# A Grid Resource Allocation Mechanism for Heterogeneous E-waste Computers

**Timothy M. Lynar, Ric D. Herbert, William J. Chivers, and Simon**

School of Design, Communication and Information Technology
University of Newcastle Australia,
Email: timothy.lynar@newcastle.edu.au

**Abstract**

While most grids and clusters are built from uniform nodes of new hardware, the notion of a grid of obsolete computers rescued from the scrapheap appeals in several ways such as lower environmental and financial cost. Existing resource allocation mechanisms, in assuming that nodes are equally capable, equally power efficient and equally reliable, fail to cater for the distinct features of an e-waste grid. This paper presents a resource allocation mechanism explicitly designed with the features of such an e-waste grid in mind, proposes an objective function to assess the mechanism, and tests the mechanism against two existing resource allocation mechanisms. In simulated tests of the three mechanisms, the new mechanism performs better than the other two, particularly under heavy load.

## 1 Introduction

Electronic waste (e-waste) is a vast and growing problem in today's society, and personal computers contribute significantly to this problem (ABS 2006). Every computer, when finished with, must be stored, dumped, recycled, or somehow re-used. Most are dumped (ABS 2006), 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. Supercomputers, grids and computer clusters provide even more power than ordinary desktop and laptop computers, but they too are subject to rapid obsolescence. Clusters and grids of obsolete computers can help to mitigate e-waste (Lynar et al. 2008). The efficient allocation of cluster and grid resources is a topic of much research (Zhao et al. 2006). Within this area, the allocation of e-waste resources is particularly challenging. This paper examines a novel resource allocation mechanism that is designed specifically for the allocation of e-waste computing resources.

Section 2 of this paper introduces a new resource allocation mechanism for e-waste resources. Section 3 defines a new objective function designed to assess the mechanism by testing it against two other market-based resource allocation mechanisms, and the results of this assessment are shown and discussed in section 4. Concluding remarks and future work are discussed in section 5.



## 2 The resource allocation mechanism

The Heterogeneous E-waste Resource allocation mechanism (HER) has been designed to specifically deal with the issues of allocation of e-waste computing resources. This section explains this novel resource allocation mechanism.

### 2.1 Why is a new mechanism needed?

There are many current grid and cluster resource allocation mechanisms (Zhao et al. 2006). However, no current mechanism appears to cater for the unique challenge of the creation of grids of heterogeneous e-waste components. These deficiencies could make any existing mechanism non-optimal. Many existing mechanisms are centralised (Chun & Culler 2000); require users to know how long their jobs will take to execute on unknown hardware (Germain-Renaud et al. 2008); fail to account for the heterogeneous nature of grid computing (Philippsen & Zenger 1997); assume that the computing resources are reliable; make the user wait until a batch auction is performed before any resources are allocated (Bubendorfer & Thomson 2006, Cao et al. 2003); and do not account for the varying power efficiency of the nodes utilised in the grid (Philippsen & Zenger 1997). These limitations are likely to result in a sub-optimal allocation when applied to a grid of e-waste resources. E-waste grids are by definition created from obsolete equipment that can be unreliable and that comes from a variety of sources (Hargrove et al. 2001). For this reason it is undesirable to assume that any individual node is reliable. E-waste resources are typically heterogeneous in age and performance capability; it is therefore impossible for users to predict at what speed their application will execute over the e-waste resource, especially when the resources are shared by a number of users with competing jobs. One important factor when dealing with e-waste resources is an understanding of the power consumption of the underlying resources. Typically the older the resource, the smaller its performance to power ratio. Some users and administrators might find it desirable to utilise resources that consume less power. As there appears to be no resource allocation mechanism that is tailored directly at heterogeneous e-waste resources, it was decided to create one. The HER allocation mechanism aims to account for the above issues.

### 2.2 Aims of the mechanism

The HER resource allocation mechanism has many aims including to:

- Allocate resources to maximise utilisation of the most able nodes;

- Facilitate the participation of less able nodes by allowing nodes to self-organise into desirable and usable resources;
- Allocate resources without requiring users to guess how long their application will take to execute;
- Allocate resources continuously, so that the user does not have to wait for a batch auction to take place;
- Make efficient use of available resources;
- Identify and exclude unreliable or broken resources;
- Take into account the power efficiency of nodes.

