# Auction Resource Allocation Mechanisms in Grids of Heterogeneous Computers

TIMOTHY M. LYNAR
The University of Newcastle
Ourimbah, NSW, 2258
AUSTRALIA
tim.lynar@newcastle.edu.au

RIC D. HERBERT
The University of Newcastle
Ourimbah, NSW, 2258
AUSTRALIA
ric.herbert@newcastle.edu.au

SIMON
The University of Newcastle
Ourimbah, NSW, 2258
AUSTRALIA
simon@newcastle.edu.au

*Abstract:* This paper examines economic resource allocation through a number of auction types for a grid of e-waste computers. It examines the time to complete tasks and the energy usage of completing the tasks on a grid. A model of a simulated grid is developed and used to evaluate the resource allocation mechanisms. The model is an agent-based simulation where by user agents submit tasks to node agents that process these tasks. We evaluate three types of resource-allocator agents which all use a type of auction. The auction types are batch auction, continuous double auction and a pre-processed batch auction. The pre-processed batch auction is developed to try to have the advantages of both the continuous double auction and the batch auction. The simulated grid is calibrated to a real e-waste grid where each node has a performance index. This grid is a test grid of eight nodes of heterogenous computer hardware and with differing computational ability and energy usage. We simulate the auction types under the same task input streams. We consider a task impulse response stream on energy usage and time to complete all tasks and a input stream step response. Finally we consider the three auction allocation mechanisms under a random task stream. The paper finds that the choice of auction method makes a substantial difference in the time to complete tasks and in total energy consumption.

*Key–Words:* grid computing, resource allocation, auctions, e-waste, energy consumption

## 1 Introduction

A computing grid is a network of distributed hardware and software resources that can be utilized to compute common tasks [22]. Foster and Karonis [10] liken computing grids to the power grid, where users and resources are distributed and users have access to a dependable and consistent supply of computational power: "A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities" [10, p.3]. Grids typically span many geographical locations and contain heterogeneous resources with varying computational abilities [33]. Often grids are made of existing computing resources (e.g. research computing clusters and high performance computers) logically formed into a consolidated computing resource. A specific example is the world-wide Large Hadron Collider (LHC) Grid, which is made up of 140 computing centers in 33 countries. Computing grids are becoming increasingly important and have been used in many applications [27].

There are many claimed potential benefits to grid computing including the ability to utilize an existing high performance computing environment for a low price [10]. Supercomputers can solve problems in seconds that would take a personal computer days to solve, but the speed of a supercomputer comes at a high monetary cost. Schneck [28] found that supercomputers are by far the most expensive form of computer. Grid computing offers one way of gaining the power and speed of a supercomputer for a fraction of the price.

The cost savings can be even greater when the grid is made up not of new hardware but of computers destined for the scrap-heap. The problem of electronic waste (e-waste) or waste electrical and electronic equipment (WEE) is inescapable in today's society, and personal computers contribute significantly to this problem. Most personal computers are dumped after just a year or two of use, at a huge cost to health and the environment, as their owners succumb to the desire to keep up with the ever-increasing power of new computers. One way to help reduce e-waste is to extend the life of obsolete computers by using them in grid computing. We have built a grid of obsolete desktop computers that were destined for the waste stream, and are using it to explore aspects of computational grids.

Within a grid, the allocation of computer node re-

sources to computing tasks is carried out by a resource allocation mechanism. In this paper we examine three different economic resource allocation mechanisms, based on different auctions, for a grid of electronic waste computers. Our aim is to create a resource allocation mechanism that dynamically allocates resources over a heterogeneous grid in a way that both maximizes the throughput of tasks and minimizes energy consumption.

## 1.1 Grid Resource Allocation

A significant body of research on resource allocation and scheduling has proposed a number of different allocation mechanisms, many of which rely on task execution time predictions and on the assumption that these predications are accurate. These studies emphasize the importance of the grid workflow [26]. Ali et al. [2] utilize an 'expected time to execute' metric which contains pre-processed estimates for all expected tasks on each computational resource. These values are typically assumed and can come from benchmarking, task profiling, prior executions or user input. The Casanova et al. [5] scheduling system assumes that accurate estimates of execution time are provided to the system from past execution history or from input by the user. Smith et al. [29] selectively utilize past execution history in combination with genetic algorithms to predict the execution times of tasks. Kim et al. [20] examine heuristic techniques of scheduling dynamically in heterogeneous computing environments. Execution times are calculated using a heuristic method known as gamma distribution. The Spooner et al. [30] approach to estimating task execution time utilizes descriptions of resources and modeling of tasks to produce estimates of application runtime. Many resource allocation schedulers are not based on economic principles, for example Santiago et al. [26].

