Overview of Grid Scheduling Systems

Peter Gradwell

Department of Computer Science, University of Bath

Abstract

Current efforts in Grid Computing operate on a zero settlement basis with users on the Internet downloading computation engines for "good causes" and subsequently returning their results.

A number of attempts, for example Condor-G, GRaDS and Nimrod-G, have been made to develop computational economy based scheduling systems.

However, these systems operate on a "top down" basis and have a number of limitations, in that they assume knowledge of all the jobs on the grid, they require the ability to manage the scheduling of jobs on grid nodes and they assume a direct linkage between the "grid to node" and "within node" scheduling processes.

Our work in the area is focused on examining how we may use multi-agent systems to develop a market place for the trading of computational grid resources such that grid scheduling is completed without the limitations of the "top down" approach.

1 Introduction

Current grid computing systems operate on a zero settlement basis. Users give their resources for free and researchers connect across their networks to utilise them. As such, availability of computational resources is unpredictable and jobs are allocated on a first come first served basis. Within a closed group, for example a research facility's own "mini-grid" of computing resource, scheduling policies may differ but they are typically still managed by an administrator who will execute jobs on a first come first served basis, or who will prioritise jobs according to political pressure.

The various methods current employed, i.e. first come first served, time slot, priority or availability based scheduling of jobs on a grid cluster all operate on a "top down" basis and have a number of limitations in that they require knowledge and direct control of all the jobs on the grid.

They are therefore, inefficient scheduling methods for this type of computing infrastructure, as discussed in [22].

Agent based trading systems would appear to provide an excellent solution to the problems inherent in a "top down" scenario, allowing cost effective scheduling based on market economics (i.e. where the user specifies how important the job is and enforces that importance with renumeration). In this paper we review the current status of different methods of job scheduling in a grid based on a "computing economy".

2 Entry of Jobs into the Grid

All grid scheduling systems work on the basis that the "new task" which wants to be executed has to make itself known to a "resource selector". In current systems, the resource selector acts as a gateway to the grid. It will select resources from a global directory (e.g. the Globus MetaDirectory Service[15]) and then allocate the job to one of the available grid nodes. Typically, job allocation is done in two stages. Firstly a job is allocated to a particular node on the grid and then within that node, the job will be scheduled onto the processor.

In this paper, we consider the first practise to be "resource allocation" whilst the second is "job scheduling". This approach has been investigated in some literature, under the term "meta scheduling". For example, the GrADS project [3] is an attempt to devise a meta-scheduling system by members of the University of Tennessee, Computer Science Department.

"The metascheduler receives candidate schedules of different application level schedulers and implements scheduling policies for balancing the interests of different applications." [20]

The Grid Scheduling Dictionary Project [16] has defined the concepts of a "meta scheduler" and "super scheduler":

"A scheduler that allows to request resources of more than one machine for a single job. May perform load balancing of workloads across multiple systems. Each system would then have its own local scheduler to determine how its job queue is processed. Requires advance reservation capability of local schedulers." [16]

3 Current Schedulers from Grid System

A number of schedulers for grid computing systems have been developed. In his paper "Survey of a Grid Meta-Scheduler" [13], Sebastian Ho discusses eight current projects. However only a few (Nimrod-G, GrADS and Condor-G) would seem to (a) be at a relatively advanced stage of development and (b) have wide spread operational deployment by current Grid operators.

It is interesting to note that, at the time of writing the Globus [15] project is the current favourite grid computing toolkit, having been adopted by IBM, HP, etc. and thus many of the smaller schedulers (Silver, Libra) [13] have now either been integrated into Globus [15], or abandoned in favour of it.

1. Nimrod-G (`http://www.gridbus.org/`) is the most interesting of all the current grid meta-schedulers. It is part of the GRid Architecture for Computational Economy (the GRACE project) and as such it is an economy based scheduler. Nimrod-G also supports quality of service based scheduling (e.g. on the basis of deadlines and budgets).
2. GRaDS (`http://hipersoft.cs.rice.edu/`) is a project which aims to produce a software execution environment for code to be run on a computational grid. The GrADSoft Architecture attempts to adapt the application according to changes in the available resources whilst attempting to maintain as high performance as possible. It relies on a feedback loop arrangement to continually update it with the current status of its applications and nodes.
3. Condor-G (`http://www.cs.wisc.edu/condor/`) is a task broker designed to front end a computational grid. It acts as an entry point to the grid dispatching jobs to run on the various nodes available. Condor-G would appear to require control of the nodes scheduling policy.