## 2.3 Description of the mechanism

This section describes in detail how the HER resource allocation mechanism operates. As a brief overview, the mechanism functions as follows:

- Periodically all nodes execute a small test application and record to a database the time it took to execute the application, which is used as a benchmark.
- Points are awarded to nodes based on the time it took to execute the test application. These points are overwritten each time the test application is executed; but if for some reason this fails to happen, the points will expire after specified times.
- When a task is submitted to the grid it is immediately allocated, through a preprocessed auction, if there are sufficient available resources.
- The task has an ask amount, set by the user, and based on the quantity of processing resources the user desires to allocate to their task.
- Each node bids for tasks based on its endowment of points, bidding a system-defined percentage of its available points.
- The job is awarded to the node that bids the highest amount of points for the task, and in winning the bid the node spends the points that were asked for the task.
- Nodes are able to self-organise into virtual clusters. Virtual clusters enable nodes to pool their resources, to enable them to compete more effectively for jobs against more powerful nodes.

## 2.4 Features

The HER resource allocation mechanism is designed to run over a grid that consists of a number of individual computing resources housed in distributed locations. This mechanism could be used for a number of different types of applications. However the authors had in mind Single Process Multiple Data (SPMD) applications, that could utilise a large number of nodes. The features of the grid resource allocation mechanism are now explained in greater detail and their benefits are highlighted.

### 2.4.1 Maximising utilisation of the most able nodes

The resource allocation mechanism maximises the use of the most able nodes by assigning points to nodes based on their performance in executing a test application. The resource allocation mechanism uses a database to store the points received by each node. More points are give to the nodes that execute the test application fastest. A node's points are allocated with an expiry time, and as they expire fresh allocations are given, the new points overwriting the older points. This ensures that the most able nodes have the most points. Nodes then bid for tasks, using a predetermined percentage of their available points. This percentage is determined by a setting in the resource allocation mechanism, requiring no input from the user and no strategies on the part of the node. The task is awarded to the node that makes the highest bid, and is awarded at the asked price not the bid price. The asked points are then removed from the node, limiting the node's ability to compete in future auctions while processing the task. When the job is complete, completion points are awarded to the node, restoring its ability to bid fairly for jobs by giving back the points that it lost by winning the bid. This is the only time the resource allocation mechanism will add points to an existing score. This auctioning system dynamically ensures that at any given time a new task will be allocated to the most able resource.

### 2.4.2 Facilitating participation of less able nodes

Less able nodes are able to participate through self-organising into virtual clusters, which enable groups of nodes to bid as one with their combined resources, instead of as individuals. Periodically each node makes a decision as to whether or not to participate in a virtual cluster, as explained in section 2.4.7. The use of virtual clusters enables the less able nodes to pool their resources to create usable resources and to compete for jobs against the more able individual resources.

### 2.4.3 Removing guesswork

Some resource allocation mechanisms require users to guess the runtime of their application on unknown hardware (Germain-Renaud et al. 2008). This mechanism does not require the user to input any estimates. The only user intervention required is for the user to indicate the amount of processing power they would like for their tasks, and to submit their tasks to the system. The amount of processing power is the asking amount in points, which is of course limited by the maximum number of available points in the system. If a user asks above this amount their job may never be executed. Users who ask for a high allocation may have to wait a considerable time until there are enough available resources to execute their task; this is designed to encourage users to moderate their requests. Greedy users may eventually have their task executed, however they will have to wait for those resources to become available.

### 2.4.4 Working as a continuous system

This system is designed to work in a continuous fashion, with pertinent resource allocation mechanisms such as continuous double auctions. However, continuous double auctions can exhibit undesirable characteristics for resource allocation because they cannot guarantee that the fastest available resource will win the auction (Dash et al. 2007). Batch auctions can

guarantee that the fastest available resource will win, however they must be run in batches which will result in a significantly delayed start for execution of the resource (Chun & Culler 2000). The auction employed in this research suffers from neither of these disadvantages. In the auction employed, resources continuously update their bid to a database every time their status changes. This enables the fastest available resource to be selected for executing any given task, without the need to delay the execution while a potentially lengthy auction is conducted. Of course the constant updating of bids by resources to the database results in an increase in network overhead by the resource allocation mechanism. This places a limit on the scalability of this mechanism, which has not yet been tested.