There have been a number of economic-based grid and cluster resource allocation mechanism studies [see, for example, 8, 18, 36, 37, 38]. Many of these mechanisms have limitations when applied to the grid environment. These limitations include not accounting for the heterogenous nature of grid computing, assuming that the computing resources are reliable, and requiring users to know how long their job will take to execute on unknown hardware [3, 4, 6, 12, 32].

## 1.2 Green Computing

While the prime focus of high-performance computing has historically been on performance, there is now an increasing focus on energy conservation [17]. E-waste computers, such as those we have used to build our grid, have significant variation in their power usage and their computational performance ability. For environmental reasons, a performance metric for such a grid should take account not only of a node's processing ability but also of its power usage.

The consideration of energy usage as a primary concern in high performance computing is relatively recent. There has been much research done in recent years on energy consumption in cluster computing and data centers [9, 11, 13, 14, 15, 16, 17, 19, 24, 25, 31, 34, 35]. However, little has been published on energy conservation in high-performance grid computing [39]. The few notable exceptions include Patel et al. [23], who examine the idea of exploiting the heterogenous nature of distributed data centers to conserve energy.

Few economic resource allocation systems consider the energy usage of the node resources. This is particularly relevant with older computer systems as there was less emphasis on 'green computing' in their design. In previous work we have shown how energy usage of a computing resource can be defined and used for an e-waste grid by giving each node a performance index that includes both computational performance and energy usage [21]. In this paper we use this performance index for a resource node in the allocation mechanism.

The aim of the performance index is to indicate, through ranking, which nodes have the most desirable mix of low power requirements and high computational performance. There are few current mechanisms available that perform this dual function, although there are many current measures of a node's performance available, including both synthetic and non-synthetic benchmarks.

The different nodes in a heterogenous grid could be ranked by many different measures of performance, including power, floating point operations per second (Flops), and any other synthetic or non-synthetic benchmark that can be executed over the grid. One measure that accounts for both a node's computational performance and its utilization of power has been described as the "power-performance ratio" [17]. These measures such as $\frac{flops}{watts}$ have been utilized by many projects as a measure of power to performance [1].

Benchmarks are an important measure of performance and allow for the easy comparison of one node against another. However, computational performance alone does not give the resource allocation mechanism enough information to discern which resource is most desirable to utilize. The resource allocation mechanism must be able to rank nodes accord-

ing not only to their computational performance but also to their power requirements. We created a performance index to give each node in the grid a comparative value based on its computational performance and power requirement. This comparative value can then be used by the resource allocation mechanism to determine which resources are most desirable to utilize at any given time. More details of the performance index can be found in Lynar et al. [21].

# 2 The Model

Our approach is to run the same input stream of tasks on the grid using different resource allocation mechanisms based on different auction types, and to compare the stream's completion time and the grid's energy usage for each mechanism. We have constructed a model of a grid and calibrated it to correspond to our actual grid. This small simulated grid consists of eight nodes, details of which are displayed in Table 1 – keeping it small allows us to easily compare the grid's responses to the different resource allocation mechanisms.

The model is agent-based. There is a set of user-agents that submit tasks to the grid, a set of resource node agents that process tasks, and a resource allocation agent that allocates the tasks to particular resources. Our focus is on differing economic resource allocation approaches by the resource allocation agent.

One issue that will significantly impact on the energy usage is the ability to power nodes off when they are not in use. An issue here is the question of when the additional energy savings of having the node turned off outweigh the additional energy required to turn the node on and off. This work is left for future research.

In our model, resource nodes consist of e-waste computers that offer computational power. The computers are heterogeneous and hence have differing computational performance and energy consumption. The grid receives from users an input stream of tasks that require computation and thus ask for resources. Each resource node then bids for tasks, the bid that it makes being its performance index. The performance indexes of the nodes in our model are shown in Table 1. Finally, the resource allocation mechanism allocates the tasks to the resources.

To thoroughly test the resource allocation mechanisms we subject each of them to three quite different input streams, an impulse stream, a step input stream, and a set of inputs submitted at random times over a fixed time horizon. The model is run with each of these streams on each of the different resource allocation mechanisms, and the stream execution time and the total energy consumption are compared.