These three scheduling systems would appear to have the following similarities which are typical of a "top down" architecture dependent on centralised control:

1. They all assume one entry point into the grid (i.e. they know about all the jobs on the whole grid of nodes).
2. They all assume that they control, or have sight of, the scheduling policies for all the nodes.
3. They all assume that the nodes are closely linked to the grid. i.e. The bit of the scheduler which allocates jobs to nodes will also be responsible for scheduling the execution of the jobs on the node.

It is proposed that, in a system where all nodes are autonomous and multiple agent systems are responsible for trading available resources on those nodes, this set of assumptions will not hold true. In the "real world" it is likely that:

1. There would be multiple trading floors i.e. "entrance points" to the grid.

2. The resource allocation would be done on the basis of market trading, which will determine the allocation of resources to nodes.
3. All nodes will have potentially different, unknown job execution policies which they may, or may not, adhere too.

4 Resource Allocation to Nodes

When an application is running, the environment in which it runs needs to monitor the resources it uses and report those back to (a) the scheduler and (b) the accounting system. From this, the agent on the trading platform responsible for selling capacity can determine at what level they should be selling.

A system in which there are multiple entities all attempting to acquire part of one entity is obviously going to experience concurrency problems. According to [20], the meta scheduler architecture devised in the GrADS system [3] suffers from a deficiency in that if two jobs are submitted to the grid at the same time then they will both be processed without regard.

This collisions problem would remain in our trading based resource allocation system. Indeed, we would expect it to be potentially worse as a trader might deliberately oversell or underprice. Whilst any sort of trading situation ought to contain the ability for buyers to ascertain past performance of a node (to prevent buyers purchasing capacity from an agent that regularly does not deliver their commitment) we believe that the problem of double-booking at the resource allocation level should be dealt with by market forces (*i.e. Traders should be able to oversell, but, if they do so and their clients suffer, we would expect clients to rate the trader badly and therefore the trader would not do so well next time.*

The idea of market based enforcement is one adopted in [20], which utilises a "Contract Negotiator". The GrADs Contract Negotiator acts as a queue manager ensuring jobs that have completed resource selection are launched on the grid nodes in the most efficient manner possible.

*For example if the performance of the new application can be enhanced by waiting for the current application to complete, it will hold the new application until the node is ready (*i.e. *a short wait may be preferable to complete reallocation.*

Once a node has received a job its job scheduling system should be responsible for ensuring minimal double booking (it may be acceptable for a system to overbook capacity, in the same way that airlines oversell their seating, however, too much overselling will lead to a damaged reputation).

In this proposed environment we can begin to support a grid whereby multiple resource selectors attempt to give jobs to multiple resources.

5 Affect of Trading Models on Resource Allocation

A Dartmouth Technical Report "Utility Driven Mobile-Agent Scheduling" [5] suggests that "markets" can be used to regulate agents on a trading floor. It is suggested that:

> *"Participants have a finite amount of currency, thus agents have incentive to attempt to evenly spread themselves throughout the network." [5]*

i.e. *Having fixed amounts of resources requires the agents to:*

1. *Be careful in their spending, as they must budget their finances appropriately to ensure they can purchase all the required resources.*
2. *Be wary of potential suppliers. For example, the report suggests:*

> *"As more agents arrive at the site, the server will most likely raise the price of service to maximise profits." [5]*

A market based system also has some interesting effects on the efficiency of resources. Consumers will not necessarily purchase resources on the basis of their availability. They might purchase the cheapest resources (thus allowing them to purchase as many resources as possible) or they might purchase expensive resources (perhaps hoping these are of better quality). However, as the report discusses:

> *"A market's efficiency depends on the consumers' ability to assess their needs and then make rational decisions that maximise their utility." [5]*
>
> *i.e. The computational market could collapse, but we have typically found that agent based markets would achieve a fairly happy equilibrium.*
>
> *It should be noted that "There are few electronic studies about the source or measurement of agent utility in an electronic market" [5], however, we can assume that factors such as:*
>
> - *Quality of Service*
> - *Completion Time*
> - *Likeliood of Completion*
> - *Accuracy of result*
> - *...as well as cost*
>
> *...will all be likely factors in determing the choice of grid node to complete a job.*
>
> *Work done by Chun and Culler [12] suggests there is an additional factor to consider when evaluating how jobs should be excecuted. They propose the concept of a* "user-centric performance metric" *as the basis for system evaluation.*
>
> > *"In our analysis, each user is modeled as having a utility function for each job which measures value as delivered to the user as a function of execution time." ... "To measure overall system performance in a user-centric manner, all users should be taken into account and value delivered to users should be the basis of performance." [11]*
>
> *The Chaun and Culler [12] metric works much like a real world customer satisfaction survey and indeed it is quite sensible to build on traditional evaluation techniques such as surveying.*
>
> *The key difference between "traditional" systems and market driven systems being, therefore that we can be nearly certain that resources* will not *be allocated solely on the basis of available capacity. Resources will, instead, be allocated on the basis of who can afford them.*
>
> *The work done on the "GridBank" project supports this hypothesis by suggesting that: "It was observed that the utility delivered by resources is enhanced when resource allocation is performed based on user's quality-of-service (QOS) requirements/constraints (e.g., deadline and budget)." [6]*
>
> *In his thesis [6], Buyya conducts an experiment which demonstrates that using "a competitive commodity-market economy" (i.e. where prices fluctuate for resources, in this case on a time basis) allows the job broker to schedule more important jobs during "peak" hours and a greater number of less important jobs during "off peak" hours. i.e. By forcing users to prioritise their jobs and splitting them better results are achieved than under the traditional "all at once" paradigm.*
>
> *A second experiment (in which Buyya fixed the deadline and budget) produced additional results. He found that his job broker was able to schedule more jobs earlier using its time-optimisation strategy (than it could using a cost optimisation strategy) but that it was more expensive for the "end user".*
>
> *The work completed by Buyya et al. clearly shows that if a user is allowed to define the priority and "cost" to complete a task then it makes the scheduling decisions much easier to compute and ultimately, the resources achive better utilisation.*

6 Job Scheduling within Nodes

A node in a grid may be a single CPU. Equally, it may be a vast super computer or a private array of workstations. It is envisaged that each node in the grid will have a single scheduling policy and a single high level scheduler. It is not necessary for the high level scheduler to only accept instructions from one client, nor is it necessary for the individual machines within the node to only accept instructions from the high level scheduler (e.g. there might be a local user who demands priority).

A number of scheduling possibilities exist. They include:

1. *Splitting a processor into time slices and allocating all jobs to it equally or on a first come first served basis.*
2. *Evaluate the current load on the machine against the loading requirements given by the application ? (i.e. Estimate which job will complete fastest and run it first.)*
3. *Determining which task the users want to complete first (market based scheduling).*

The GrADS project [20] defined an interesting set of objectives for their scheduling system:

1. *"Verifying that the applications made their scheduling decisions based on conditions of the system when competing applications are executing."*
2. *"Accommodating short running jobs by temporarily stopping long running and resource consuming jobs."*
3. *"Facilitating new applications to execute faster by stopping certain competing applications."*
4. *"Minimising the impact that new applications can create on already running applications."*
5. *"Migrating running applications to new machines in response to system load changes to improve the performance or to prevent performance degradation."*

Within the context of a node, it is clear that much work has been done on determining an effective scheduling strategy.

7 Task Reallocation

Typically task reallocation can occur when a given node needs to execute some other process more urgently than the one in which it is currently executing. The most common instance of this in current usage is when a user returns to their desktop computer and begins using it again - because mostly, grid-computing resources are pooled from idle personal computers on the understanding that owners can have the use of their computer back when necessary.