#### 2.4.5 Accounting for the heterogeneous power of resources

This resource allocation mechanism accounts for the heterogeneous nature of resources through a point system. Resources are given points based on their speed at executing a test task. Nodes then use these points to bid for jobs. This accounts for the heterogeneous nature of the individual nodes by assigning the nodes differing allocations of points to reflect their differing performance.

To account for a node's differing power-performance ratio a multiplier is included in the formula to calculate a node's points. A higher multiplier is given to nodes that have a reduced power consumption. The node's multiplier is simply its calculated power to performance ratio, as explained in section 2.5.2.

#### 2.4.6 Removing unreliable nodes

A simple method is utilised to detect nodes that are faulty or become faulty. Nodes that are faulty will not finish executing the test task and therefore will not receive points to bid with, and thus will not be able to acquire jobs to execute. This method theoretically offers a reasonable level of detection of faulty nodes, however it is still possible that a node will complete a task but return a faulty result, possibly as an undesired result of over-clocking the CPU, or of some other hardware or software fault (Anderson 2004).

#### 2.4.7 Self organisation

Self organisation allows nodes to pool their resources to form a virtual grid. Each node periodically compares its points and performance with those of other nodes. Nodes also examine the uncleared asks. If nodes are performing poorly, or they are idle and there are unclaimed tasks, then the nodes will decide to join a collective group called a virtual cluster. The bids of the individual nodes are summed by the system to produce a bid for the virtual cluster. Nodes in a virtual cluster will contribute to the processing of the jobs in the virtual cluster, and will not bid on their own while they are members of a virtual cluster. The virtual cluster will last only until its expiry time, which is set when it is created. Virtual clusters have a limitation. The applications executed can only be executed over a finite number of nodes. The administrator may set an upper limit on the number of nodes that can join a virtual cluster. Obviously applications that only require a small number of nodes will be disadvantaged by utilising virtual clusters.

### 2.5 Functions

This section describes in detail the individual functions of the proposed resource allocation mechanism: the performance test, the process of assigning points to nodes, the auction, and the self-organisation test.

#### 2.5.1 Performance test

The performance test is run periodically on all nodes in the grid. The test consists of a small and simple executable that returns the time that a task took to execute. This test is run across all nodes at regular intervals. The application utilised for this test should be modified by the system administrator to be indicative of the type of application that is to run most frequently over the grid. This will result in a more accurate indication of the ability of nodes to process a particular task. It is envisaged that this application will be small, so as to minimise overhead for the system.

#### 2.5.2 Assignment of points

Points are assigned to nodes by the resource allocator after the test task is executed. The points credited to the node are calculated using the performance index. The performance index $P_i$ of each node $i$ includes the test performance and energy performance of the node. Define the energy performance as:

$$E_i = W_i/3600 \tag{1}$$

where:
$W_i$ is the node's power consumption in watts per hour.

Define test performance as: The reciprocal of the time taken on the benchmark task.

$$C_i = 1/T_i \tag{2}$$

where:
$T_i$ is the number of seconds the node took to complete the benchmark task.

Then the performance value for each node $i$ is given by:

$$A_i = \frac{C_i}{E_i} \tag{3}$$

We use a relative performance index for node $i$ which is given by:

$$P_i = \frac{A_i}{A_{max}} \tag{4}$$

where:
The subscript $max$ refers to the node with the maximum value.
Hence $0 \leq P_i \leq 1$. So that 1 is the value that is given to the node with the highest performance value.

Points are awarded to each node and are given an expiry time. When the expiry time is reached the node re-runs the test application. On completion of the test application, the node's points are reissued by the mechanism. The power consumption data of nodes $W_i$ is measured and stored in the database off-line.

#### 2.5.3 Auction

A simple modified double auction is employed. In this auction, nodes bid for asks and users set the asking price of the ask. Nodes show no preference for any particular ask, nor do they employ any strategies. Nodes bid a system set percentage of their available

points for each ask. This auction is able to guarantee the fastest available resource for any given ask, and also operate in a continuous fashion. This is achieved by nodes placing persistent bids in a database, which are updated as the node's status changes. When an ask is made the largest bid is accepted, unless that bid is less than the the ask. Clearing then occurs and the points asked are deducted from the points of the winning node, inhibiting that node's ability to win future tasks until the task is completed and the points are returned. If the ask price is too high, the user may have to lower their ask amount or wait until an acceptable supply of resources become available to execute their task at the desired speed. The algorithm utilised in the auction is presented in algorithm 1.