Table 1: Table of inputs from real nodes showing watt usage when processing ($W_i$), Millions of Flops per second ($F_i$), Performance Index value ($P_i$), and processor type (CPU)

| *Node* | $W_i$ | $F_i$ | $P_i$ | CPU |
|---|---|---|---|---|
| 1 | 60 | 236.625 | 39.33 | Pentium 3 |
| 2 | 62 | 235.357 | 37.90 | Pentium 3 |
| 3 | 63 | 286.661 | 45.40 | Pentium 4 |
| 4 | 63 | 286.302 | 45.40 | Pentium 4 |
| 5 | 83 | 1544.0 | 186.02 | Athlon 64 5600+ |
| 6 | 103 | 1795.0 | 174.27 | Athlon 64 6000+ |
| 7 | 79 | 318.208 | 40.25 | Pentium 4 |
| 8 | 80 | 317.737 | 39.63 | Pentium 4 |

The structure of the simulation is described in Algorithm 1.

**Algorithm 1** The algorithm of the simulation

```
Define Resources
Define Workload
for all Time steps do
    Choose resource allocation mechanism
    Assign tasks
    Analyse resources
    Process tasks
    Record statistics
end for
```

## 2.1 Model Assumptions

To simplify the model we make the following assumptions:

- a resource can only process one task at a time;
- the computational requirement of incoming tasks is unknown;
- any resource can process any task;
- each task can be processed on a single resource; and,
- bids from resources arrive in a random order.

## 2.2 Auction Methods

In the resource allocation mechanism we compare three auction types, a batch auction, a continuous double auction, and a pre-processed batch auction.

### 2.2.1 The batch auction

The batch auction (Algorithm 2) gathers bids from all of the resources, sorts the bids on their size, and then assigns incoming tasks to resources in order of submission.

Batch auctions can guarantee that the fastest available resource will win, however the batch task of gathering bids can result in a significantly delayed start for execution of the resource [6].

In our model the batch auction takes place every ten seconds and input tasks are queued to be allocated. A resource may be busy completing a task previously allocated to it, in which case it will not take part in the current auction.

**Algorithm 2** The batch auction

```
loop
    The allocation mechanism requests each resource for a bid
    The mechanism waits for the bids to be returned or the auction to time out
    The mechanism sorts resources' bids so that the resource with the highest bid is first
    for all Tasks (in order of submission) do
        Assign task to next unallocated resource
    end for
end loop
```

### 2.2.2 The Continuous Double Auction

In a continuous double auction or CDA (Algorithm 3) the first bid that matches or exceeds an ask is allocated. Continuous double auctions cannot guarantee that the fastest available resource will win the auction [7]. This can be undesirable for a resource allocation mechanism in a grid.

In our model any resource's bid can match any task's ask as all tasks can be processed by any resource. Hence the first bid will always be allocated to the first ask. To simulate a real double auction, we shuffle bids in a random order. In this manner the CDA as implemented in this model is basically a random allocation.

**Algorithm 3** The continuous double auction

```
Shuffle known resources
loop
    for all Tasks (in order of submission) do
        for all Resources do
            if The resource is not assigned a task and task is unassigned then
                Assign task to resource
            end if
        end for
    end for
end loop
```

### 2.2.3 The Pre-processed Batch Auction

We developed a pre-processed batch auction or PPB (Algorithm 4) that aims to retain the advantage of being continuous like the CDA while still attempting to allocate the fastest available resource to the task with the greatest computational requirements. In the PPB auction there is a central database of current bids, and every time a node's status changes the node updates its bid on the database. This enables the fastest available resource to be selected for executing any given task, without the need to delay the execution while a time-consuming auction is conducted. In other words, the resource allocation mechanism remembers the prior bid of each resource and assumes that prior bid to be correct for the next ask. Because the resources continually update their bids, there is always a current bid available for the auction to sort by. That is, nodes place fresh bids whenever their status changes, and it is these bids that are used when a task requests a resource.

