The Advertisement System – a Distributed Approach to Resource Reservation and Scheduling in Grid Networking

Toby Moncaster and Dr David Hunter
Department of Electronic Systems Engineering, University of Essex

Abstract: This paper presents a novel distributed approach to resource reservation and allocation in Grid Networking. It looks at the drawbacks of current centralised schemes and shows why a distributed scheme is preferred. It presents some results of a simulation of such a system using OPNET and discusses further work that will be undertaken on the basis of these results.

1 Introduction

Grid networking [1,2,3] is a major expansion of the current ideas of computer internetworking. It envisages a user being able to call on computing resources from across the global network to complete a required operation. In order to do this it is necessary to be able to reserve these resources and control their scheduling. The great majority of current work in this area of Grid is looking at ways of adapting systems used for resource reservation in parallel computing. A good example of such a system is the Globus Architecture for Reservation and Allocation (GARA)[4]. There are a number of well-known problems with such centralised systems. This paper presents an alternative approach using a distributed resource reservation scheme.

2 Existing Resource Management Schemes

Several resource management schemes already exist for Grid networks. Generally these schemes are based on extensions to resource managers for parallel distributed computer systems. A good example of such a scheme is that used by the Globus Alliance[5]. They have been developing Grid applications for several years. They have produced a number of systems that are widely used in existing Grid projects. The Globus Architecture for Reservation and Allocation[4] is one of these systems. It is based on another system known as the Globus Resource Allocation Manager (GRAM)[6]. This is designed for resource management in real-time meta-computing environments.

2.1 Globus Architecture for Reservation and Allocation (GARA)

The GRAM architecture[6] is only capable of managing resource in a real-time meta-computing environment. The majority of Grid services are likely to require some form of advance resource reservation. With this in mind, the Globus Alliance[5] proposed the Globus Architecture for Reservation and Allocation[4]. GARA is based heavily on GRAM. It utilises 3 key components – the resource broker, the co-allocation agent and the co-reservation agent.

Co-Allocation Agent – this takes resource requests destined for multiple sites and breaks them into their constituent requests. It then passes these requests to the appropriate resource managers. It also monitors the progress of all the requests.

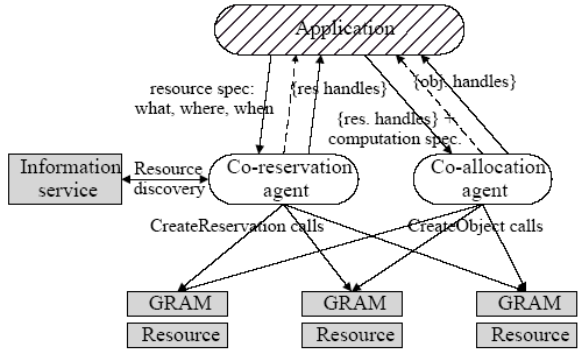


Figure 1 GARA architecture [4]

Co-Reservation Agent creates advance reservations for resources. It does this by exhaustively searching the set of available future resources. It then passes the advance reservations to the appropriate local resource manager, and creates the appropriate reservation handles and passes these back to the Application. The Application then sends these to the Co-Allocation Agent.

Resource Broker – this takes high-level, complex RSL [7,8] requests from the application and converts them (through a process known as Specialisation) into simpler resource requests. The broker then consults the information service to see what resources are available and sends appropriate resource requests to the co-allocation manager.

2.2 Resource Specification Language

The resource specification language[7] and the more recent Extended RSL[8] utilise a syntax based on that used for filter specifications in the LDAP (lightweight Directory Access Protocol)[9] and hence, the MDS (Metacomputing Directory Service)[10] protocols. It is a highly flexible language that can be used to describe resources accurately and simply. Complex specifications can be assembled from simpler constituent parts by

using the Conjunction (&), Disjunction (|) or combine (+) operators. It can also take other operators (<, <=, =, >=, >, !=) and a series of parameters (such as time required/available, executable to be run, any arguments, any environment variables, etc). For example, the following specification:

&(executable=prog) (|(&(count=5)(mem>=64))(&(count=10)(mem>=32)))

requests 5 nodes with at least 64 MB memory or 10 nodes with at least 32 MB. In this request, executable and count are scheduler attribute names, while memory is an MDS attribute name.