However, it is suggested in [20] that task reallocation should also take place when the scheduler feels that a particular application would benefit, either by temporarily suspending a competing job or by moving the current job to another, presumably more lightly loaded, CPU. This approach is adopted because the Meta scheduler in [20] works toward achieving job completion within a set time frame, where as the more normal approach is to complete a job "as and when".

The concept of task reallocation becomes more interesting when a scheduling system based on monetary value is introduced. In this instance, it is possible that a remote user might offer sufficient reward to the owner of the node that the node owner might decided to surrender their veto and let their computer be used by the remote user.

8 Nimrod

Nimrod [18] is a grid scheduling system developed in Australia at Monash University. Nimrod was a "first cut" at implementing a system to distribute parametric computing problems across a set of computing resources on the basis of a computation economy where by the user could determine the available budget and the priority of the job.

It is suggested in [1] that Nimrod worked successfully for static sets of computational resource but that it exhibited a number of flaws which rendered it unsuitable when "implemented in the large-scale dynamic context of a computational grid" [1]. Larger scale grids in the "real world" typically exhibit a number of different properties:

1. *Nodes in the grid are typically scattered across a number of different administrative domains.*
2. *Each domain will typically have their own resource allocation policy, determining for example, if resources must be surrendered to local users.*
3. *Each domain will have their own job queueing system, their own access cost, and various amounts of available computational power.*

The new system "Nimrod/G", which uses the Globus [15] middleware for interfacing with the grid, has been designed to address these shortcomings.

Nimrod/G utilises a "Job Wrapper" which is responsible for the staging of application tasks and data. The "Job Wrapper" acts as a mediator between the parametric engine and the actual machine on which the task runs. Its primary function is to send the results from the remote client back to the main engine. It is therefore conceivable that it could also monitor the resource utilised by the remote client and report that data to the resource accounting system.

Nimrod/G supports an integrated computational economy in its scheduling system. This means that Nimrod/G can schedule jobs on the basis of deadlines and budget. Therefore users can make assertions such as:

- *"Please complete this task by date X and within budget Y"*
- *"If you can complete task X by date Y I shall provide reward Z"*

9 Feedback from Job Execution into the Scheduling & Accounting System

The AppLeS project [4] is a set of interesting experiments into developing an "application-management system" which sits on top of grid middleware, such as Globus [15], Legion [17] and PVM [19]. AppLeS software uses this grid middleware to handle resource management whilst the AppLeS system focuses on prioritising the applications. The AppLeS system is an agent based one. Each agent is responsible for a particular task. AppLeS agents use a "Heterogeneous Application Template" (HAT) to allow a user to specify the structure, characteristics and the current implementations of the application that needs to be run and its tasks. A "User Specification" is also used to provide information on the user's performance, timing and other criteria.

The AppLeS Project [2] considers "Application-Level Scheduling" - a paradigm in which applications are scheduled on the basis of feedback from the host system. Most importantly, when considering a change to the system "Application-Level Scheduling" considers the impact of the change on the performance of the application.

Clearly, this is a helpful approach to take because applications can execute more quickly and efficiently if they are closely matched with the resource capabilities of a system.

There are a number of different methods for determining the best resource match. Firstly, we would expect that all entrants to the grid would pre-package their software with a "resource profile" in order to describe resources needed to execute that particular job. This information is helpful, firstly on the trading platform to ensure that sufficient resources are bought/sold to enable the job execution but secondly, at the node scheduling level, to allow CPU schedulers to determine the most efficient order of job execution.

However, in order to ensure that any resource profile is as accurate as possible there needs to be feedback about actual usage once the application has been deployed. This data should ideally be captured by the operating system (could be a virtual machine, e.g. the Java Virtual Machine) which supervises the application execution and feedback into the resource allocation system.

The AppLeS Project [2] contains a number of elements which support resource allocation on the basis of performance profiles (e.g. their "Planner", "Performance Estimator" and "Actuator") but none of the components of the AppLeS project support a feedback mechanism through which performance can be monitored and tasks adjusted.