**Algorithm 4** The pre-processed batch auction

```
loop
    The mechanism constantly receives bids from resources and assumes the last bit to be accurate
    The resources are sorted so that the resource with the highest bid is first
    for all jobs (in order of submission) do
        for all Resources in order of bid size do
            if The resource is not assigned a job and the task has not already been assigned then
                Assign Job to Resource
            end if
        end for
    end for
end loop
```

Table 2: Time to Complete Impulse Response by Auction Type

| Number of tasks | Batch | CDA | PPB |
|---|---|---|---|
| 1 | 139 | 700 | 130 |
| 2 | 139 | 700 | 130 |
| 3 | 709 | 700 | 700 |
| 4 | 709 | 848 | 700 |
| 5 | 709 | 848 | 700 |
| 6 | 709 | 852 | 700 |
| 7 | 857 | 852 | 848 |
| 8 | 861 | 852 | 852 |

Table 3: Order in which Nodes Used in Impulse Response Results

| Auction | Node Order |
|---|---|
| Batch | 5, 6, 4, 3, 7, 8, 1, 2 |
| CDA | 3, 7, 8, 1, 4, 2, 5, 6 |
| PPB | 5, 6, 4, 3, 7, 8, 1, 2 |

Table 4: Time in seconds to complete steady state stream for each level of difficulty and each auction type

| | Batch | CDA | PPB |
|---|---|---|---|
| Easy | 450 | 50 | 50 |
| Medium | 452 | 89 | 89 |
| Hard | 458 | 212 | 202 |

# 3 Results

## 3.1 Impulse Response

A good understanding of the differences in the resource allocation mechanisms can be obtained by examining the time the system takes to complete a single task. This can be considered as a steady-state impulse response. Beginning with a single input task, we increase this to a stream of eight input tasks, at which point each of the eight nodes will have a task allocated to it. All of these eight tasks require the same computational effort.

Table 2 shows that for a single task the Batch auction based resource allocation mechanism completes the task in 139 seconds. The CDA takes 700 seconds and the PPB takes 130 seconds. Why are the batch based methods faster? This is because they allocate the task to the node with the highest Performance Index, which is typically the node with the greatest processing power. On the other hand the CDA mechanism, as implemented for this model, allocates the task to a random node. The PPB is faster than the conventional Batch mechanism as it relies on pre-lodged bids and so does not have to wait for all nodes to return a bid.

Table 2 also shows the response for completion time as the number of tasks is increased in steps of one until there is a task for each node. As the number of tasks increases the Batch and PPB take about the same amount of time and the CDA still illustrates the random allocation. (We use the same sequence of quasi-random numbers in each run in Tables 2 and 3, so as the number of tasks increases, the same nodes are allocated tasks in the same order.) When there are the same number of tasks as nodes, all mechanisms take about the same time to complete the tasks, but the Batch mechanism shows the additional overhead of calling for and waiting for all bids.

The order that resource nodes are allocated by the mechanisms is given in Table 3. For the Batch and PPB mechanisms this is the same order, and corresponds to the order of ranking of their Performance Index (see Table 1). For the CDA it is a random order.

One result that is clearly seen from Table 2 is that, when the tasks are all the same size, and there are fewer tasks than nodes (both somewhat artificial situations), there can be a substantial difference in the time of completion between the batch-based and the CDA mechanisms.

## 3.2 Step Input Response

For the next comparison of the resource allocation mechanisms we examine a step input, in which the input stream of tasks is one task per second for 50 seconds. We use three input streams of tasks – easy, medium and hard. Easy tasks are small enough that even the least efficient node can perform them in a single time step; medium tasks can be performed in a single time step by a node with the mean Performance Index value; and for hard tasks, even the best-performing nodes require more than a single time step to compute them.

Table 4 shows the time (in seconds) for all tasks to be completed for each input stream and each auction type. Perhaps most obviously, task difficulty makes very little difference with the Batch auction: while the medium and hard input streams take longer, the difference is very small. This is clearly not the case for the CDA and PPB auction mechanisms, in both of which the hard tasks take more than four times as long as the easy tasks.