3 Drawbacks of Centralised Schemes Compared to Distributed Schemes

The GARA[4] system works by utilising a number of centralised elements (the broker, the co-allocator, the co-reservation agent and the information service). Such centralised systems are fine where the network is relatively small, stable and reliable. However they have several drawbacks:

Centralised systems can be very sensitive to link or node failure: If one of the nodes or links near the centre of the system fails there may be a disproportionate affect on the overall performance of the system. By contrast distributed schemes can cope with failure promptly and with limited effect on the overall system.
Centralised systems are very susceptible to malicious attack: If there is a single central control then it becomes an obvious target for attack by malicious programs. By contrast in a distributed system there is no single target for such attacks.
Centralised systems are slow to respond to changing state within the network: Because the control is centralised it needs to become aware of any change of state within the network before it can react to it. Distributed systems can react much faster to changes of state.
Centralised systems create congestion in the centre of the network: Most control traffic in a centralised network is concentrated in the vicinity of the central control. In distributed networks the traffic is more evenly spread across the network.
In centralised systems nodes will suffer from variable response times from the server: Nodes near the centre of the system will have a very rapid response time whereas distant nodes will take a long time to get a response from the central server. In a distributed system this is no longer a problem.

4 Requirements for a Distributed System

In order to design a distributed resource reservation and scheduling scheme it is first necessary to define what functionality the system must have and the operating constraints the system must meet. These include:

- The system must be able to create and destroy resource reservations
- It must be able to police the reservations
- It should be able to cope smoothly with node or link failure

In addition, since this is a distributed system the reservation tables will be held locally at each node, rather than in a central information service. Each node will have a different set of resources and some resources may become unavailable over time (for instance storage is a finite resource).

4.1 Possible Systems

Two possible approaches exist which satisfy the above conditions. These systems approach the problem from opposite sides. The first system is called the advertisement system. Nodes within the network have no knowledge of resource availability within the rest of the network. Requests are broadcast across the network in the hope of finding suitable resources. By contrast, in the Explicit Request System (which isn't explored in this paper) each node has a global knowledge of resource availability and will use this to send a request to a specific node which has the resources available.

4.2 Advertisement System

As already stated, in the advertisement system nodes have no knowledge of global resource availability. When a node needs a resource that isn't available locally it will broadcast an advert packet setting out what resource it wants and a time by which it needs that resource. It will then receive response from the rest of the network and will choose the most appropriate. The detailed behaviour is as follows:

- The origin node will broadcast a request across the network. This request will include details of what resources are required, where the request originated, a unique identifier and a time-to-complete-by field.
- When a downstream node receives the request it will check whether the time-to-complete has passed. If it has it will drop the request. It will also check the request index and if it has already parsed this request it will drop it.

- If there is still time it will parse the request and check whether it is able to meet it.
- If it is able to meet the request it will send back an acknowledgement to the origin and make a “soft” reservation for that resource.
- If the node can’t meet the request it will broadcast it on all downstream links.

If multiple nodes respond to the request then the origin will have to decide between them according to some heuristic. This could be as simple as accepting the first response it receives or could be more complex and take into account distance to node, bandwidth availability, etc. It will then contact the node that it chooses and confirm the reservation.

5 Modelling the Advertisement System

In order to test the behaviour of the advertisement system it has been modelled using the OPNET Modeller network simulation tool. In the initial model there is only a single resource available within the network. Each node within the network has a certain capacity to process this resource. Nodes generate requests for this resource according to a Poisson time distribution. The size of the request is negative exponentially distributed.

