

Optimal Tree Structures for Large-Scale Grids

J. Palmer I. Mitrani

School of Computing Science, University of Newcastle, NE1 7RU, UK.

jennie.palmer@ncl.ac.uk isi.mitrani@ncl.ac.uk

Abstract

We consider the problem of how best to structure and control a Computing Grid containing many processors. The performance trade-offs associated with different tree structures are evaluated by analysing appropriate queueing models. It is shown that, for a given set of parameters and job distribution policy, there is an optimal tree structure that minimizes the overall average response time. This is obtained numerically through comparison of theoretical average response times. A simple heuristic policy is shown to perform well under certain conditions.

Keywords: Optimal tree structures, Grid computing, Dynamic programming, Heuristic policies.

1 Introduction

In the provision of a *Grid service*, a provider may have heterogeneous clusters of resources offering a variety of services to widely distributed user communities. Within such a provision of services, it will be desirable that the clusters are hosted in a cost effective manner. Hence, an efficient structure of the available resources must be decided upon. A 'flat' structure, where a single master node controls all processors and decides where incoming jobs should be executed, is not always efficient. The master node can become easily overloaded when demand is high, leading to poor processor utilization and large response times. On the other hand, a tree structure involving a hierarchy of master nodes controlling subsets of processors, avoids the bottleneck problem but introduces additional processing and transfer delays.

The problem of load balancing across a network of available resources has been discussed in distributed systems literature for more than two decades. A comprehensive discussion of diffusion techniques for dynamic load balancing can be found in [2]. The objective of load balancing is to enable each available resource to perform an even share of the network load. Work is quantified in terms of *tasks*, each of which require an amount of processing time to be completed. Tasks may be reallocated from one processor to another, balancing the load across multiple machines. The aim is to minimise the overall execution time, using specified load balancing algorithms. A description of customized load balancing strategies for a network of workstations is given by Zaki et al [3]. Hine and Holzer [4] present their results on different scheduling algorithms for load balancing. Most recently, Houle et al [5] consider algorithms for static load balancing on trees, assuming that the total load is fixed.

In all of these studies, the network configuration is treated as fixed and immutable.

Our approach considers the dynamic reconfiguration of the underlying tree structure as load changes, rather than the reallocation of jobs across a fixed tree. This does not appear to have been studied before. We consider a system consisting of a pool of N machines, and compare the theoretical average response times for different network configurations. It is assumed that the average service time of a master at a given node of the tree (i.e., the time taken to decide where to send the job for processing), is proportional to the number of children of that node, which we will refer to as its *dependents*. This is a reasonable assumption, given that current tools for building grid applications query the available resources before routing a job. One example would be the Condor [6] which uses the ClassAd mechanism to match resource specifications to job requirements.

In addition, independent transfer delays are assumed when sending jobs from one node to another. Different policies for distributing jobs among the nodes in the tree (both master and processing) are considered. The model assumptions are described in section 2. Computation of the optimal tree structure is described in section 3, while section 4 presents some numerical analysis results. Section 5 summarizes the results and discusses future work.

2 The model

Our model considers i levels of master nodes. A *flat* structure where $i = 1$ is illustrated in figure 1. A preliminary tree structure to be optimized with $i = 2$ is illustrated in figure 2. Jobs arrive according to an independent Poisson process with rate λ . Their required service time is distributed exponentially with mean $1/\nu$. The average service rate μ_i of a master at a given node of the tree at level i is inversely proportional to the number of dependents of that node, i.e.

$$\mu_i = \frac{c_i}{n}, \quad (1)$$

where c_i is a constant of proportionality at level i and n is the number of dependents. It is also assumed that these constants of proportionality may be different at different levels of the tree. This is particularly likely for a large scale Computing Grid, across which the querying of geographically distant nodes at the master level will take more time on average than a cluster manager querying its local service nodes. In addition, independent transfer delays T_i are assumed when sending jobs from one node to another. Jobs arrive into a FIFO queue in front of each master node. Following a routing decision, the master node distributes jobs among its dependents. Different policies for distributing jobs among the nodes in the tree have been considered. These include

1. Each dependent has a separate queue; the master places new jobs into those queues in random order.
2. Each dependent has a separate queue; the master places a new job into the queue which is currently shortest.
3. Each dependent has a separate queue; the master places new jobs into those queues in cyclic order.
4. Dependents at a final service cluster level have a joint queue

Simulation results have shown that, at the service cluster level, n parallel separate queues using policies 2 or 3 may be approximated by a single $M/M/n$ queue (described further in section 3) so long as n is reasonably large. This assumption is appropriate for any realistic Grid hosting environment. We continue to formulate the model using an $M/M/n$ queue at each final sub-cluster. Each master node will perform a query to determine where a job should be routed, the service rate of which is inversely proportional to the number of its dependents. If a master node is also head of a final sub-cluster, the job service is modelled by an $M/M/n$ queue with service rate ν .

