034115

PHOSPHORUS

Lambda User Controlled Infrastructure for European Research

Integrated Project

Strategic objective:
Research Networking Testbeds

Deliverable reference number <D.5.5>

Recommendations for Control Plane Design

Due date of deliverable: 2009-06-30
Actual submission date: 2009-06-30
Document code: <Phosphorus-WP5-D5.5>

Start date of project:
October 1, 2006

Duration:
30 Months

Organisation name of lead contractor for this deliverable:
IBBT

Revision 2

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	
PP	Restricted to other programme participants (including the Commission	
RE	Restricted to a group specified by the consortium (including the Commission	
CO	Confidential, only for members of the consortium (including the Commission Services)	

<Recommendations for Control Plane Design - Addendum>

Abstract

This deliverable serves as an addendum to Deliverable D5.5 "Recommendations for Control Plane Design". More specifically, it aims to provide deeper insight in the scalability of the architectures that originated from WP1 "Network Resource Provisioning Systems (NRPS) for Grid Network Services (GNS)" and WP2 "Enhancements to the GMPLS Control Plane for Grid Network Services". First, we proposed several techniques for aggregating the resource information of the sites in hierarchical Grid domains and performing task scheduling using this information. This study is highly relevant for the flow of information states in a G2MPLS protocol stack, and as such we investigated its scalability to measure aggregation efficiency. Also, we described our simulation environment that was created to study scalability issues of the Harmony service plane solution. More specifically, we implemented a simulator that incorporates the different architectural approaches: centralized, hierarchical and distributed.

Project:	Phosphorus
Deliverable Number:	<D.5.5>
Date of Issue:	30/06/2009
EC Contract No.:	034115
Document Code:	<Phosphorus-WP5-D5.5>

Table of Contents

0 Executive Summary

This deliverable serves as an addendum to Deliverable D5.5 “Recommendations for Control Plane Design”. More specifically, it aims to provide deeper insight in the scalability of the architectures that originated from WP1 “Network Resource Provisioning Systems (NRPS) for Grid Network Services (GNS)” and WP2 “Enhancements to the GMPLS Control Plane for Grid Network Services”.

First, we proposed several techniques for aggregating the resource information of the sites in hierarchical Grid domains and performing task scheduling using this information. This study is highly relevant for the flow of information states in a G2MPLS protocol stack, and as such we investigated its scalability to measure aggregation efficiency. In particular, we studied the trade-off between the amount of information exchanged (and used by the scheduler) and the scheduling efficiency. We also introduced domination relations and showed that they can increase the quality of the aggregated information. Finally, we observed that the uniformity of the sites’ characteristics significantly affects the task delays that can be achieved.

Also, we described our simulation environment that was created to study scalability issues of the Harmony service plane solution. More specifically, we implemented a simulator that incorporates the different architectural approaches: centralized, hierarchical and distributed. We demonstrated some results related to the average process duration of a connectivity request, and showed the existence of large variations in the behavior of the various architectures. More specifically, we demonstrated the poor scalability of the centralized approach, and the very high scalability performance of the distributed approach. Likewise, we showed that the hierarchical approach is highly dependent on the structure of the Harmony service plane tree. This implies that further research will be focused on optimizing the location and interconnection of the service nodes.

1 Resource Information Aggregation in Hierarchical Grid Networks

1.1 Introduction

In Grid Networks, a scheduler receives requests for the use of resources and assigns the tasks so as to optimize some objective function. The scheduler makes its decisions based on information about the resources, such as their computational or storage capacity, their availability, etc, which are usually collected by information services called monitoring systems [1,3].

We propose a number of information aggregation techniques for summarizing the resource-related information, used by the task scheduler. With the emergence of a number of Grid services (e.g., Amazon EC2 and S3, Microsoft Azure) it will soon become necessary of summarizing their resource related information with a unified manner. This way it will be possible for a task scheduler to use efficiently the resources of the one or the other service, without at the same time being necessary for the corresponding service to publish in detail its resources characteristics. Moreover the proposed techniques can reduce the amount of resource-related information that is transferred, stored and processed. We expect that in the near future, as more resources of various types (clusters, PCs, mobile phones, etc) participate in Grids, the amount of information that will have to be transferred and processed will be quite large. This can lead to network congestion and overuse of the resources.