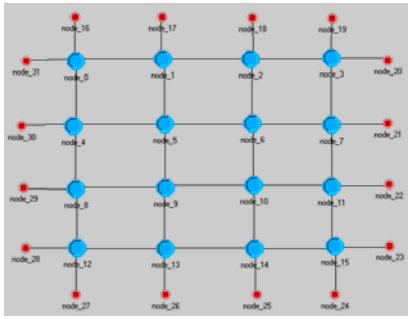


Figure 3 the test network topology

The current model is a full duplex mesh network. Each node can connect to up to 4 others and unconnected edge links are terminated with simple sink nodes (see figure 3).

Requests are processed from a FIFO (first in first out) queue at each node. As a request travels through the network it records the route it took. Replies to the request return down the same route. Requests are allowed to travel a maximum of 6 hops through the network to try and reduce the amount of control traffic. If after 6 hops the request hasn’t been met it will be dropped. Results from simulating this network are shown in section 6.1.

5.1 Simulation Results

The following graphs show come from simulating a 4 x 4 mesh network. Each node has a capacity of 200 units/s. These results are obtained by using an inter-request time of Poisson(1s). The load is varied by altering the mean size of the request

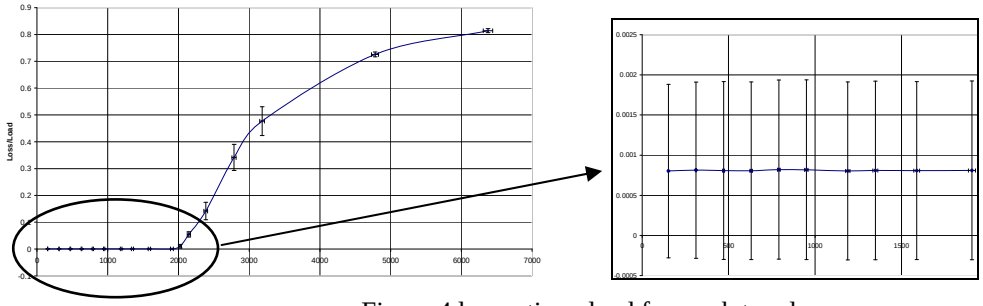
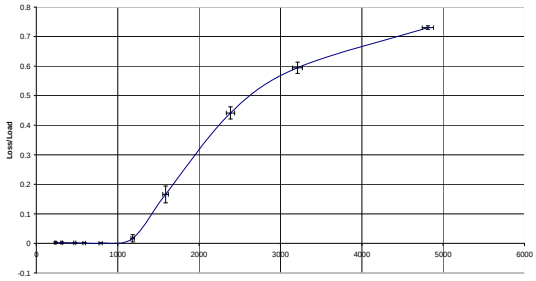


Figure 4 loss ratio v. load for mesh topology

Initially the loss ratio is almost 0 until the load reaches 2000 units/s (approx 60% capacity) and then increases rapidly until the load reaches 3200 when the loss ratio is approx 0.5 as would be expected. Most of this initial loss is caused by the relatively small number of requests that don’t get processed within 5 hops and consequently get dropped. It is essential to ensure failed requests are cleared from the system rapidly. This ensures that the control channel is never congested.



In the graph on the left the mean request size has been kept constant with the load being altered by varying the inter-request time. As can be seen the loss is similar to that shown in figure 3 but the loss starts to rise at a lower load.

This is probably a function of the relatively large fixed request size which means at low loads the loss of a single request can have a disproportionate effect on the loss ratio.

Figure 5 loss ratio v. load constant request size

5.2 Measuring Delay

Each link in the network has a fixed delay of 1ms (approximately equal to 200km of fibre). A number of delay statistics can be collected in the model. In the following graphs delay has been calculated by measuring the time between a request being transmitted and the time at which that request is assigned a slot to get processed. This delay is then averaged across the entire network. Requests that don't receive a processing slot are ignored.