**Algorithm 1** The algorithm of the auction

```
loop
  bids are periodically entered into the database
  by resources (nodes and virtual clusters)
  as asks are submitted to the database
  for all asks (in order of submission) do
    the highest current bid is retrieved from the
    database
    if the value of the highest bid is ≥ the value
    of the ask then
      the ask is awarded to the bid and the ask's
      tasks are assigned to the resource that cre-
      ated the bid
      the bid and the ask are removed from the
      database
    else
      the ask is not cleared
    end if
  end for
end loop
```

#### 2.5.4 Self-organisation test

A self organisation test is performed by each node on a periodic basis. The purpose of this function is two-fold. First it enables less endowed resources to take active part in the system, and second it permits the allocation of a greater number of resources to tasks than would be possible using individual nodes. The resource allocation mechanism periodically asks each node to indicate if the node wishes to be part of a virtual cluster. The logic behind a node's decision is elucidated by algorithm 2.

**Algorithm 2** The algorithm for self-organisation

```
A node will decide to join a virtual cluster if the
following four conditions are all true:
if the node is not executing a task &
the node is not already in a virtual cluster &
there are uncleared asks in the system &
the maximum ask in the system is greater than a
bid that this node could create then
  the node returns the value of yes to joining a
  virtual cluster
end if
```

If the node returns yes to joining a virtual cluster, the resource allocation mechanism adds the node to a virtual cluster. This process simply results in setting a flag in the node to indicate that it is part of a virtual cluster, and removing the node's current bid from the system and adding it to the cluster's bid. While a node is part of a virtual cluster the resource allocator will not ask it for individual bids.

#### 2.5.5 Virtual clusters

Virtual clusters are created by the resource allocation mechanism if one or more nodes decide to join a virtual cluster. Many nodes can join a virtual cluster, but a virtual cluster will accept new participants only if its total point allocation is less than the largest ask in the system. If a virtual cluster is full and nodes wish to join, those nodes will join a new virtual cluster. Virtual clusters are created with an expiry time. Upon entry into a virtual cluster a node's bid is removed from the database, while the nodes that make up the virtual cluster have their bids summed to form the bid of the virtual cluster. Upon exit from the virtual cluster, nodes once again bid individually. When an ask is assigned to a virtual cluster the individual tasks within the ask are assigned to the individual nodes of the virtual cluster.

## 3 Assessing the mechanism

An objective function has been designed in order to test whether the resource allocation mechanism achieves the aims for which it was developed. The mechanism has been tested against two existing mechanisms to determine how well it performs.

### 3.1 Objective function

In order to test whether the mechanism developed achieves the above aims an objective function has been created. Some of the goals of the mechanism are implicit in its design and so will not be presented as part of the objective function. Four remaining primary goals have been selected to create an objective function by which the mechanism will be tested:

**Maximise utilisation of the most able nodes** this goal is measured by $I$, the percentage of idle time experienced by the most powerful 10% of nodes.

**Facilitate the participation of less able nodes** this is measured by $P$, the percentage of unique nodes that never process or assist in the processing of tasks.

**Efficiently utilise available resources** this is measured by $T$, the percentage of assigned tasks that remain incomplete.

**Allocate resources in a continuous fashion** this is measured by $A$, the percentage of asks that remain unassigned.

**Exclude unreliable or broken resources** is measured by $B$, the percentage of tasks that were assigned to a broken resource.

We define the instantaneous objective function as the sum of the above components at time step $t$. Then:

$$J_t = I_t + P_t + A_t + T_t + B_t \tag{5}$$

The aim of this mechanism is to maximise the throughput of jobs, so the objective function $\mathcal{J}$ is:

$$\mathcal{J} = \sum_{t=1}^{M} J_t \tag{6}$$

Where $M$ is the final time in the time horizon.
The best mechanism is therefore the mechanism that minimises the objective function $\mathcal{J}$.