It should be noted that whilst it is desirable for the host operating system to report back to the grid accounting and allocation systems little work has been done on this area. Some work, for example that completed by Chun and Culler in their paper "Market-based Proportional Resource Sharing for Clusters" [12] suggests that most of the focus has been on the development of mechanisms for providing quality of service for CPU time and that whilst there have been many novel suggestions and academic experiments, none of the proposed solutions have been widely implemented - perhaps because so far, contention for normal CPU resources has been very minimal.

10 GridSim

GridSim [14] (`http://www.gridbus.org/gridsim/`) is a toolkit which allows modeling and simulation of entities in parallel and distributed computing (PDC) systems-users, applications, resources, and resource brokers (schedulers) for design and evaluation of scheduling algorithms.

10.1 Brief Overview of GridSim

GridSim comes in three parts. The first is essentially a Java API which can be used to create simulations of grid computer networks. It contains classes which represent nodes, machines, IO data etc.

The second is a graphical interface (Visual Modeler) which can be used to generate java code which is then run in conjunction with the GridSim library, to actually produce the simulation.

It is important to remember that GridSim was designed for testing out different scheduling algorithms, rather than simply emulating a grid. To this end, the third component is a GridBroker library. This is used to emulate the resource broker and report on scheduling efficency. Typically, the person completing testing would implement their own scheduler and plug this into the GridBroker class library.

10.2 Building GridSim

GridSim was origionally packaged to be built with JBuilder. In order to make it more useable the first activity was to re-package it so that it could be used with the open

source java build tool "ant".

The GridSim code was already split into three sections, but it had not been split into java packages. Further, the code did not entirely conform to the java standard of having file names correspond exactly to class names. However, having packaged the code and tidied the file structure it was reasonably straight forward to build the code using ant.

Our packaged version of GridSim with Ant is available at `http://www.cs.bath.ac.uk/~csppjg/gridsim-ant.tar.gz`.

10.3 Trial Simulation

Using the Visual Modeler tool in the GridSim package a simple simulation supporting two nodes and two jobs was created. The Visual Modeler tool generates Java source code which can then be executed using the gridsim toolkit in order to run a scheduling simulation.

Our simple simulation was mainly done in order to test the packaging of the simulator. In order to maintain consistency, we developed a new package "sim" and added the generated class to it.

Finally, we added an ant build task to tie together all the dependencies, i.e. ensure the entire system is compiled and execute the simulation.

10.4 Evaluation of GridSim

GridSim is clearly a useful tool for evaluating different grid computing network scheduling algorithms. It provides an easy to use mechanism for creating a grid simulator into which code can be "plugged in" for execution and evaluation.

However, its use beyond simulation of grid scheduling systems is limited and it cannot be used to simulate a grid for the execution of other programs.

If we develop an agent based grid resource management system then gridsim might be a useful tool for its evaluation.

11 Use of Agents for Grid Resource Management

Nearly all of our evaluation so far has focused on what the various methods for allocating resources on a grid are available and we have investigated whether any of those methods utilise agent based systems as their core. Perhaps suprisingly, we have found very little evidence of such work.

Reversing the question provides further insight into some of the issues as there are a number of research groups who have considered how agents may be used to managed grids.

The majority of the work has been completed in Warwick, UK and at Vrije Universiteit Amsterdam, NL.

11.1 AgentScape

The AgentScape project [21] provides a multi-agent infrastructure that can be employed to integrate and coordinate distributed resources in a computational grid environment.

The objective of the AgentScape system is to provide a "minimal but sufficient" environment for agent applications. It differs from traditional agent platforms by ex-

tending the CoABS grid project (`http://coabs.globalinfotek.com/`) and focusing on the "scalability, security and extensibility" of the agent environment.

The AgentScape system provides a number of components, notably a kernel, a number of directory and resource discovery services, etc. In their paper [21], the authors also discuss how the AgentScape system could be used to support the management of grid systems. However, it would appear that they have not yet completed any work in this area and that AgentScape is still a prototype agent platform.

