The Application of Bioinspired Engineering Principles to Grid Resource Allocation

Simon Davy, Karim Djemame, and Jason Noble

School of Computing, University of Leeds
{wavy,karim,jasonn}@comp.leeds.ac.uk

Abstract. Resource allocation is an essential problem to be solved in building effective Grid systems. Existing mechanisms employ a centralised approach which is likely to run into performance problems as the size of the Grid increases. We suggest taking inspiration from the biosystems paradigm in order to construct a resource allocation protocol that is adaptive, efficient, scalable, and robust. We have constructed a simplified Grid model in which resource nodes make competitive offers in order to be accepted by jobs. Our simulation results show that our bioinspired resource allocation algorithm is a viable contender for use in future Grid implementations.

1 Introduction

Grid systems are a highly active area of research, aiming to provide computing resources as transparently as the national power grid provides electrical power. One of the essential problems to overcome in reaching this aim that of resource allocation. There is a need to facilitate automatic discovery and allocation of a dynamic set of resources. The major difference between resource allocation on the Grid compared to other environments is the level of heterogeneity involved. The possible variations in type, configuration, deployment and performance of Grid resources is vast. In particular, the autonomy of each diversely owned and administered resources poses difficult problems to a global solution.

Existing mechanisms for resource allocation are based on allocation paradigms from previous work in other environments, where ownership and access has been relatively uniform[5]. These initial ideas have been greatly extended for the Grid environment. However they are based on a centralised approach which, we will argue, is likely to run into performance problems as the size of Grid systems increases.

We suggest that various features of the problem of Grid resource allocation call for a fully distributed approach. Existing research in the *biosystems* paradigm may lead us to relevant solutions. Biosystems research involves constructing complex adaptive computational systems that take inspiration from similar systems in nature such as insect colonies or even, at a higher level, market economies. The biosystems area is relatively new, and attempts to apply these ideas to resource allocation on the Grid are in their infancy. The current paper sets out an application of the biosystems paradigm to Grid resource allocation. We are continuing to develop a simulation framework with which to explore various bioinspired allocation mechanisms and to compare their performance with existing approaches.

We begin in section 2 with a summary of existing solutions to the problem of Grid resource allocation, and a discussion of their limitations. In section 3 we present an overview of biosystems research and present the case for its application to Grid resource allocation. In section 4 we describe a simple Grid simulation model and outline our bioinspired allocation algorithm. In section 5 a detailed overview of the simulation is presented, before moving on to the results in section 6. We conclude and outline further work in section 7.

2 Resource Allocation

The goal of Grid resource allocation is to provide a robust and scalable mechanism for automatic and transparent allocation of appropriate resources. The much of the work on automatic resource allocation[9] has been carried out within the Globus Grid project[12]. Firstly, they define a "Virtual Organisation" (VO) as a group of resource owners that have a common agreement about sharing resources[6]. They facilitate resource allocation within a VO by maintaining a centralised aggregate directory service that contains information about the resources available to that particular VO[3]. This approach has two components : resource discovery and resource enquiry.

Resource discovery is achieved via a soft-state registration protocol that publishes a resource's information to a centralised server. Some interesting work has been done on making this process distributed [8], however, it still feeds into a centralised server. Resource enquiry is handled by querying the centralised server for suitable resources and then negotiating with those resources for allocation. This approach allows a potentially globally optimal allocation of resources to a specific request, and is helpful in giving an overall view of a VO's state and activity. However, there are a number of possible disadvantages to this approach.

- *latency* - the central directory will always be somewhat behind the actual state of the resources, resulting in possible false matches when enquiring for suitable resources. The impact of this will vary depending on the dynamism of the VO resources, with highly dynamic resources implying a greater negative impact.
- *centralisation* - the need to globally capture the VO's resource implies a scalability limitation for large numbers of resources. The current solution to this is to aggregate the server to multiple hierarchical servers to achieve a greater distributed performance, but again, this is ultimately limited in scale.
- *maintenance* - these servers must be allocated and maintained which is a significant overhead to administration, as well as Grid activity.