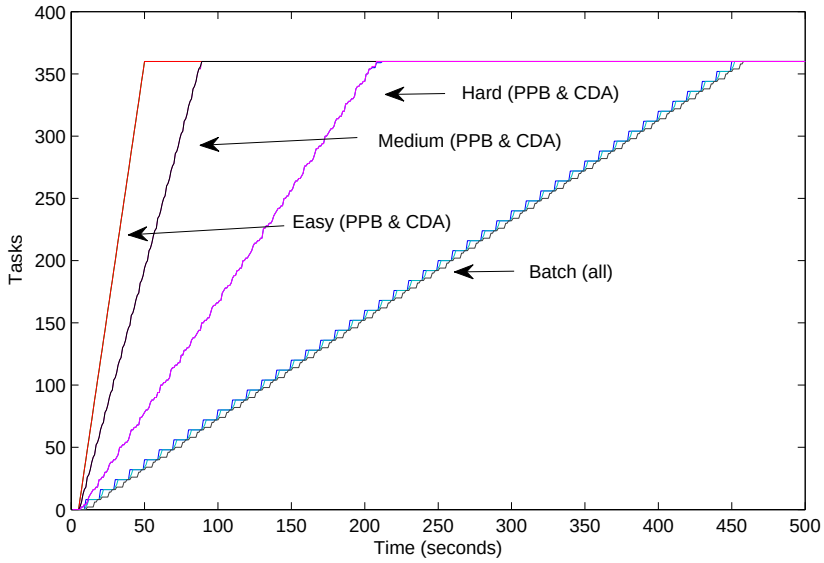


Figure 1: Cumulative Completed Tasks for Auction Types and Computational Effort from Step Input.

Faced with a steady stream of uniform jobs, the CDA and PPB mechanisms both outperform the Batch auction mechanism. For easy tasks they are nine times as fast, and for hard tasks they are about twice as fast. These differences are due principally to the Batch auction's overhead of polling the nodes and waiting for their responses each time an ask is submitted.

Of the two batch auction approaches, preprocessing is certainly worthwhile – partly because the model assumes that each node's bid will be the same as it was prior to the previous task, and that assumption is correct with this form of input stream.

The transient dynamics of the step input streams are shown in Figure 1. The figure plots the *accumulated* completed tasks in each second for the nine combinations of auction mechanism and task difficulty. A number of the lines overlap, leaving four result patterns, as labeled in the figure: easy CDA and PPB; medium CDA and PPB; hard CDA and PPB; and Batch. The overhead of conducting an auction every ten seconds can be seen in the stepped nature of the Batch auction responses.

Figure 1 shows that the step-response dynamics are linear for each of the combinations of auction type and task difficulty. This suggests that economies or diseconomies of scale are unlikely to result from scaling the system by changing the number of nodes.

In Figure 2 we present the dynamics of energy consumption at each second. The figure displays the transient energy usage of the grid whilst using the differing auctions. The energy consumption of the Batch auction constantly switches between high and low because of the idle time spent waiting for each auction to complete. This behavior indicates that the tasks are very small in computational requirements. The CDA and PPB auctions consume the same peak energy continuously until all tasks are complete, then drop back at the same time to the energy used by the grid when it is idle.

The energy consumption of idle nodes, as evident in Figure 2, highlights the potential value of powering down nodes when they are not in use. This is a feature that we intend to build into the model and implement in our physical grid of e-waste computers.

### 3.3 Random Workload Response

Our final results compare the resource allocation mechanisms with an input stream whose submissions are made at random times. We use seven workload sets with different combinations of computational effort and number of tasks. In each set, the input stream submits the same tasks, but with submissions occurring at random times within a fixed time horizon. We examine the time taken and energy consumed to complete all tasks in the stream.

The seven workload scenarios are given in Table 5. The first scenario (Set 1) consists of a small number of large tasks; Set 2, a small number of small tasks; Set 3, a large number of large tasks; Set 4, a large number of small tasks; Set 5, a small number of huge tasks; Set 6, a large number of huge tasks; and Set 7, a large number of tiny tasks.

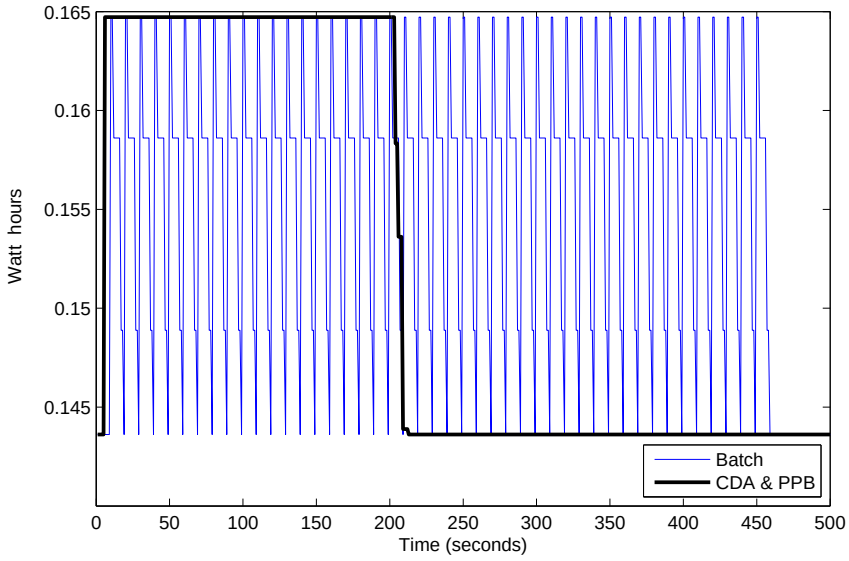


Figure 2: Energy Used for Auction Types and Computational Effort from Step Input.