### 3.2 Simulating alternative models

Two alternative resource allocation mechanisms were produced for comparison purposes: a batch resource allocation mechanism and a continuous resource allocation mechanism. The batch mechanism executes an auction periodically. The users submit asks to the system and the nodes submit bids. The bids and asks are sorted and the highest ask is assigned to the highest bid that is greater than or equal to it in amount. This continues until no more asks can be satisfied by bids available. Like all simulated mechanisms in this paper, the node's bids are based on a fixed percentage of their processing endowment.

The second alternative mechanism is a continuous double auction. In this mechanism bids are constantly updated by nodes and asks are constantly submitted to the system by users. As soon as an ask is submitted to the system it is cleared by the first bid that is found to be equal to or greater than the ask in amount.

It is believed that the two above alternative mechanisms are representative of other available market-based resource allocation mechanisms.

### 3.3 Methodology

To test the mechanism, a model of the mechanism was created. The model simulated the grid environment and workload. The simulation allows clear comparisons of allocation mechanisms because it eliminates all other variables and allows researchers to evaluate one mechanism against another. Three alternative resource allocation mechanisms have been created and simulated: a batch auction mechanism, a continuous double auction (CDA) mechanism and the HER resource allocation mechanism. The same workloads were simulated across all three allocation mechanisms and results were recorded and compared. The results examine the capability of the resource allocation mechanism to achieve the objectives embodied in the objective function.

The workload simulated over the grid was identical for all three resource allocation mechanisms. A total of 300 asks were submitted with a total of 52100 tasks. The asks were submitted over a period of 300 time steps in 3 groups of 100 time steps. The first group of 100 asks was submitted through time steps 0 to 99. In this group each ask was for a value of 500 and had one task. The next group of asks occurred through time steps 2000 to 2099 and contained 100 asks for a value of 2000 with 20 tasks each. The next group of asks occurred through time steps 5000 to 5099 and contained 100 asks of a value of 50000 with 500 tasks each.

The nodes present in the system were the same for all three resource allocation mechanisms. Each node had a pseudo-random power capacity of between 400 and 2400.

We propose the following hypothesis: The HER resource allocation mechanism when run with the same workload as the batch and CDA resource allocation mechanisms will achieve a lower value of the objective function $J$.

## 4 Results

The simulation produced raw data relating to the components of the objective function for each resource allocation mechanism. Comparative results were also produced. These results show the objective function over time and the cumulative objective function over time.

### 4.1 Efficiency at achieving objectives

In order to assess the ability of each mechanism to achieve the objective function, each component of the objective function will first be examined.

#### 4.1.1 Minimising idle time

The amount of idle time of the fastest 10% of nodes has been recorded as $I$. The idle time of the three resource allocation mechanisms is displayed in figure 1. With the above workload all mechanisms resulted in a high percentage of idle time for the fastest nodes. The batch model appeared to have performed the best, with a mean of 93.97% and a minimum of 20%. The HER also performed well with a mean of 94.26% which is only marginally higher than the batch mechanism and a minimum of 15.56% which is even less than that of the batch mechanism. The CDA performed the worst of all mechanism with 100% idle time as its average and minimum: its most able nodes did not perform any computation. However, as figure 1 shows, the difference between the batch and HER mechanisms is slight.