We assume that resources/sites are grouped into hierarchical domains and the information related to the sites in each domain, is aggregated before being sent to a higher level. Each site is assigned a vector of cost parameters that records its computation or storage capacity, its availability, and other parameters. Next, the cost vectors of the sites belonging to a given domain are aggregated into a single cost vector for the entire domain, by performing appropriate associative operations to the cost parameters. We also introduce so-called domination relations that reduce the number of vectors aggregated and stored. When a task request arrives, the scheduler selects the domain where the task will be executed, by applying an optimization function to the collected and aggregated cost vectors.

A drawback of information aggregation is that the efficiency of a scheduler using such information may be negatively affected. This introduces a trade-off between the amount of information exchanged (and used by the scheduler) and the efficiency of the scheduling decisions. We propose information aggregation schemes that produce aggregated information of different quality, improving or deteriorating the scheduling decisions. These techniques are presented in a general way, permitting their combination and the creation of new ones. We perform a large number of experiments to evaluate the proposed aggregation techniques. We use as a metric the Stretch Factor (*SF*), defined as the ratio of the task delay when the task is scheduled using complete resource information over the task delay when an aggregation scheme is used. Our simulation results show that the proposed schemes achieve large information reduction, while maintaining good scheduling quality, in comparison to the case where no aggregation is performed. The amount of information is measured with the number of cost vectors used by the task scheduler in order to make its decisions.

1.2 Previous Work

In Grid Networks, information collection is performed by the monitoring systems. In [1] a number of monitoring systems are presented and categorized. These systems are often organized in a hierarchical structure, which is consistent with the structure of the Grids and the Data Networks. In these networks, sites are organized in domains that build up to hierarchical structures. Hierarchical routing plays a major role in Data Networks, as a way to minimize the routing tables required for the very large topologies encountered in Internet’s infrastructure. [4] is one of the first works investigating hierarchical routing.

A central issue in hierarchical routing is topology information aggregation [5,6], which tries to summarize and compress the topology information advertised at higher levels. In order to perform routing efficiently, the aggregated information should adequately represent the topology and the other characteristics of the network, such as the delay and bandwidth. In [7] a topology aggregation scheme subject to multi-criteria (delay and bandwidth) constraints is presented.

Generally, most scheduling algorithms presented to date [8-12] make their decisions using exact resource information, which may, however, be outdated by the time it is used due network delays. On the other hand, by using resource information aggregation techniques the scheduler makes its decisions based on aggregated resource information and a trade-off is introduced between the amount of information exchanged (and used by the scheduler) and the quality of these decisions.

Most previous works [5-7] consider the aggregation of network-related information and the effects of this aggregation on the routing process. Also, the idea of information aggregation has appeared in P2P [17] and in sensor networks [18]. To the best of our knowledge this is the first time grid information aggregation techniques are investigated and evaluated. Scheduling using incomplete information has also been considered in [15], where, a technique is presented for monitoring large Grid Networks that selects a statistically valid sample and measures the behavior of the sample members, instead of monitoring each individual system.

1.3 Problem Formulation

We consider a Grid consisting of N sites, partitioned according to a hierarchical structure in a total of L domains D_j, $j=1,2,\ldots,L$. Each site i, $i=1,2,\ldots,N$, has computational and storage capacity C_i and S_i, respectively, and belongs to one of the L domains. Site i publishes its resource information as a vector V_i, that may contain various parameters:

$$V_i = (C_i, S_i, \ldots).$$

These vectors are collected per domain D_j and are published to a higher level of the hierarchy, in the form of a matrix of vectors:

$$M_j = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_{|D_j|} \end{bmatrix} = \begin{bmatrix} (C_1, S_1, \ldots) \\ (C_2, S_2, \ldots) \\ \vdots \\ (C_{|D_j|}, S_{|D_j|}, \ldots) \end{bmatrix},$$

where $|\cdot|$ denotes the cardinality of a set and 1,2, …,$|D_j|$ are the sites contained in domain D_j. By performing appropriate operations on the parameters of the information vectors contained in the information matrix M_j, M_j is transformed into the *aggregated information matrix* $\hat{M}_j$.