How the routing decision is made will affect average response times, particularly for a heavily loaded system. Simulations have been used to compare pure *random* choice with that of querying the shortest queue or using a cyclic order. The latter two methods of decision making give very similar average response times, both performing significantly better than random choice. So far, we have performed detailed analysis for a model using only random choice. The other methods are for future work, as outlined in section 5.

Any server may be assigned as either a master node or a service node. Known theoretical results [1] for average response times have been used to compare different configurations of the tree structure shown in figure 2. The service nodes are split into k different sub-clusters, each with its own master node responsible for routing an incoming job to an appropriate service node in its cluster. Of interest is the particular value of k , which for a given number of servers N , arrival rate λ , service rate ν , transfer times T_i and constants of proportionality c_i , minimizes the overall average response time of the system.

3 Computation of The Optimal Tree Structure

The average response time may be evaluated as follows. The master node at level i querying its dependents is an $M/M/1$ system, with service rate μ_i inversely proportional to the number of dependents. The average response time for an $M/M/1$ system with offered load ρ_i (which is in this case equal to λ/μ_i , $i = 1, 2$) is given by

$$W_i = \frac{1}{\mu(1 - \rho_i)}. \quad (2)$$

Each final service cluster of n nodes is modelled as an $M/M/n$ system. The response time for an $M/M/n$ system with arrival rate λ and offered load ρ (which now is equal to λ/ν) is given by

$$W_{final} = \frac{1}{\lambda} \left[\sum_{j=1}^{n-1} \frac{\rho^j}{(j-1)!} + \frac{\rho^n(n^2 - n\rho + \rho)}{(n-1)!(n-\rho)^2} \right] \left[\sum_{j=0}^{n-1} \frac{\rho^j}{j!} + \frac{\rho^n}{(n-1)!(n-\rho)} \right]^{-1}. \quad (3)$$

A flat structure, as shown in figure 1, is possible so long as the master node is not saturated, i.e. $\rho_1 < 1$ where $\rho_1 = \lambda/\mu_1 = \lambda N/c_1$. So, for a non-saturated flat structure we require $c_1 > \lambda N$. When this condition holds, the average response time for the flat structure is given by

$$W = W_1 + W_{final}. \quad (4)$$

If the condition $c_1 > \lambda N$ does not hold, the master node in the flat structure will be over-loaded. It is now necessary to introduce a second network layer of k master nodes,