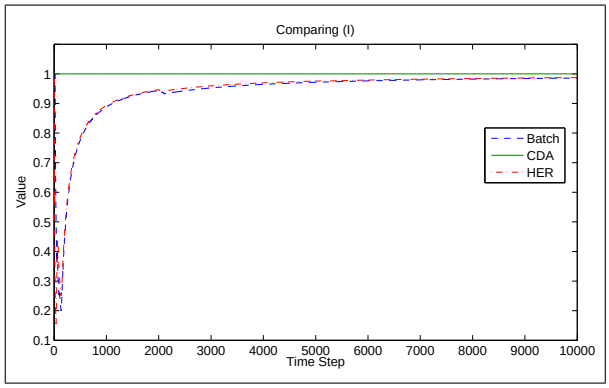


Figure 1: Proportion of time spent idle by the most able 10% of nodes

#### 4.1.2 Maximising utilisation of all resources

The utilisation of resources was measured with $P$, which measures the percentage of resources that never process a task. The minimum value for $P$ is therefore important because it indicates what percentage of nodes were never utilised throughout the simulation. The average value for $P$ is also important because it gives an indication of when the nodes were utilised. Figure 2 is a graph of the value of $P$ for the three mechanisms over time. The batch mechanism performed poorly on this metric, with an average of 0.4326 and a minimum of 0.43 indicating that 43% of nodes never processed a task. The performance of the batch mechanism was hampered by its ability to guarantee that it could select the fastest available nodes. The batch mechanism consistently utilised fewer nodes than the other mechanisms. The CDA mechanism performed better than the batch mechanism, with an average of 0.3827 and a minimum of 0.38 indicating that 38% percent of nodes never processed a task. The HER mechanism had the highest percentage of nodes utilised in processing of the three mechanisms. The HER mechanism achieved an average $P$ of 0.186 and a minimum of 0, indicating that in the HER all nodes participated in the processing of tasks.

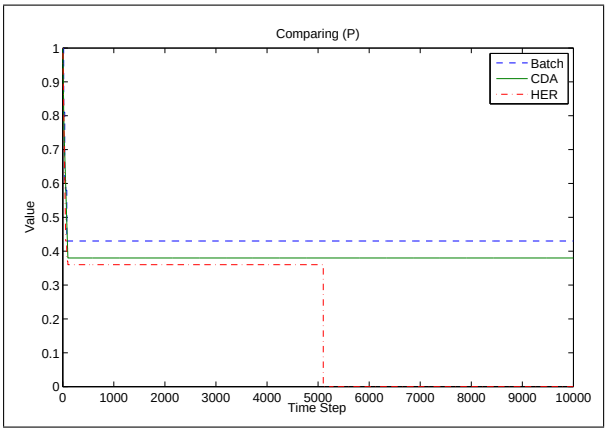


Figure 2: The proportion of nodes that have not processed a task

#### 4.1.3 Efficiently utilise available resources

$T$ is a measure of uncompleted but assigned tasks, and is thus a good indication of the efficiency of a mechanism in getting tasks processed. The higher the value of $T$, the slower the execution of the tasks. The CDA mechanism performed the worst in this metric, with an average of 0.0195. The HER performed marginally better with an average of 0.0114, and the batch mechanism performed the best with an average of 0.0087. It is important to note that the number of tasks processed is directly related to the number of asks allocated. If a mechanism does not allocate asks then it cannot process tasks and will have a low $T$ value.

#### 4.1.4 Allocate resources in a continuous fashion

$A$ is a measure of the percentage of asks that are unassigned at any point in time, and is used to measure the efficiency of the mechanism in allocation of asks. The higher the value of $A$, the more unallocated asks. The batch mechanism performed the worst at this metric, with a maximum of 100% of asks being unallocated and an average of 16.91% of asks being unallocated. The CDA performed marginally better than the batch mechanism with a maximum of 33.3% of asks being unallocated and an average of 16.5% of asks remaining unallocated. The HER performed the best on this metric with a maximum the same as the CDA at 33.3 percent but with an average of only 1.84% of asks being unallocated. This is an indication of the mechanism's ability to quickly match a suitable resource to a suitable ask.

#### 4.1.5 Percentage of tasks assigned to broken resources

The number of times a broken or faulty resource is given a task to execute is measured by $B$. However in this simulation no broken resources were simulated as the CDA and batch auction models were not designed with any features that would enable them to deal with broken resources. All models therefore received a score of 0 for $B$.

#### 4.1.6 Overall efficiency

Two measures of the overall efficiency of the mechanism were taken. A measure of the performance of the three mechanisms at meeting the objective function over time (Figure 3) and the integrated sum of this value over time (Figure 4).