– *single point of failure* - if the central server fails, the whole allocation mechanism also fails. Even if aggregated, it would decrease scalability and fault tolerance.

3 Biosystems Engineering for the Grid

There are many naturally occurring systems that achieve a high level of complex adaptive behaviour. One example that has been applied to network routing with good results is based on colonies of social insects such as ants[4]. In the natural world, ants deposit a pheromone when they carry food, thus developing a pheromone trail from food sources to the nest. Other ants can follow this trail and in turn reinforce it, so more ants follow it. The net effect is that the food source is exploited quickly and efficiently. This metaphor has been applied to routing tables in switched networks, where ant constructs traverse the network building up 'pheromone' trails indicating good routes. Other examples of biosystems work include evolutionary algorithms, neural network models and economic market models[13][7].

Bioinspired systems consist of autonomous agents based in an environment:

– *autonomous agency* - the system is populated with agents acting concurrently with all other agents: it is inherently parallel. Each agent in a biosystem is autonomous, making its own decisions based on its internally defined behaviour; there is no exterior overriding control.
– *environmentally situated* - the agents in a such a system are not in a vacuum, they are situated in an environment of some description. They can change and be changed by this environment. An agent is only able to interact with its local environment and other agents in that local environment. It has no global communication or access to global state. It may be mobile and thus able to change its locality.

A Grid system can also be viewed as consisting of agents in an environment. Depending on one's point of view, the agents on the Grid could be computational resources, users or even jobs. The environment for such agents may be a VO, the Grid infrastructure, or a distribution of jobs in time. Therefore it is reasonable to apply bioinspired algorithms to a problem such as match requests with resources.

Bioinspired systems have several interesting and desirable properties:

– *adaptive* - they can adapt quickly and efficiently to change in the environment.
– *efficient* - they can develop efficient solutions to difficult problems. (This efficiency is in terms of an individual agent, and there may be a tradeoff with global system efficiency.)
– *scalable* - they are highly scalable and performance is often enhanced with a greater number of agents.
– *robust* - faults and failures in a minority of individual agents will not affect the performance of the overall system.

These are the same desirable characteristics we would wish to have in a Grid resource allocation system. They are essential for the goal of automatic transparent allocation in a Grid of arbitrary size.

There will rarely be a direct mapping from natural biosystems to artificial computing systems. However, what we are proposing is to use the central ideas and principles evident in these natural systems to construct artificial systems with similar characteristics. Some of the key principles that are responsible for robustness, efficiency and scalability are listed below.

- *indirect communication* - agents often communicate indirectly through the environment. e.g., the use of pheromones by foraging ants to leave trails for others.
- *compound communication* - effective use of multiple agents to verify a particular solution's quality. Positive feedback on good solutions gathers others to those locations.
- *decay* - biosystems are never inactive, there is always some background activity, often in the form of decay of some attribute or other. This works directly with the positive feedback mentioned above, in that little-used solutions decay and die away. e.g., ant pheromone trails evaporate over time, meaning they will disappear unless reinforced.
- *adaptive simplicity* - agents often use simple behavioural functions in which the relevant parameters are adapted through decay and positive feedback mechanisms.

We propose that these bioinspired design ideas, possibly alongside more traditional approaches, may be used to develop a bioinspired system for Grid resource allocation that is scalable and robust.

4 Investigation Framework

To investigate the applicability of these ideas to Grid resource allocation, we are developing a simulation tool to analyse the performance of different algorithms. Simulation is used because current Grid technologies are not developed enough to implement our ideas on a real system. In addition the size of current Grids is still relatively small: given that one of the factors we are interested in is scalability, we need to use many more resources than are currently available. Finally, the biosystems approach necessitates a modified Grid architecture that current Grid middleware solutions do not provide. These factors lead us to use simulation as our investigative tool.