The Grid scheduling problem is usually viewed as a hierarchical problem that has two levels. At the higher level a central scheduler decides the domain D_j a task will be assigned to, and at the lower level a domain scheduler DS_j, decides the exact site in the domain where the task will be executed (**Figure 1.1**). The information collection and aggregation is performed, similarly, by a two level monitoring system, consisting of a central monitor *CM* and the domain monitors DM_j, j=1,2,…,L. The presented work can also apply in the case of a multi-level Grid system with distributed scheduling and monitoring entities. A user located at some site generates tasks T_m, m=1,2,…, with computational workload W_m.

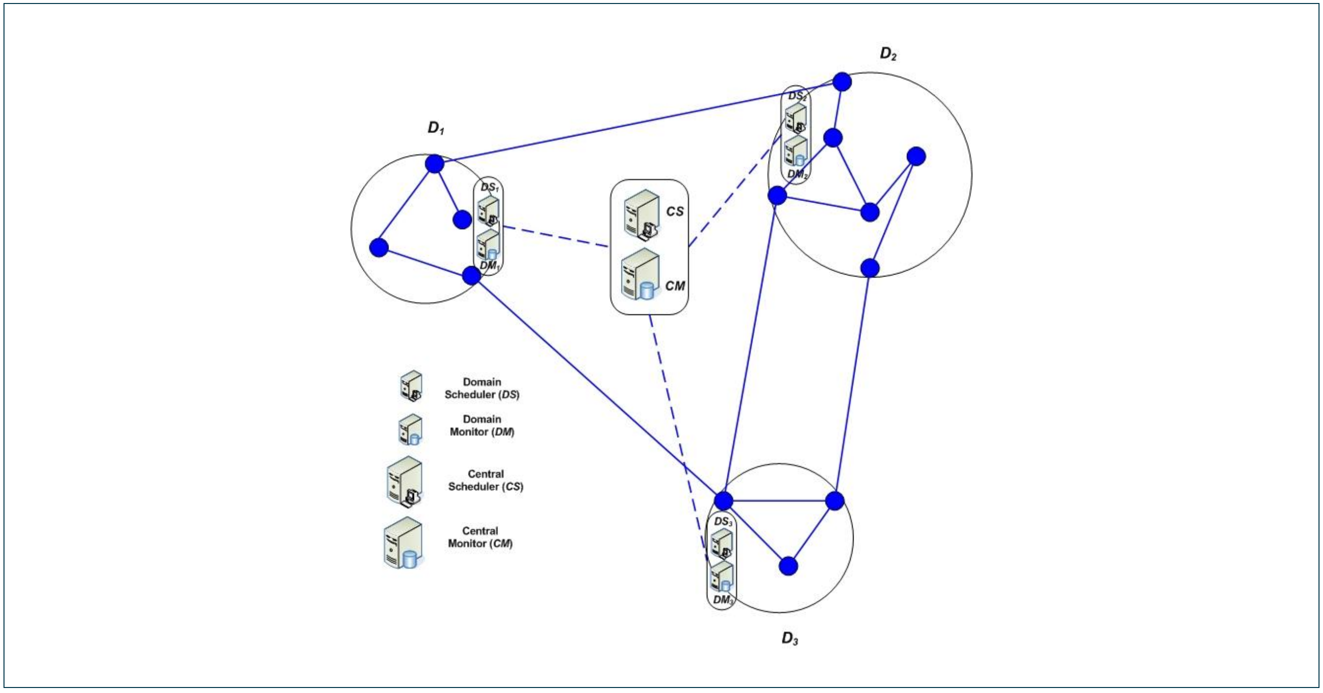


Figure 1.1: A two-level monitoring and scheduling system. Each domain D_j has a domain scheduler DS_j and domain monitor DM_j. There is also a central scheduler *CS* and a central monitor *CM*.