11.2 A4 (Agile Architecture and Autonomous Agents) and ARMS

The A4 (`http://www.dcs.warwick.ac.uk/research/hpsg/` and `http://www.ccrl-nece.de/~cao/A4/`) project at Warwick has likewise developed a framework for agent based resource management on grids. The researchers at Warwick have produced a number of papers, including [10], [8] describing their work.

A4 is a a design methodology which focuses on solving the problems associated with scalbility and dynamism in large multi-agent systems.

The ARMS project is an Agent Based Resource Management project for grid computing networks which uses the A4 methodology and ideals.

> *"ARMS couples the performance prediction techniques of the PACE toolkit with a scheduling algorithm designed to manage a local grid resource. At the meta-level, ARMS utilises the agent-based methodology described in [9], where each agent acts as a representative for a local grid resource and considers this resource to be its high performance computing capability." [8]*

In the ARMS system agents are set-up to be both buyers and sellers of computational services. Agents perform cooperative service advertisment and discovery which allows them to work together to schedule applications on the grid.

The Agent scheduling service operates at the "service level" with agents advertising their resources as "services". Once the agent has sold the resource the actual internal-resource scheduling is done using a predictive performance tool called PACE (Performance Analysis and Characterisation Environment) [7].

It is not clear whether agents in the ARMS system bundle cpu resources together to form a service or indeed what the differences in granularity are between the PACE scheduler and the agent based resource negotiation however, it seems that the use of agents is primarily targetted towards service discovery whilst the PACE scheduler is targetted torwards the machine level scheduling.

12 Conclusion

Having discussed the current available methods of grid scheduling, we have seen that a number are available. However, they all suffer from limitations in that they do not separate node allocation from node scheduling and they insist upon control of both the node allocation and the local node scheduling policies.

Further, we have considered the use of Agents in scheduling and we have seen that whilst multi-agent systems are used to trade for grid resources, this tends to happen at the higher "services" level and not at the base "resource" level.

We see that raw computational resources are normally scheduled using a more classical scheduler, using performance prediction or time based technqiues.

12.1 Blueprint for Agent Based Grid management system

A real world grid computing system is composed of multiple layers. At the bottom there are simple resources, in the middle, those resources come together to create a service. Finally, at the top, end-users utilise the services to complete their tasks.

We consider that it is likely that individual resources (e.g. individual CPU, Memory, Databases, etc.) are unlikely to be useful on their own.

It is thus envisaged that a two tier trading situation could develop:

1. *Firstly, we consider that it is likely that individual resources (e.g. individual CPU, Memory, Databases, etc.) are unlikely to be useful on their own. Therefore, we think that agents can be used to negotiate with other service offering agents (through a process of "resource discovery") and thus create useful "service bundles". It might be that this negotiation is done on a monetary basis, but we think it is more likely that agent resource managers will operate cooperatively, sharing the benefit of the cooperation after the service has been sold to a client.*
2. *Secondly we think that there will be a market place of services where by end users and service providers trade freely for "service bundles".*

It is proposed that a real world grid "service bundle" trading system would need to support:

1. *Multiple trading floors i.e. "entrance points" to the grid.*
2. *The resource allocation would be done on the basis of market trading, which will determine the allocation of resources to nodes.*
3. *All nodes will have potentially different, unknown job execution policies which they may, or may not, adhere too.*

There does not appear to be a grid scheduling system that meets this proposed specification.

References

[1] *David Abramson, Rok Sosic, J. Giddy, and B. Hall. Nimrod: A tool for performing parameterised simulations using distributed workstations. In* HPDC*, pages 112–121, 1995.*

[2] *F. Berman and R. Wolski. The AppLeS Project: A Status Report, 1997.*

[3] *Francine Berman, Andrew Chien, Keith Cooper, Jack Dongarra, Ian Foster, Dennis Gannon, Lennart Johnsson, Ken Kennedy, Carl Kesselman, John Mellor-Crummey, Dan Reed, Linda Torczon, and Rich Wolski. The GrADS Project: Software support for high-level Grid application development.* The International Journal of High Performance Computing Applications*, 15(4):327–344, 2001. Available via* `http://www.cs.rice.edu/~johnmc/papers/GrADS-JHPCA-01.pdf`.