Based on our case for the potential benefits of a biosystems approach to Grid resource allocation, we are currently developing a simple bioinspired algorithm to explore the basic performance and characteristics of this approach. An aim for the immediate future is to gauge the effectiveness of our approach in comparison to existing approaches; however, the current paper is about demonstrating that our proposed algorithm can effectively allocate jobs on a simulated Grid.

4.1 Simple Grid Model

The initial Grid will consist of simple computational resources only. The ideas of storage and network resource types will be added in at a later date. For now, we will ignore the differences in speed or power and concern ourselves simply with size, due to the complexities involved with predicting performance of a particular job on a particular resource — we model the idea that some nodes have greater capacity than others, but beyond that we assume all nodes are equal. This should serve for an initial investigation.

A job will initially simply consist of a size value and a length (time). Large jobs will only be able to run on large nodes. For now we ignore the possibility of multiple resource node co-allocations. Another factor we are not considering initially is advance reservation; we assume a simple immediate accept or reject allocation. Job size and rate will be modelled using established distributions from the simulation literature[10], i.e., Gamma distributions for job size and length, and a negative exponential distribution of job interarrival times.

4.2 Simple Bioinspired Allocation Algorithm

We propose a relatively simple decentralised algorithm for allocating jobs to resources based only on job size and available resources. We also use an overlay network in order to introduce a notion of locality.

Resource nodes are interconnected via an 'internet' so any node can interact directly with any other, but on top of that there is an overlay network between nodes. This overlay network is used to forward allocation messages between resource nodes, and represents a single VO in organisational terms. Initially this network is based on a small world topology[14]; extensions could investigate other topologies, including random or hierarchical networks[11].

The basic mechanism for allocating a job is as follows. An initiating node sends a description of the job out on the overlay network. This is forwarded through the overlay network to distribute the request. Upon receiving a job description, a resource node may make an offer on that job directly back to the initiating node. A node waits to receive a fixed number of offers, and chooses the best offer [1]. This bidding metaphor is used for two reasons. Firstly it is the suggested economic model for Grid systems[2], and secondly it is a relatively well understood mechanism within economics and biosystems[7].

The mechanism a node uses for deciding whether or not to make an offer, or indeed how much to offer, is based around a simple heuristic. This assumes a natural bias toward it being more profitable to accept a small number of large jobs rather than large numbers of small jobs. This is the intuitive direction given the overhead of job startup/teardown costs. This is captured by a sigmoid function of available resources against job size (see figure 4.2). The shape of the sigmoid reflects the acceptance strategy. This function is inspired by similar functions in ants[4], and is chosen for simplicity.

[1] A possible extension is to combine this with a time out mechanism to prevent resource allocation failure

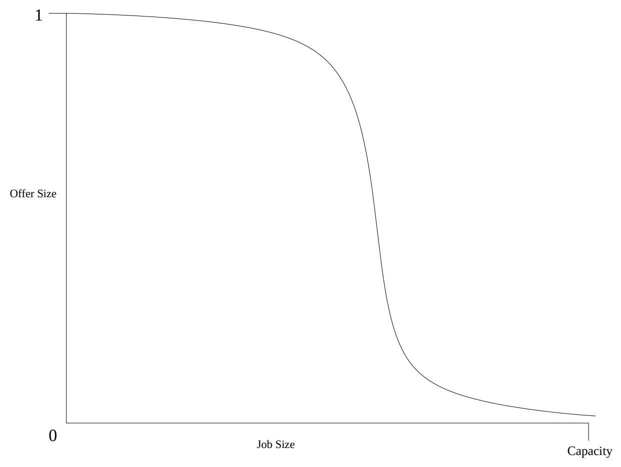

Fig. 1. Simple sigmoid function making offers based on size. Note that in the bidding metaphor, resources are sellers and jobs are buyers — therefore the more a resource 'wants' a job, the lower the bid it will make.