1.4 Information Aggregation

1.4.1 Proposed Scheme

The proposed scheme consists of an information aggregation algorithm together with a task scheduling algorithm that uses this information. In Algorithm 1 we present the pseudocode for the information collection and aggregation scheme.

Algorithm 1 Resource Information Collection and Aggregation

```
1. Each site $i$, $i$=1,2,…,$N$, belonging to some
   domain $D_j$ periodically or reactively (when
   information changes) publishes its
   information vector $V_i$ to the domain monitor
   $DM_j$.

2. Each domain monitor $DM_j$, $j$=1,2,…,$L$, puts
   together these vectors to form the
```

Project:	Phosphorus
Deliverable Number:	<D.5.5>
Date of Issue:	30/06/2009
EC Contract No.:	034115
Document Code:	<Phosphorus-WP5-D5.5>

information matrix M_j.

3. Domain monitor DM_j, j=1,2,…,L, periodically or reactively (when information changes) computes its aggregated information matrix $\hat{M}_j$ and publishes it to the central monitor *CM*.

4. The *CM* collects the aggregated information matrices.

In Algorithm 2 we present the scheduling scheme that uses the aggregated information.

Algorithm 2 Task Scheduling

1. Upon the arrival of a task T_m,, the central scheduler *CS* looks at the domain matrices provided by the central monitor *CM*.

2. The central scheduler *CS* applies an optimization function to the vectors contained in the domain matrices and selects the information vector *V* that produces the largest value.

3. The *CS* assigns the task T_m to the domain D_j, where the vector *V* originated from, and forwards the task to the domain scheduler DS_j.

4. The domain scheduler DS_j receives the task and selects the exact site the task will be scheduled on, using exact resource information.

1.4.2 Information Parameters and Aggregation Operators

We present the resource information parameters of interest in this work, and the operators used for their aggregation. For every parameter, different operators can also be used (e.g. min, max, sum, average), depending on the needs of the applications and the scheduling algorithms used. Next, we list some of these parameters and operators, giving a brief explanation of their usage:

- The computational capacities C_i of the sites, measured in Millions Instructions per Second (MIPS), in a domain D_j can be aggregated by performing a minimum representative operation or an additive operation:

  - $\hat{C}_j = \min_{i \in D_j} C_i$ or $\hat{C}_j = \sum_{i \in D_j} C_i$.

  Using the minimum representative operator we obtain the minimum capacity of any site in the domain D_j, which would be useful for conservative task scheduling. Using the additive operator we obtain the total computational capacity in the domain, which would be useful for scheduling when a task's workload is divisible, and can be assigned to different resources simultaneously.

- The storage capacities S_i of the sites, measured in MB, in a domain D_j can be aggregated as following:

  - $\hat{S}_j = \sum_{i \in D_j} S_i$ or $\hat{S}_j = \max_{i \in D_j} S_i$.

  The first definition is useful when the data of a task can be stored in a distributed way across the domain, while the second when the data have to be stored at a single site.

- The number of tasks N_i assigned to the sites can be aggregated over a domain D_j as following:

  - $\hat{N}_j = \sum_{i \in D_j} N_i$.

- The estimated time FT_i in the future at which a computational resource belonging to site *i* will be freed can be aggregated over all sites of domain D_j by using a minimum representative operator:

  - $\hat{FT}_j = \min_{i \in D_j} FT_i$.

  Using this aggregated value the scheduler will know the earliest time at which some site in domain D_j will be free to execute a new task.

- The *Start times* (*ST*) and *End times* (*ET*) of the tasks assigned to sites of a domain can be aggregated by finding the time periods where all sites in the domain are executing a task. This means that during the remaining time periods, there is at least one resource that is idle and available for scheduling new tasks. This information may be useful for schedulers performing timed and advance resource reservations [13,14].

