithm for determining how resources should be allocated to sites. This algorithm is general in the sense that it can be used for allocating different kinds of resources. This algorithm can be applied either statically, used on a quiescent system, or used dynamically as the system runs. Throughout, we consider only systems in which a resource has a fixed cost associated with it regardless of where it may be located (i.e. the system is homogeneous.)

The cost of any technique for coordinating activity in a distributed system must include both the classic complexity measures of time and space. For distributed systems, the amount of communication required by the algorithm constitutes an added

cost which must be considered. The algorithm presented in this paper requires only a modest amount of communication. In addition, it can be shown that the algorithm terminates in a reasonable amount of time, and produces allocations whose 'badness' is bounded.

Previous work in the area of resource allocation strategies has proceeded along two different directions: First, there has been a significant amount of work done in the area of load levelling[1-7]. In load levelling, the objective is to balance the computational load created by the processes in the system between the different CPU's. The second area of research has studied how files can best be allocated to sites[8]. In both cases, a wide range of such problems have been studied. Typically, assumptions are made to simplify or restrict the problem. These assumptions include: studying homogeneous systems in which all the sites are identical; developing static algorithms that are run either once or on a quiescent system; varying the objective function that measures the goodness of the allocations.

1) Definitions

Throughout this paper the following definitions are used:

- 1) The system consists of a total of S sites. Each site consists of a single, autonomous computer.
- 2) The resource units that must be allocated will be referred to collectively as resources. There are a total of R resource units.
- 3) Each resource unit has associated with it a cost/size. The cost/size for the i th resource unit is designated by s_i .
- 4) An allocation of resources to sites is a partition of the resources that is pair wise empty and whose union is exactly the set of all resources.

We are not concerned with what the resources are. For our purposes, the important property of a resource unit can be abstracted by its cost. The algorithm presented in this paper only attempts to balance the per site costs evenly. For instance, the resources may be files. Two possible costs that may be ascribed to a file are its size or its reference frequency. When the cost gives the file's size, the allocation

technique will attempt to balance the storage requirements evenly across the system. When the cost is the file's reference frequency, the expected number of accesses to each site is balanced by the algorithm. User jobs constitute another kind of resource. The cost of a job can be either its expected running time, memory requirements or some more complex measure of the resources needed by a job. When applied to jobs, the allocation technique will balance expected running times, memory requirements or other job resource measure.

2) Problem Statement

The allocation technique presented in this paper attempts to balance evenly the distribution of resources in the system. The cost of the resources allocated to a particular site is given by the sum of the resource costs assigned to the site. Intuitively, the best allocation is the one with the smallest site cost difference. Formally:

Definition: Let Resources_a be the set of resources allocated to site a. Then the cost of allocating Resources_a to site a is given by $Cost_a = \text{SUM} \{i \text{ in Resources}_a \mid s_i\}$.

Thus, the per site cost of an allocation is the sum of the costs of the resources allocated to the site. The best system wide allocation is determined by examining the pair wise differences in per site cost:

Definition: The optimal allocation of resources to sites is the one which $\min \{ \text{by choice of allocation} \}$
 $\max \{ \text{choice of sites } a, b \} \Delta_{a,b} =$
 $| Cost_a - Cost_b |$.

The balance of a particular allocation is measured by the maximum difference in per site allocations occurring between a pair of sites. This maximum difference is a conservative measure because it identifies the worst imbalance in the allocation. The best allocation is the one with the smallest worst pair imbalance.

Efficient algorithms of any kind are not known for this problem[9]. The problem inherently possesses a very rich combinatorial structure with size $S**R$. Any fast algorithm for determining the best allocation would have to choose between the exponential number of possible allocations. Such a fast

algorithm exists for this problem iff fast algorithms exist for a number of other problems such as the travelling salesman problem. Thus, any practical algorithm for the allocation problem must be heuristic in nature. Such an algorithm attempts to determine a reasonable allocation for most instances of the problem. The distributed technique given in this paper is heuristic.

3) The General Form of the Distributed Algorithm

The distributed allocation algorithm is iterative in form. It repeatedly optimizes unbalanced pairs of sites until all pairs of sites are balanced:

while there is any pair of sites that are not balanced do

 select an unbalanced pair of sites to optimize

 Optimize the allocation of this pair by
 moving resources units from the site with higher
 site cost to the site with lower site cost.
 The optimization of the two sites stops when the two
 sites are balanced.

end

General Form of the Allocation Algorithm

Note that no rules are given for selecting which pair of unbalanced sites to optimize and which resource units to move in a pair wise optimization. Also, the concept of balance is not defined. Such an algorithm is reasonable only if the following conditions are met:

- 1) There must be a communications non-intensive way to test for balance.
- 2) The pair wise optimization must be done with very little communication of control information.
- 3) There must be some guarantee as to how unbalanced the resulting allocation can be.
- 4) The algorithm must terminate in a reasonable amount of time.