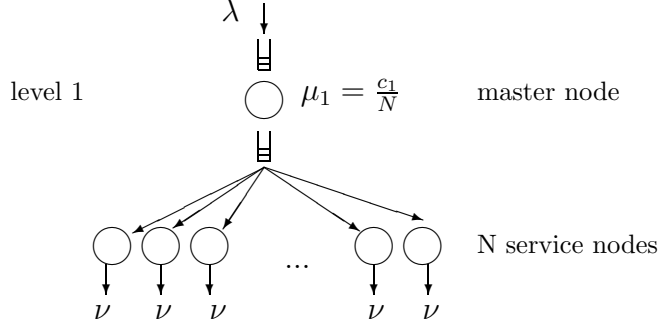


Figure 1: Flat structure

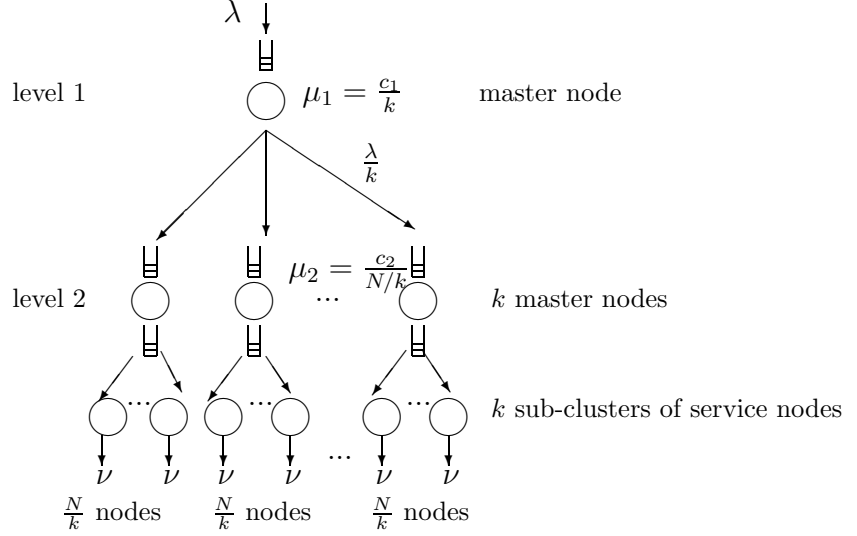


Figure 2: Service cluster split into k sub-clusters

as shown in figure 2, thereby reducing the service time for a master node to query its subset of dependents. The average response time will now be given by

$$W = W_1 + T_1 + W_2 + W_{final} . \quad (5)$$

The objective is to minimise the right-hand side of (5) with respect to k . The optimal value of k is difficult to obtain in closed form, but is easily and quickly computed numerically. At a service cluster, the arrival rate is λ/k and the available service rate is $N\nu/k$, so the offered load is $\lambda/N\nu$ regardless of k . If the overall system is not to be saturated, we require first $\lambda/N\nu < 1$. Then, for a given parameter set, k has upper and lower bounds so that no master node within the network becomes saturated. For the tree structure to be possible, we require $\rho_i < 1$ at each master node. At the first master node this gives $k < c_1/\lambda$. At the next master node this gives $k > \sqrt{\lambda N/c_2}$. Hence, possible values of k are within the range

$$\sqrt{\frac{\lambda N}{c_2}} < k < \frac{c_1}{\lambda} . \quad (6)$$

Average response times for each value of k within this range are evaluated and compared to find the minimum response time, and hence the optimal value for k . This gives the optimal network configuration with a single layer of master nodes. As λ increases, the range of possible values for k given in (6) diminishes and eventually there will be no acceptable value of k . The system will be saturated and will not cope with the demand. At this point, another network layer could be introduced, and this is a topic for future work discussed in section 5.

3.1 A simple heuristic

Consider the total offered load, $f(k)$, at the level 1 master node and one of the level 2 master nodes. It is given by

$$f(k) = \frac{\lambda k}{c_1} + \frac{\lambda N}{c_2 k^2} + \frac{\lambda}{N\nu} . \quad (7)$$

This total load may be minimized with respect to k to find an initial value for k given the number of service nodes N and the constants of proportionality c_1 and c_2 , which represent the speed at which routing decisions may be made at different network levels. Differentiation of (7) and equating with zero to determine a minimum leads to

$$k = \sqrt[3]{\frac{2Nc_1}{c_2}} . \quad (8)$$

This value for k matches the numerically obtained optimal value for k quite closely when the overall network load is low, and may be used as a starting configuration of the network. The numerical calculation to obtain the optimal value for k described in the previous section is relatively simple and can be used for dynamic network configuration.

4 Results

Graphs of average response time as k varies clearly show the benefit of carefully choosing the configuration of the tree. Figure 3 shows the average response time for a system with $N = 100, c_1 = c_2 = 100, T_1 = 0.001, \lambda = 8$ and $\nu = 0.1$. Hence, this system is reasonably heavily loaded at 80%. A 'flat' structure is not feasible, since $c_1 < \lambda N$. Possible values for k from (6) are $3 \leq k \leq 12$. The optimal value for k which minimises the average response time is $k = 4$. The predicted value from (8) is $k = 6$. Hence, while the simple heuristic may give a good starting configuration for a network of a given size and defined speed of decision making, the benefits of dynamic configuration using the numerically obtained optimal value for k are clear.