1.4.3 Aggregation Schemes

1.4.3.1 *Single Point Aggregation Scheme*

In the *single point* aggregation scheme the information vectors of the sites in each domain are aggregated into a single information vector by applying various associative operators. We show an example of the application of the *single point* aggregation technique, where the size of the information matrix M_j is reduced from $|D_j|$ to 1:

$$M_j = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_8 \end{bmatrix} = \begin{bmatrix} (C_1, S_1, ...) \\ (C_2, S_2, ...) \\ \vdots \\ (C_8, S_8, ...) \end{bmatrix} \Rightarrow \hat{M}_j = \begin{bmatrix} \hat{V} \end{bmatrix} = \begin{bmatrix} (\hat{C}, \hat{S}, ...) \end{bmatrix}$$

The information transferred to the higher levels is greatly reduced using this aggregation technique, at the cost, however, of degraded quality of the aggregated information.

1.4.3.2 *Intra-Domain Clustering Aggregation Scheme*

In the *intra-domain clustering* aggregation technique, the sites of each domain D_j j=1,2,…,L, are partitioned into $h_j \le |D_j|$ *intra-domain clusters*. For the sites belonging to each cluster l, l=1,2,…,h_j, the aggregated vector $\hat{V}_l$ is calculated and sent to domain monitor DM_j. The aggregated information matrix $\hat{M}_j$ that contains the aggregated information vectors of the clusters $\hat{V}_l$, l=1,2,…,h_j, is sent to the higher levels.

Various approaches can be used for clustering the sites of a domain:

- Sites can be clustered randomly.
- A clustering function can be applied to each site's information vector and the sites that yield closer values are grouped together. This way the intra-domain clusters obtained consist of sites with similar characteristics and the aggregated information vector better represents the sites in the intra-domain cluster.
- The clustering can be performed so as to maximize the time periods during which the sites belonging to a given cluster are unavailable (as indicated by their ST and FT's). This way the start ($\hat{ST}$) and finish times ($\hat{FT}$) of an aggregated vector will better describe the availability of the sites in a cluster. In [16] a resource selection method is presented that increases the time overlapping of the tasks assigned to different sites and decreases it for tasks belonging to the same site. We can use a similar method for performing the clustering of the sites.

We show an example of the application of the intra-domain clustering aggregation technique, where the size of the information matrix M_j is reduced from $|D_j| = 8$ vectors to $h_j = 3$ vectors.

$$M_j = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_7 \\ V_8 \end{bmatrix} = \begin{bmatrix} (C_1, S_1, \ldots) \\ (C_2, S_2, \ldots) \\ \vdots \\ (C_7, S_7, \ldots) \\ (C_8, S_8, \ldots) \end{bmatrix} \Rightarrow \hat{M}_j = \begin{bmatrix} \hat{V}_1 \\ \hat{V}_2 \\ \hat{V}_3 \end{bmatrix} = \begin{bmatrix} (\hat{C}_1, \hat{S}_1, \ldots) \\ (\hat{C}_2, \hat{S}_2, \ldots) \\ (\hat{C}_3, \hat{S}_3, \ldots) \end{bmatrix}$$

The number of intra-domain clusters per domain influences the amount of information passed to higher levels and the efficiency of the scheduler's decision.

1.4.3.3 *Reducing Aggregated Information using Domination Relations*

Using the concept of *dominated resources,* we can further prune the number of information vectors processed by the domain monitors or the number of aggregated information vectors processed by the central monitor. Specifically, we will say that information vector V_1 *dominates* information vector V_2, if V_1 is better than V_2 with respect to all the cost parameters.

For example, consider the information vectors $V_1 = (C_1, S_1, FT_1)$ and $V_2 = (C_2, S_2, FT_2)$. We say that V_1 dominated V_2 if the following conditions hold:

$$C_1 > C_2,\ S_1 > S_2 \text{ and } FT_1 < FT_2$$

The V_2 information vector can then be discarded from further consideration, since the site (or domain) characterized by V_2 is inferior to the site (or domain) characterized by V_1 with respect to all parameters of interest.