After defining what is meant by balance, the algorithm can be shown to meet requirements 1-3. Termination can be guaranteed when a particular rule is used for selecting the pair of sites to be optimized. Note that no assumption is made on which resource units to move in a pair wise optimization. The results outlined below support this claim.

Definition: Two sites are balanced when no single resource unit can be moved from the site with larger site cost to the site with lower site cost and not increase the pair's Delta.

Intuitively, the definition of pair wise balance says that two sites are balanced when the re-allocation of any resource unit from the larger site to the smaller site would worsen the site pair balance. Pair wise balance can be tested with relatively little communication:

Theorem: Two sites, a and b where $\text{Cost}_a > \text{Cost}_b$, are unbalanced iff $\text{Cost}_a > \text{Cost}_b + \text{min}_s_a$, where min_s_a is the size of the smallest sized resource unit allocated to site a.

This theorem implies that pair wise balance can be tested by broadcasting for each site, a, its over all cost, Cost_a , and the size of the smallest resource unit on site a, min_s_a . Therefore, the amount of control communication needed to perform the (distributed) while test of the algorithm is $2*S$. In addition, stopping the optimization of a pair of sites can be done with no extra control communication. The larger cost site must know the site cost of the smaller site from the while test. As optimization proceeds, the larger cost site can update its copy of the smaller site's cost.

It can be shown that the resulting allocation can not be infinitely unbalanced:

Theorem: The allocation produced by the algorithm must have its maximum system wide Delta $< \text{max}_s$, where max_s is the largest resource unit size.

To establish the termination result it is necessary to require that the unbalanced pair optimized be the one with the largest pair wise delta. With the assumption the following result can be established:

Theorem: When the largest delta unbalanced pair is next optimized, the algorithm terminates in time $o(S*R)$.

4) Simulations Studies

The algorithm given in the previous section is not completely specified. The rule used in doing a pair wise optimization that governs which resource is next moved is needed. Two pair wise optimization techniques are being investigated by simulation (Theoretically, they must behave as outlined in the preceding section.) We wish to determine how the convergence rate of the algorithm for both pair wise optimization rules is effected as a function of:

- 1) the number of sites in the system,
- 2) the number of resource units,
- 3) the initial allocation of resource units and
- 4) the statistical distribution of resource unit sizes.

5) Conclusions

We have identified a class of resource allocation algorithms that can be implemented efficiently on a distributed system and that can be shown to terminate in a reasonable amount of time and that produce allocations whose maximum unbalance is bounded. Simulation studies will attempt to measure the performance of two particular algorithms in this class to determine how the realized performance is effected by variations of certain key parameters.

Bibliography

- 1) H. S. Stone, "Multiprocessor Scheduling with the Aid of Network Flow Algorithms," IEEE Transactions on Software Engineering, SE-3, 1 (January 1977,) 85-93
- 2) H. S. Stone, "Critical Load Factors in Two-Processor Distributed Systems," IEEE Transactions on Software Engineering, SE-4, 3 (May 1978,) 254-258
- 3) W. D. Roome and H. C. Torng, "Modelling and Design of Computer Networks with Distributed Computing Facilities," Proceedings of the 1974 Symposium on Computer Networks, 30-38
- 4) L. J. Miller, Optimal Scheduling of Task Groups on Tightly Coupled Multiprocessors, State University of New York at Buffalo Technical Report No. 149, January 1979
- 5) L. Casey and N. Shelness, "A Domain Structure for Distributed Computer Systems," Proceedings of the Sixth ACM Symposium on Operating System Principles, November 1977, 101-108
- 6) Ann Kratzer and Dan Hammerstrom, "A Study of Load Levelling," COMPCON, Fall 1980, 647-654
- 7) S. B. Wu and M. T. Liu, "Assignment of Tasks and Resources for Distributed Processing," COMPCON, Fall 1980, 655-662
- 8) Lawrence W. Dowdy and Derrell V. Foster, "Comparative Models of the File Assignment Problem," Computing Surveys, 14, 2 (June 1982,) 287-313
- 9) Michael R. Garey and David S. Johnson, Computers and Intractability - A Guide to the Theory of NP-Completeness, W. H. Freeman and Company, 1979