This sigmoid gives a measure of how profitable the job is for the node. The sigmoid score represents a competitive price for the job. In this situation, the resources are producers and the jobs are consumers. If it is a profitable job for the resource (i.e. it will fill the resource to capacity or close) it offers a low price. The less profitable the job is, the less competitive a price (i.e., a higher price) the node offers to do the job.

There is a need for bidding to be adaptive over time, or else some resources may be starved of jobs. This idea is consistent with adaptive thresholds in some species of insects[4] as well as market economy price fluctuations[7]. For the sake of simplicity, we have not included this aspect in this initial work. However, further work will include this.

4.3 Performance Metrics

In order to assess the performance of our algorithm, we collect data on allocation latency, the number of successfully allocated jobs, the number of failed allocations, and the proportion of failed allocations among the set of all jobs. These metrics will be gathered across different contexts of Grid size and job load.

5 Experimental Design

5.1 Simulation Tool

We designed and implemented a sequential discrete event simulator in C++. The simulator provides control over Grid size, load, job duration and resource node capacity. It uses these values to determine an appropriate time frame and arrival rate for new jobs into the system.

5.2 Simulation Scenario

The simulation begins by setting up a randomly generated architecture for the overlay network. As noted above, the network has a small world topology. This network is generated by creating a mother node, and then iteratively creating additional nodes each with two two-way connections. These connections run from the new node to an existing node which is randomly chosen with a bias toward nodes that already have more connections. This “rich get richer” effect is characteristic of small-world networks. The capacity of a new node is a random variate drawn from a Gamma distribution; see table 1 for the relevant parameter values.

The simulation proceeds to generate jobs with a negative exponential distribution of inter-arrival times. The mean job arrival rate is chosen such that it gives a specified load level for the network, where a load of 1.0 represents a situation in which, hypothetically speaking, 100% of the network's resources would be in use assuming an ideal distribution of jobs. Job arrival rate therefore depends on Grid size (number of nodes), mean job duration, mean job size, mean resource capacity, and the desired load level for the simulation.

Description	Values	Distribution
Grid size (nodes)	100, 256	
Load	0.8 .. 1.1	
Mean resource capacity	100	Gamma ($\alpha = 0.5$)
Mean job duration	500	Gamma ($\alpha = 2.0$)
Mean job size	20	Gamma ($\alpha = 2.0$)
Mean overlay communication time	0.1	Normal ($\sigma = 0.1$)
Mean 'internet' communication time	0.3	Normal ($\sigma = 0.3$)
Mean service time	0.1	Normal ($\sigma = 0.1$)
Overlay hop limit	3	
Minimum number of offers	5	

Table 1. Parameter values for the simulation. Note that variates from each of the three normally distributed latency times are resampled if negative.

A new job is assumed to arrive on the network at a random node; this node is designated as the job handler. The job is associated with a randomly determined size and duration; both of these numbers are random variates drawn from a Gamma distribution; table 1 gives the relevant means. The handler node immediately forwards the job's service request to all of its connections on the overlay network. Transmitting these messages involves a modest, normally distributed delay time. Those connections in turn forward the service request on to all of their overlay network connections; however, care is taken not to send the request on to a node that has previously seen it. This process comes to an end when the overlay hop limit is reached. The overlay hop limit exists in order to avoid never-ending message broadcasts; the idea is to start looking for an appropriate resource in the local area, to look further afield if necessary, but never to scan the whole network. (Without a hop limit, our approach would not have any potential advantages over the centralised approach.)

When a resource node receives a service request, it evaluates that request according to its sigmoid function and sends back an offer (directly, via the 'internet') to the job handler. The evaluation process involves a normally distributed delay, as does the direct communication over the 'internet' (see table 1). The job handler waits until it has received a minimum number of offers (in this case five; see table 1), and then attempts to allocate the job with the node that has made the lowest or cheapest offer. For the purposes of this simple initial simulation, failure to receive the full complement of offers (typically due to system load) results in the complete and permanent failure to allocate the job. In a richer model, it would of course be possible to implement a timeout system whereby the best offer is eventually accepted even if too few offers have arrived. In the event that the node making the lowest offer is no longer available at the point in time when allocation is attempted, the job handler chooses the next lowest offer. Note that this is not an uncommon occurrence as system load increases, because in the current system, the bidding node does not reserve space for the job in any way.