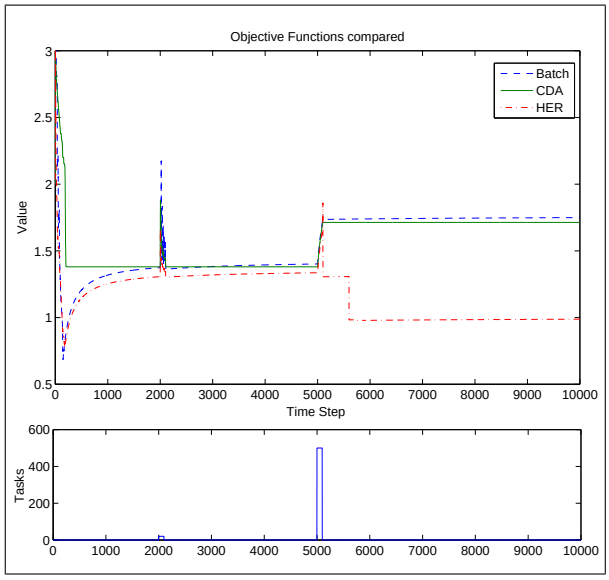


Figure 3: The instantaneous objective function of each resource allocation mechanism

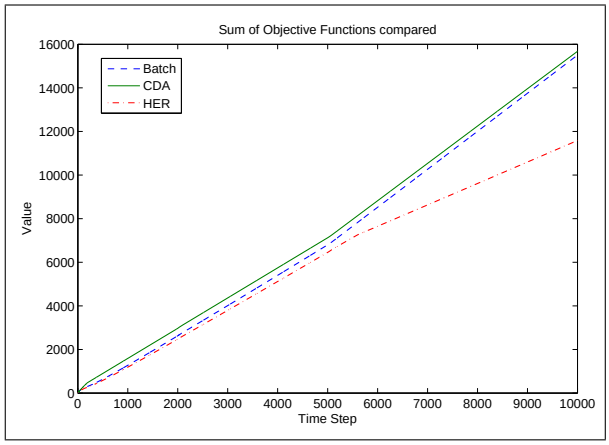


Figure 4: The objective function

Of the three mechanisms, the CDA mechanism was the worst at minimising the objective function $J$. CDA had the highest total integrated value, 15676. The batch mechanism performed next best, with an integrated value of 15503. The HER mechanism performed best, with an integrated value of 11586.

The results also show, when viewed with task submission information (lower part of Figure 3) that the point at which the HER mechanism substantially moved away from the other two mechanisms was when under significant load of new tasks.

## 5 Concluding remarks

This paper has presented a new resource allocation mechanism explicitly designed to work with a grid of e-waste computers, a grid whose components are likely to be non-uniform and possibly less reliable than new hardware. An objective function has been established to test the performance of the mechanism in those respects where an e-waste grid differs from a more conventional grid. In simulated tests, the new mechanism performed extremely well against a batch allocation system and the CDA allocation system, particularly when under heavy load.

Although the measurements reported in this paper were of simulations, we have built a grid of e-waste

computers, and will continue to explore the uses of this grid, using the new resource allocation mechanism.

## References


ABS (2006), Australia's environment: Issues and trends 2006, Technical Report cat no. 4613.0, The Australian Bureau of Statistics (ABS), Canberra.

Anderson, D. P. (2004), Boinc: a system for public-resource computing and storage, *in* 'Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on', pp. 4–10.

Bubendorfer, K. & Thomson, W. (2006), Resource management using untrusted auctioneers in a grid economy, *in* 'Second IEEE International Conference on e-Science and Grid Computing. e-Science '06.', pp. 74–74.

Cao, J., Spooner, D., Jarvis, S., Saini, S. & Nudd, G. (2003), Agent-based grid load balancing using performance-driven task scheduling, *in* 'Parallel and Distributed Processing Symposium, 2003. Proceedings. International', p. 10 pp.

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

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

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

Hargrove, W. W., Hoffman, F. M., Sterling, T. & Kay, C. (2001), 'The do-it-yourself supercomputer', *Scientific American* **285**(2), 72 – 80.

Lynar, T. M., Herbert, R. D. & Chivers, W. J. (2008), Implementing an agent based auction model on a cluster of re-used workstations, *in* 'Proceedings of the 5th International Conference on Information Technology and Applications (ICITA 2008)', Cairns, Queensland, pp. 642–646.

Philippsen, M. & Zenger, M. (1997), 'Javaparty - transparent remote objects in Java', *Concurrency: Practice and Experience* **9**(11), 1225–1242.

Zhao, J., Chen, D., Tian, Z. & Zhai, Z. (2006), A novel auction-based grid resource allocation algrithm, *in* 'Pervasive Computing and Applications, 2006 1st International Symposium on', pp. 269–272.