1.4.4 Domain Selection Cost Functions

When a new task arrives the *CS* performs the following operations in order to select the appropriate domain for the task's execution:

- It discards all the aggregated information vectors that do not satisfy the task requirements (e.g. storage requirements).
- An optimization function is applied to the remaining vectors and the domain giving the largest value is selected.

1.5 Performance Evaluation

1.5.1 Simulation Environment

We consider a number of sites that are randomly grouped into domains, each of an approximately equal number of sites. Site *i* is characterized by its computational capacity C_i, measured in MIPS and number of tasks N_i under execution or in its queues. Unless stated otherwise, the capacities of the sites are chosen from a uniform distribution between 1000 and 10000 MIPS. The number of tasks at each site is also chosen from a uniform distribution between 5 and 200 tasks. One could argue that the assumption of a uniform distribution of tasks per site is not so realistic, since a good scheduling algorithm would result in a more balanced and correlated distribution. However, we are not interested in a specific scheduling algorithm, but in examining the quality of the information provided by the aggregation schemes for performing scheduling decisions. In our simulations we also examine other distributions of tasks to the sites. Each new task has workload uniformly distributed between 1000 and 10000 MI and no data dependencies, so no data transfers occur. Moreover, network related issues are not considered.

1.5.2 Aggregation Schemes Evaluated

We implemented and evaluated the following schemes:

- *FlatCpuFreeStart*: This scheme assumes a-priori knowledge of the task workloads. Site *i* calculates and publishes an information vector $V_i = \{C_i, FT_i\}$ containing its computational capacity C_i and the estimated future time FT_i when all the queued tasks will have completed their execution. The scheduler has complete knowledge of the information vectors of all the sites based on which it assigns a new task T_m to the site *i* that will execute the task sooner:

$$\min_i \{FT_i + \frac{W_m}{C_i}\}.$$

- *HierCpuFreeStart*: In this scheme the information vectors of the sites belonging to the same domain are aggregated. The aggregation of the site computational capacities and finish times is performed using the minimum representative operator: $\hat{C} = \min_i C_i$ and $\hat{FT} = \min_i FT_i$. The central scheduler *CS* assigns task T_m to the domain D_j that will complete the task sooner, using only the aggregated information vectors of the domains:

$$\min_j \{\hat{FT}_j + \frac{W_m}{\hat{C}_j}\}.$$

- The selected domain's scheduler DS_j then assigns the task to a domain site, having complete knowledge of the information vectors of all the sites in the domain. The assignment again is performed based on the minimum completion time criterion:

$$\min_{i \in D_j}\{FT_i + \frac{W_m}{C_i}\}.$$

- *FlatCpuTasks*: This scheme is similar to *FlatCpuFreeStart*, except that there is no a-priori knowledge of the task workloads. The information vector $V_i = \{C_i, N_i\}$ of site *i* contains its computational capacity C_i and the number of tasks N_i queued at it. A new task T_m is assigned to the site *i* that minimizes the optimization function (Section 1.4.4):

$$\min_i\{\frac{C_i}{N_i}\}.$$

- *HierCpuTasks*: In this scheme the information vectors of the sites belonging to the same domain are aggregated using the minimum representative and the additive operators, respectively: $\hat{C} = \min_i C_i$ and $\hat{N} = \sum_i N_i$. The central scheduler *CS* initially assigns, using only the aggregated information vectors of the domains, a task T_m to the domain D_j that minimizes the optimization function:

$$\min_j\{\frac{\hat{C}_j}{\hat{N}_j}\}.$$

- The selected domain's scheduler, DS_j, receives the task and assigns it to a domain site, having complete knowledge of the information vectors of all the sites in the domain. The assignment again is performed based on the same optimization function:

$$\min_{i \in D_j}\{\frac{C_i}{N_i}\}.$$

- *HierDominanceCpuTasks*: This scheme is similar to the *HierCpuTasks*, except that domination relations are applied to the vectors of the sites in a domain, before they are aggregated.

- *HierICCpuTasks*: This scheme is similar to the *HierCpuTasks*, except that the intra-domain clustering method is used, where sites are randomly clustered into intra-domain clusters.