Table 5: Workload Data Sets

| Set | Effort | Tasks | Batch | | CDA | | PPB | |
|---|---|---|---|---|---|---|---|---|
| | | | Time | Energy | Time | Energy | Time | Energy |
| 1 | 2000 | 200 | 7971 | 1437 | 7974 | 1439 | 7969 | 1437 |
| 2 | 200 | 200 | 7970 | 1436 | 7968 | 1437 | 7968 | 1436 |
| 3 | 2000 | 2000 | 8001 | 1455 | 7993 | 1466 | 7993 | 1446 |
| 4 | 200 | 2000 | 8000 | 1441 | 7992 | 1441 | 7992 | 1440 |
| 5 | 20000 | 200 | 9037 | 1609 | 8706 | 1609 | 7992 | 1440 |
| 6 | 20000 | 2000 | 997549 | 144157 | 997543 | 146407 | 997546 | 144168 |
| 7 | 20 | 2000 | 8000 | 14421 | 7992 | 1441 | 7992 | 1440 |

The high and low values were chosen based on the computational capability of the nodes used in the model. The least powerful node would take two seconds to compute a task with a computational requirement of 200 and would take 20 seconds to compute a task with a computational requirement of 2000. The most powerful node would take less than one second to compute a task with a computational requirement of 200 but would take 3 seconds to compute a task of 2000.

Table 5 shows the time to complete all tasks and the total energy used to complete the tasks for each of these input streams. Notice that in these results the Batch auction based mechanism performs about as well as the other mechanisms. This may be because the flow of tasks over time is steady and the overheads of the periodic auction are counter-balanced by allocating the tasks to the better performing nodes compared to the faster allocation of tasks in the other allocation mechanisms.

Comparing the auction types on time-to-complete and energy usage for the different workload scenarios it can be seen that there is little to prefer one auction type over the other. All resource allocation mechanisms produce about the same results for each of the workloads. However, it should be noted that these workflows consist of tasks that can be quickly computed on any resource.

In terms of energy consumption, Set 6, the most demanding workload, shows the Batch auction performing best, although not by a great deal.

# 4 Conclusion

In this paper we have implemented a model of a grid of e-waste computers, calibrating the model against a real physical grid. We have then examined the performance of three economic resource allocation mechanisms based on different auction types.

The auction-based resource allocation mechanisms were tested by considering a variety of small simulated workflows of user-allocated tasks and examining the time to complete the task workflow and the energy consumption in completing the workflow. The workflow types comprised an impulse workflow, a step workflow and a random workflow.

We have shown that the choice of auction method to be used in the economic resource allocation mechanism can make a difference in the time to complete tasks. It can also make a difference in total energy consumption used by the workflow. We discovered that the effect of changing the mechanism is highly dependent on the workflow.

In the workloads examined for this paper, all tasks had the same size, so each task required the same computational effort as the preceding task. The tasks in each workload were also generally evenly spaced. In future work we plan to introduce variability into task size and spacing.

Another issue that can be seen in this paper is that the e-waste computers in our grid tend to consume substantial energy when they are idle. In further work we will examine power-on-power-off as a means to save energy.

We also intend to implement these resource allocation mechanisms on our physical grid and more substantial grids, and to introduce larger and more realistic workflows. Further we will do some closed-loop work on dynamic power-on-power-off of nodes and on dynamically choosing the auction type to suit the circumstances.

*References:*

[1] Adams, J. C. and Brom, T. H. [2008], Microwulf: a beowulf cluster for every desk, *in* 'SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer science education', ACM, New York, NY, USA, pp. 121–125.

[2] Ali, S., Siegel, H. J., Maheswaran, M., Hensgen, D. and Ali, S. [2000], 'Representing task and machine heterogeneities for heterogeneous computing systems', *Journal of Science and Engineering, Special 50 th Anniversary Issue* **3**, 195–207.

[3] Bubendorfer, K. and Thomson, W. [2006], Resource management using untrusted auctioneers in a grid economy, *in* 'Second IEEE International Conference on e-Science and Grid Computing', pp. 74–84.