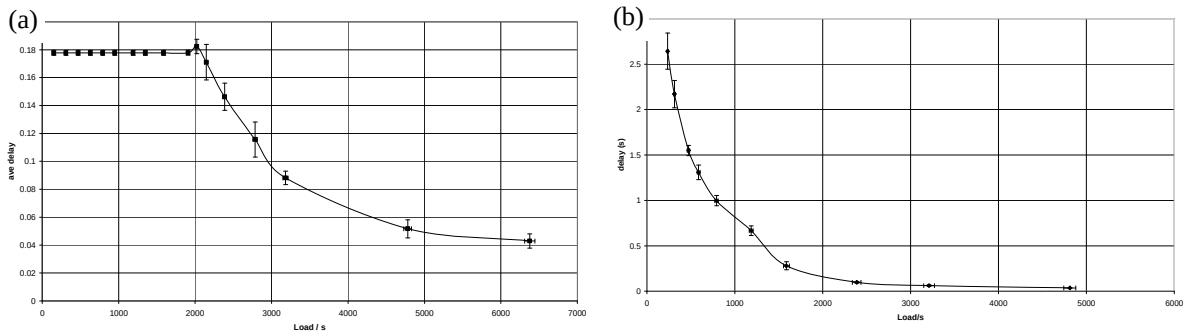


Figure 6 queuing delay for (a) constant inter-request time (b) constant request size

As can be seen, in 5(a) the delay is constant whilst the load is less than 2000. Then as the load increases beyond this and more packets are lost the delay drops rapidly. This ties in with the loss graph in figure 5. In 5(b) the delay is initially very high. However this appears to be a function of the way OPNET handles interrupts. What is significant is that at loads above 2000 the delay is very similar in both cases.

6 Future Work

A number of extensions to the current model are being looked at:

Complex Topologies: The effects of failing a single node within the mesh network model have been explored and as expected in a mesh network seem to have little effect on overall system. It is intended to model a more realistic network based on one of the trans-European research networks (such as GEANT [11]).

Complex Queuing Schemes: Alternative queuing schemes will also be examined to see if they have any impact on the network behaviour.

Multiple resource types: Real systems will have multiple resource types. Requests will be complex with more than one resource needed co-dependently. The model will be expanded to look at such systems.

7 Conclusions

The results shown here demonstrate that the simple distributed approach to resource reservation appears to work for systems that are moderately loaded. The delay between a request being made and it being assigned a slot to be processed is relatively short and in systems where the individual requests are not large the delay is fairly constant until the load exceeds about 60%. Clearly the current model needs to be expanded to more accurately reflect real-life Grid networking applications. It would also be useful to compare this approach with an alternative system such as an explicit request system where nodes have a knowledge of global resource availability that is updated periodically.

References

[1] The Grid.org website http://www.grid.org/home.htm
[2] "The GRID 2 Blueprint for a New Computing Infrastructure" edited by Ian Foster & Carl Kesselman © 2004 Morgan Kaufmann
[3] "Understanding Grids" from the Global Grid Forum website: http://www.ggf.org/ggf_grid_understand.htm
[4] "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation" I Foster, C Kesselman, C Lee, R Lindell, K Nahrstedt & A Roy © 1999
[5] The Globus Alliance website: http://www.globus.org/
[6] "A Resource Management Architecture for Metacomputing Systems" K Czajkowski, I Foster, N. Karonis, C.Kesselman, S. Martin, W.Smith, and S.Tuecke © 1998
[7] Globus Resource Specification Language: http://www-unix.globus.org/developer/rsl-schema.html
[8] The Extended Resource Specification Language © NORDUGRID 2003 http://www.nordugrid.org/documents/xrsl.pdf
[9] LDAP Standards and Documents: http://www.mozilla.org/directory/standards.html
[10] "A Performance Study of Monitoring and Information Services for Distributed Systems". X. Zhang, J. Freschl, and J. Schopf ©2003
[11] The GÉANT Network: http://www.geant.net/upload/pdf/Topology_Oct_2004.pdf