Figures 4 and 5 compare the optimal value of k with that predicted by the simple heuristic given in (7). In figure 4, the load is fixed at 80%, $\nu = 0.1, T_1 = 0.01, c = 1000$. The number of servers N increases, with the arrival rate λ increasing appropriately as N increases to keep the load at 80%. We observe the simple heuristic consistently predicting a higher value for k than the optimal at this network load. However, the difference in average response time for the different values of k is minimal. Conversely, in figure 5, the number of servers is fixed, $N = 100$, and the percentage load increases with λ . Now, $c = 100, \nu = 0.1$ and $T_1 = 0.001$. The simple heuristic predicts $k = 5$, which we observe is an under-estimate when the load is low, and an over-estimate when the load is high.

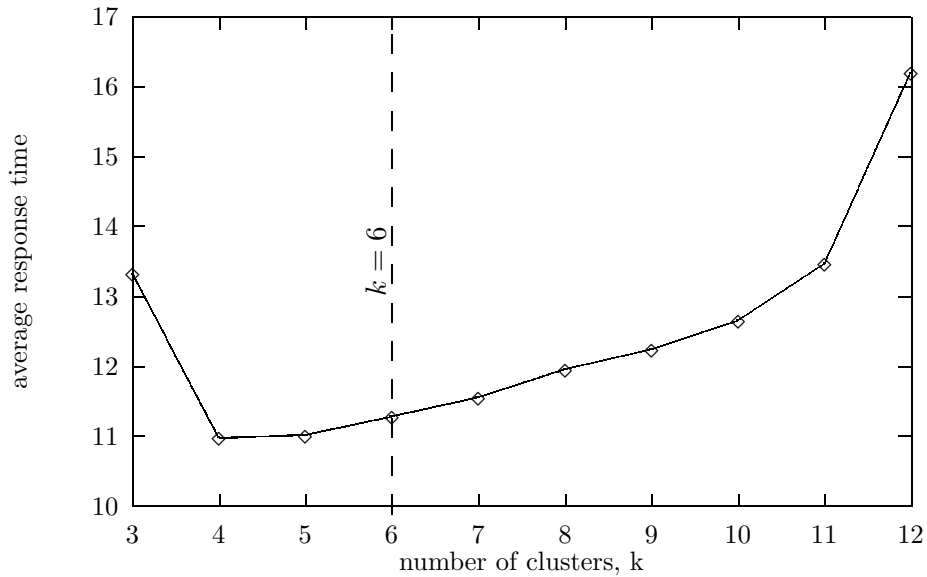


Figure 3: load = 80%

5 Conclusions and Future Work

A problem of interest in the area of distributed processing and dynamic Grid provision has been examined. Through numerical analysis, a tree structure for a Computing Grid is decided upon which minimizes the overall average response time. The encouraging results of figures 3 and 4 suggest that the benefits of dynamic reconfiguration of the network as load changes will reduce long term average response times and help make the Computing Grid more commercially viable. A simple heuristic is available for the configuration of a network with a given number of service nodes and specified decision making speeds. This heuristic gives an adequate configuration which may be improved upon as the network load increases.

As network load increases further, a further tier of master nodes will be necessary to avoid saturation. This is a problem of considerably increased complexity, and would be a worthy object of further study.

Our model assumption that the routing decision made by a master node is one of random choice is not particularly realistic. Different models need to be considered in more detail, and their effect on overall average response times noted. It is more likely that a master node would make an *intelligent* decision based on shortest queue or round-robin. Theoretical results are available for the round-robin method of routing decision. Early simulation results imply that round-robin compares favourably with querying for the shortest queue. More work is required in this area.

As yet only one job type has been considered. This is a simplification of Grid provision, where clusters of servers may provide a variety of services to widely distributed user communities. A natural generalization of this problem would be to consider more than two job types, again leading to a significantly more complex model.