[4] Cao, J., Spooner, D. P., Jarvis, S. A., Saini, S. and Nudd, G. R. [2003], Agent-based grid load balancing using performance-driven task scheduling, *in* 'Proceedings of the International Parallel and Distributed Processing Symposium', pp. 10–20.

[5] Casanova, H., Zagorodnov, D., Berman, F. and Legrand, A. [2000], Heuristics for scheduling parameter sweep applications in grid environments, *in* 'HCW '00: Proceedings of the 9th Heterogeneous Computing Workshop', IEEE Computer Society, Washington, DC, USA, p. 349.

[6] Chun, B. N. and Culler, D. E. [2000], Market-based proportional resource sharing for clusters, Report UCB/CSD 00/1092, Berkeley : Computer Science Division, University of California,.

[7] Dash, R. K., Vytelingum, P., Rogers, A., David, E. and Jennings, N. R. [2007], 'Market-based task allocation mechanisms for limited-capacity suppliers', *IEEE Transactions on Systems, Man and Cybernetics* **37**(3), 391–405.

[8] de Assunçaõ, M. D. and Buyya, R. [2009], 'Performance analysis of allocation policies for intergrid resource provisioning', *Information and Software Technology* **51**(1), 42 – 55.

[9] Elnozahy, E., Kistler, M. and Rajamony, R. [2003], *Energy-Efficient Server Clusters*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, chapter Power-Aware Computer Systems, pp. 179–197.

[10] Foster, I. and Karonis, N. T. [1998], A grid-enabled MPI: Message passing in heterogeneous distributed computing systems, *in* 'SC Conference', IEEE Computer Society, Los Alamitos, CA, USA, p. 46.

[11] Freeh, V. W., Pan, F., Kappiah, N., Lowenthal, D. K. and Springer, R. [2005], Exploring the energy-time tradeoff in mpi programs on a power-scalable cluster, *in* 'IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)', IEEE Computer Society, Washington, DC, USA, p. 4.1.

[12] Germain-Renaud, C., Loomis, C., Mościcki, J. T. and Texier, R. [2008], 'Scheduling for responsive grids', *Journal of Grid Computing* **6**(1), 15–27. Great article.

[13] Harada, F., Ushio, T. and Nakamoto, Y. [2006], Power-Aware Resource Allocation with Fair QoS Guarantee, *in* 'RTCSA '06: Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications', IEEE Computer Society, Washington, DC, USA, pp. 287–293.

[14] Heath, T., Diniz, B., Carrera, E. V., Meira Jr., W. and Bianchini, R. [2005], Energy conservation in heterogeneous server clusters, *in* 'PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming', ACM, New York, NY, USA, pp. 186–195.

[15] Hsu, C. and Feng, W. [2005*a*], A Feasibility Analysis of Power Awareness in Commodity-Based High-Performance Clusters, *in* '7th IEEE International Conference on Cluster Computing (CLUSTER'05)', Boston, Massachusetts.

[16] Hsu, C. and Feng, W. [2005*b*], A Power-Aware Run-Time System for High-Performance Computing, *in* 'ACM/IEEE SC2005: The International Conference on High-Performance Computing, Networking, and Storage', Seattle, Washington.

[17] Hsu, C., Feng, W. and Archuleta, J. S. [2005], Towards Efficient Supercomputing: A Quest for the Right Metric, *in* '1st IEEE Workshop on High-Performance, Power-Aware Computing (in conjunction with the 19th International Parallel & Distributed Processing Symposium)', Denver, Colorado.

[18] Huang, P., Peng, H., Lin, P. and Li, X. [2008], 'Macroeconomics based grid resource allocation', *Future Generation Computer Systems* **24**(7), 694 – 700.

[19] Khargharia, B., Hariri, S. and Yousif, M. S. [2008], 'Autonomic power and performance management for computing systems', *Cluster Computing* **11**(2), 167–181.

[20] Kim, J.-K., Shivle, S., Siegel, H. J., Maciejewski, A. A., Braun, T. D., Schneider, M., Tideman, S., Chitta, R., Dilmaghani, R. B., Joshi, R., Kaul, A., Sharma, A., Sripada, S., Vangari, P. and Yellampalli, S. S. [2003], Dynamic mapping in a heterogeneous environment with tasks having priorities and multiple deadlines, *in* 'IPDPS '03: Proceedings of the 17th International Symposium on Parallel and Distributed Processing', IEEE Computer Society, Washington, DC, USA, p. 98.1.