Once jobs are allocated, they are assumed to automatically complete after their scheduled duration has been reached and they have no further effect on the system. The simulation is run for a period of 2000 units of simulated time; this particular value was chosen such that each resource is expected to have been allocated to capacity three over. When this time limit is reached, no new jobs are added (i.e. load drops off to 0), however allocations in progress are allowed to succeed or fail.

6 Results and Evaluation

We simulated Grid systems of both 100 and 256 nodes. The results for both sizes were qualitatively similar, and so we will report only the results for the larger, 256-node system.

Figure 6 shows that larger numbers of jobs are successfully allocated under higher load levels. This is not surprising given that there are more jobs in the system. The results also show that jobs are allocated successfully with a relatively small failure rate. As load increases we see increasing failure rates, however the failure rate remains linearly proportional to load until the highest load levels are reached.

Given that the mean communication latency for overlay network hops is 0.1 and for 'internet' communication is 0.3, the allocation times shown in figure 6 indicate that a significant amount of back and forth communication is occurring

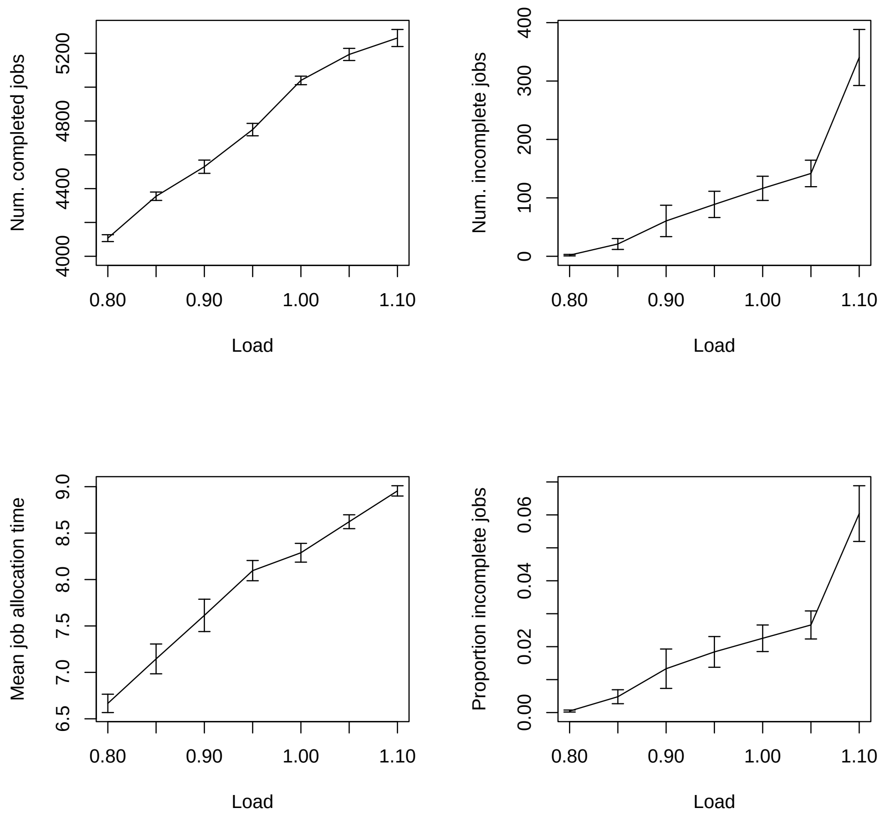

Fig. 2. Performance of the simulation. Clockwise from top left, the figure shows the number of completed jobs, the number of incomplete jobs, the proportion of incomplete jobs, and the mean job allocation time. All of these measures are shown across 7 load levels; each data point shows the mean and standard error across 10 simulation runs.

before jobs are assigned to nodes. In other words, the negotiation process between jobs and resources takes time. On the positive side, the mean allocation time scales linearly with load. As the load exceeds 1.0 and the system becomes more 'crowded', unsurprisingly, some jobs fail to be allocated. However, there is no corresponding explosion in allocation time for those jobs that have been successfully allocated.