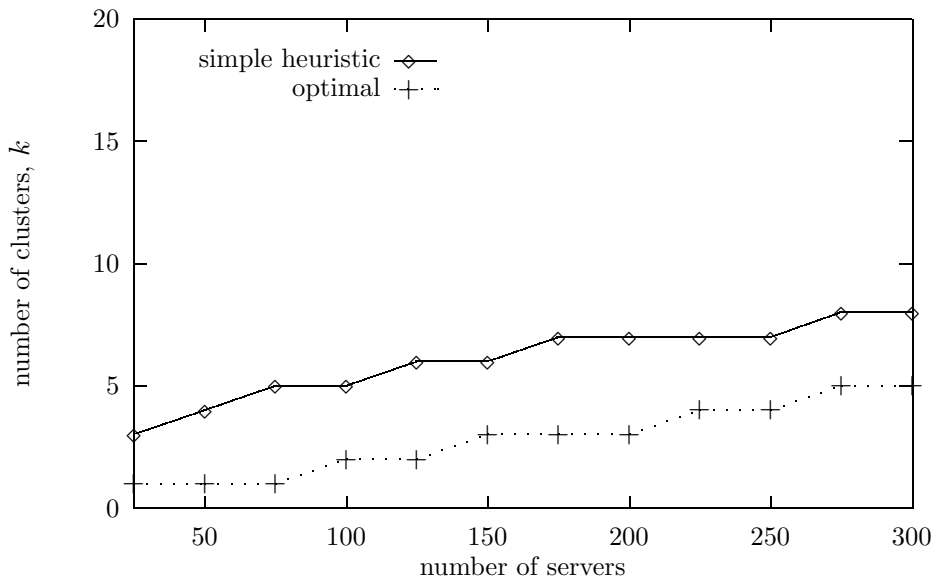


Figure 4: increasing N , load = 80%

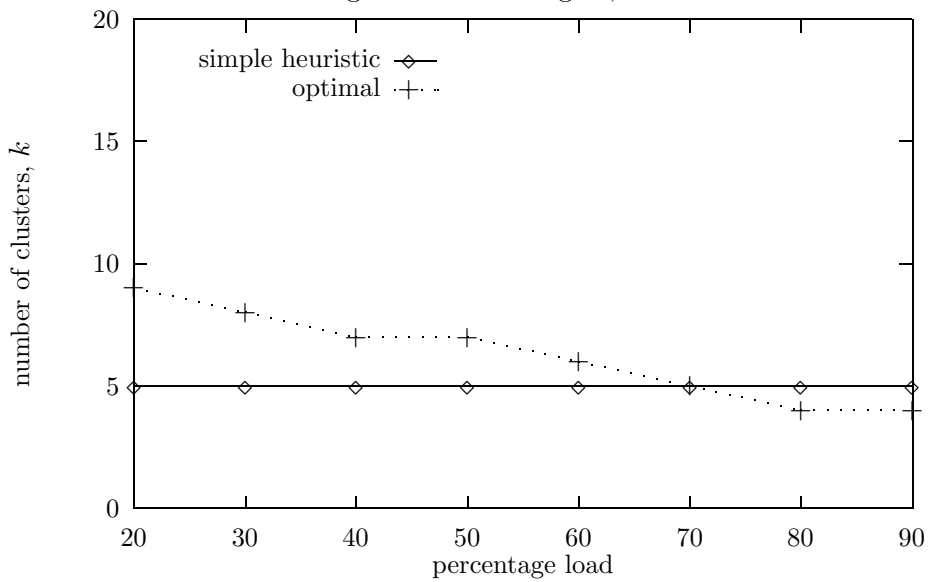


Figure 5: increasing load, $N = 100$

Acknowledgement

This work was carried out as part of the collaborative project GridSHED (Grid Scheduling and Hosting Environment Development), funded by British Telecom and the North-East Regional e-Science centre.

References

- [1] L. Kleinrock, *Queueing Systems Volume 1: Theory*, Wiley, 1975.
- [2] G. Cybenko, “Dynamic Load Balancing for Distributed Memory Multiprocessors”, *Journal of Parallel and Distributed Computing* 7: pp 279-301, 1989.
- [3] M.J. Zak, W. Li, S. Parthasarathy, “Customized Dynamic Load Balancing for a Network of Workstations”, *Procs. of the 5th IEEE Int. Symp.*, pp 282-291, HDPC, 1996.
- [4] J. Hine, T. Holzer, “Task Allocation in a Distributed System”, *Procs. of UniForum’97: Untangling the Web*, 1997.
- [5] M. Houle, A. Symvonis, D. Wood, “Dimension-Exchange Algorithms for Load Balancing on Trees”, *Procs. of 9th Int. Colloquium on Structural Information and Communication Complexity*, pp 181-196, 2002.
- [6] D. Thain, T. Tannenbaum, and M. Livny, “Condor and the Grid”, in *Grid Computing: Making The Global Infrastructure a Reality* (eds F. Berman, A.J.G. Hey, G. Fox), Wiley, 2003.