[4] *Francine Berman and Richard Wolski. Scheduling from the perspective of the application. In* HPDC*, pages 100–111, 1996.*

[5] *Jonathan Bredin, David Kotz, and Daniela Rus. Utility driven mobile-agent scheduling. Technical Report PCS-TR98-331, Dartmouth College, 1998. Available via* `http://citeseer.nj.nec.com/bredin98utility.html`.

[6] *Rajkumar Buyya.* Economic-based Distributed Resource Management and Scheduling for Grid Computing*. PhD thesis, Monash University, Melbourne, Australia, 2002. Available via* `http://www.buyya.com/thesis/`.

[7] *Junwei Cao, S. Jarvis, D. Spooner, J. Turner, D. Kerbyson, and G. Nudd. Performance prediction technology for agent-based resource management in grid environments. In* 16th International Parallel and Distributed Processing Symposium (IPDPS '02 (IPPS and SPDP)), *page 86, Washington - Brussels - Tokyo, April 2002. IEEE. Available from* `http://www.dcs.warwick.ac.uk/~hpsg/html/downloads/public/docs/CaoJ.PPTARM.pdf`.

[8] *Junwei Cao, Stephen A. Jarvis, Subhash Saini, et al. ARMS: An agent-based resource management system for grid computing.* Scientific Programming, *10(2):135–148, 2002. Available from* `http://www.dcs.warwick.ac.uk/~saj/papers/arms.pdf`.

[9] *Junwei Cao, Darren J. Kerbyson, and Graham R. Nudd. Use of agent-based service discovery for resource management in metacomputing environment.* Lecture Notes in Computer Science, *2150:882–??, 2001.*

[10] *Junwei Cao, Daniel P. Spooner, Graham R. Nudd, et al. Agent-based resource management for grid computing, 2002. Available at* `http://www.dcs.warwick.ac.uk/~hpsg/html/downloads/public/docs/CaoJ.ARMGC.pdf`.

[11] *B. CHUN and D. CULLER. User-centric performance analysis of market-based cluster batch schedulers. Available via* `http://berkeley.intel-research.net/bnc/papers/ccgrid02.pdf`. *Last Checked 20030214.*

[12] *B. CHUN and D. CULLER. Market-based proportional resource sharing for clusters, 1999. Available via* `http://citeseer.nj.nec.com/238183.html`. *Last checked 20030213.*

[13] *Sebastian Ho. Survey of a grid meta-scheduler, 2002. Available via* `http://student.bii.a-star.edu.sg/~sebastianh/Scheduler_Survey.pdf`. *Last checked 20030213.*

[14] *M. Murshed, R. Buyya, and D. A. Abramson. Gridsim: A toolkit for the modeling and simulation of global grids., 2001.*

[15] *Globus Project.* `http://www.globus.org/`.

[16] *Grid Scheduling Dictionary Project. Grid scheduling dictionary project. Available via* `http://www-unix.mcs.anl.gov/~schopf/ggf-sched/GGF5/sched-Dict.1.pdf`.

[17] *Legion Project.* `http://www.cs.viginia.edu/~legion/legion.html`.

[18] *Nimrod Project.* `http://www.csse.monash.edu.au/~davida/nimrod/`.

[19] *PVM Project.* `http://www.em.ornl.gov/pvm/`.

[20] *S. S. Vadhiyar and J. J. Dongar. A metascheduler for the grid, 2002. Available via* `http://www.cs.utk.edu/~vss/publications/vadhiyar-metascheduler.pdf`. *Last checked 20030213.*

[21] *Niek J. E. Wijngaards, B. J. Overeinder, Maarten van Steen, and Frances M. T. Brazier. Supporting internet-scale multi-agent systems.* Data Knowledge Engineering, *41(2-3):229–245, 2002. Available from* `http://www.iids.org/publications/bnaic2002_dke.pdf`.

[22] *Vipin Kumar William Leinberger, George Karypis. Job scheduling in the presence of multiple resource requirements. Technical report, Department of Computer Science and Engineering, University of Minnesota, 1999. Available from* `http://www.supercomp.org/sc99/proceedings/papers/leinberg.pdf`.