A closer inspection of the results shows that jobs that fail to be allocated tend to be large jobs. The overall mean size of unallocated jobs is 56.3, compared to a mean size of 20 for all jobs. We have also found that many observed allocation failures were due to the lack of a time out mechanism for responding to offers. Many of the failed jobs never received a full set of five offers, and therefore never responded to an offer and were never allocated. A timeout mechanism would have allowed these jobs to eventually respond to their best existing offer.

The mechanism that allowed the simulation to continue past 2000 time units led to an overall average of 18 time units being added on. This represents about three consecutive job allocation times but does not seem to have any significant effect on the results.

7 Conclusion and Further work

We have shown that our bioinspired resource allocation algorithm is a viable contender for use in future Grid implementations. In particular the linear scaling of job allocation times suggests that graceful degradation under high load conditions may be exhibited by real systems of this type.

Clearly, the most important step for the further exploration of these results is the systematic comparison of this algorithm with the existing centralised approaches such as that used in Condor/G [1].

There are a number of ways in which this algorithm could be refined. Currently the sigmoid function for evaluating a job is static. Greater flexibility could be achieved by allowing each node to adapt its evaluation function over the course of the simulation run. An under-utilised resource could lower its offers according to how long it has been idling. Similarly a busy resource could raise its offers in order to achieve better balance over the network.

Further expansions that would allow as to more accurately model a real Grid system include:

- Performance — estimation of a job's performance factored into decision to bid and bid amount.
- Network topology — examine different networks for their effects on performance and robustness[11].
- Multiple resource types — model various types such as storage, network, I/O devices, etc.
- Multiple resource co-allocation — spreading large jobs across multiple resource nodes.
- Advance resource reservation — reserving capacity for jobs that are expected to arrive in the future.

– Multiple VO networks — the current system assumes a common usage policy across all nodes, but this is not likely to be the case on future large scale Grid systems.

References

1. Jim Basney and Miron Livny. Deploying a high throughput computing cluster. *High Performance Cluster Computing*, 1:Chapter 5, May 1999.
2. Rajkumar Buyya, Steve J. Chapin, and David C. DiNucci. Architectural models for resource management in the grid. In *In Proc. of 1st IEEE/ACM International Workshop on Grid Computing (GRID 2000), Springer Verlag LNCS Series*, December 2000, Bangalore, India.
3. K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing, 2001.
4. Guy Theraulaz Eric Bonabeau, Marco Dorigo. *Swarm Intelligence : From Natural to Artificial Systems.* Oxford University Press, 1999.
5. I. Foster and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, 1999.
6. Ian Foster. The anatomy of the grid: Enabling scalable virtual organizations. *Int. Journal of Supercomputing Applications*, Vol 15, No. 3, 2001.
7. B. A. Huberman, editor. *The Ecology of Computation.* NorthHolland Publishing Company, 1988.
8. Adriana Iamnitchi and Ian Foster. On fully decentralized resource discovery in grid environments. In *International Workshop on Grid Computing*, Denver, Colorado, November 2001. IEEE.
9. R. Buyya K. Krauton and M. Maheswaran. A taxonomy and survey of grid resource managment systems for distributed computing. Int. Journal of Software: Practice and Experience, Feb 2002.
10. A. Law and W. Kelton. *Simulation Modeling and Analysis.* McGraw-Hill, 3rd Edition, 2000.
11. M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.
12. Globus Project. The globus project homepage. http://www.globus.org/ [25/4/2002], 2002.
13. Mitchel Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds.* MIT Press, 1994.
14. D. J. Watts and S. H. Strogatz. Collective dynamics of ‘small-world’ networks. *Nature*, 393:440–442, 1998.