[21] Lynar, T. M., Herbert, R. D., Simon and Chivers, W. J. [2009], Impact of node rankings on outcomes of grid resource allocation, *in* 'GCA'09 Proceedings of the 2009 International Conference on Grid Computing and Applications'.

[22] Németh, Z. and Sunderam, V. [2002], *Computational Science ICCS 2002*, Vol. 2330/2002 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, chapter A Comparison of Conventional Distributed ComputingEnvironments and Computational Grids, pp. 729–738.

[23] Patel, C., Sharma, R., Bash, C. and Graupner, S. [2002], Energy aware grid: Global workload placement based on energy efficiency, Technical Report HPL-2002-329, Hewlett Packard, HP Laboratories Palo Alto.

[24] Pinheiro, E., Bianchini, R., Carrera, E. and Heath, T. [2001], Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems, *in* 'Proceedings of the Workshop on Compilers and Operating Systems for Low Power'.

[25] Rusu, C., Ferreira, A., Scordino, C. and Watson, A. [2006], Energy-efficient real-time heterogeneous server clusters, *in* 'RTAS '06: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium', IEEE Computer Society, Washington, DC, USA, pp. 418–428.

[26] Santiago, A. S., Yuste, A., Expósito, J. M., Galán, S. G., Marn, J. M. and Bruque, S. [2009], 'A dynamic-balanced scheduler for genetic algorithms for grid computing', *WSEAS Transactions on Computing* **8**(1), 11–20.

[27] Santos, M. F., Mathew, W., Kovacs, T. and Santos, H. [2009], 'A grid data mining architecture for learning classier systems', *WSEAS Transactions on Computing* **8**(5), 820–830.

[28] Schneck, P. B. [1990], 'Supercomputers', *Annual Reviews Computer Science* **4**, 13–36.

[29] Smith, W., Foster, I. and Taylor, V. [2004], 'Predicting application run times with historical information', *Journal of Parallel and Distributed Computing* **64**(9), 1007–1016.

[30] Spooner, D., Jarvis, S., Cao, J., Saini, S. and Nudd, G. [2003], 'Local grid scheduling techniques using performance prediction', *Computers and Digital Techniques, IEEE Proceedings* - **150**(2), 87–96.

[31] Springer, R., Lowenthal, D. K., Rountree, B. and Freeh, V. W. [2006], Minimizing execution time in MPI programs on an energy-constrained, power-scalable cluster, *in* 'PPoPP '06: Proceedings of the eleventh ACM SIGPLAN symposium on Principles and practice of parallel programming', ACM, New York, NY, USA, pp. 230–238.

[32] Tan, Z. and Gurd, J. [2007], Market-based grid resource allocation using a stable continuous double auction, *in* 'Grid Computing, 2007 8th IEEE/ACM International Conference on', pp. 283–290.

[33] Tarricone, L. and Esposito, A. [2004], *Grid computing for electromagnetics*, Artech House, Boston, MA, USA.

[34] Vahdat, A., Lebeck, A. and Ellis, C. S. [2000], Every joule is precious: the case for revisiting operating system design for energy efficiency, *in* 'EW 9: Proceedings of the 9th workshop on ACM SIGOPS European workshop', ACM, New York, NY, USA, pp. 31–36.

[35] Verma, A., Ahuja, P. and Neogi, A. [2008], Power-aware dynamic placement of HPC applications, *in* 'ICS '08: Proceedings of the 22nd annual international conference on Supercomputing', ACM, New York, NY, USA, pp. 175–184.

[36] Wolski, R., Plank, J. S., Brevik, J. and Bryan, T. [2001], 'Analyzing market-based resource allocation strategies for the computational grid', *International Journal of High Performance Computing Applications* **15**(3), 258–281.

[37] Yamamoto, H., Kawahara, K., Takine, T. and Oie, Y. [2006], 'Performance comparison of task allocation schemes depending upon resource availability in a grid computing environment', *IEICE Transactions on Information and Systems* **E89-D**(2).

[38] Zhao, J., Chen, D., Tian, Z. and Zhai, Z. [2006], A novel auction-based grid resource allocation algorithm, *in* 'Proceedings of the 1st International Symposium on Pervasive Computing and Applications', pp. 269–272.

[39] Zong, Z., Qin, X., Ruan, X., Bellam, K., Yang, Y. and Manzanares, A. [2007], A simulation framework for energy efficient data grids, *in* 'Simulation Conference, 2007 Winter', pp. 1417–1423.