- *HierDominanceICCpuTasks*: This scheme combines the *HierDominanceCpuTasks* and *HierICCpuTasks* schemes, where domination relations are applied to the vectors of the sites belonging to the same intra-domain cluster, before their aggregation.

1.5.3 Simulation Metrics

We are interested in the quality of the information produced by the aggregation schemes when making scheduling decisions. In our experiments we use the *Stretch Factor* (*SF*) metric, defined as the ratio of the task delay *TD* when scheduling is performed using complete resource information (*FlatCpuFreeStart, FlatCpuTasks*) over the task delay when an aggregation scheme is used (*HierCpuFreeStart, HierCpuTasks, HierICCpuTasks, HierDominanceICCpuTasks*). The task delay is the time that elapses from the task's creation until the completion of its execution. The *SF* is also encountered in the hierarchical networks related literature, where it is defined as the ratio of the average number of hops from a source to a destination when flat routing is used, over the corresponding value when hierarchical routing is used. We define the following stretch factor metrics:

- $$\mathrm{SFCpuFreeStart} = \frac{\mathrm{TD}_{\mathrm{FlatCpuFreeStart}}}{\mathrm{TD}_{\mathrm{HierCpuFreeStart}}}$$
- $$\mathrm{SFCpuTasks} = \frac{\mathrm{TD}_{\mathrm{FlatCpuTaks}}}{\mathrm{TD}_{\mathrm{HierCpuTasks}}}$$
- $$\mathrm{SFCpuTasksDominance} = \frac{\mathrm{TD}_{\mathrm{FlatCpuTaks}}}{\mathrm{TD}_{\mathrm{HierCpuTasksDominance}}}$$
- $$\mathrm{SFICCpuTasks} = \frac{\mathrm{TD}_{\mathrm{FlatCpuTaks}}}{\mathrm{TD}_{\mathrm{HierICCpuTasks}}}$$
- $$\mathrm{SFICCpuTasksDominance} = \frac{\mathrm{TD}_{\mathrm{FlatCpuTaks}}}{\mathrm{TD}_{\mathrm{HierICCpuTasksDominance}}}$$

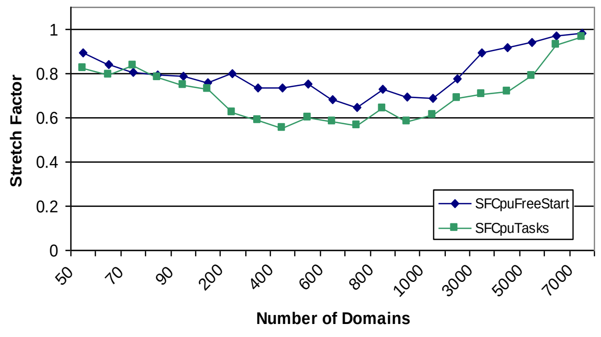
In all cases *SF*≤1, since when a scheduler has complete knowledge of the resources information, it can make better decisions than when this information is aggregated. An aggregation technique is efficient when its corresponding *SF* is close to 1. An additional metric for evaluating the schemes is the amount of information (number of information vectors) produced and used by the central scheduler in making its decisions.

1.5.4 Simulation Results

1.6 Simulation Results

In our experiments each site's characteristics are chosen among a finite set of values. For example, a site's computational capacity is an integer value between 1000 MIPS and 10000 MIPS, while the number of queued tasks is between 5 and 200 tasks. Thus, as the number of sites increases the probability that sites in different domains have similar information vectors also increases, and so does the probability that more than one "best" sites or sites similar to the "best" site exist in different domains. We represent this probability as $P_{multiple\text{-}best}$, and as our results will indicate it strongly affects the stretch factor. By “best” we mean the site that optimizes the metric of interest (task delay, or some other optimization function).

Figure 1.2 shows the measured stretch factors when 10000 Grid sites are clustered in a variable number of domains. The *HierICCpuTasks* and *HierDominanceICCpuTasks* aggregation schemes use *h*=5 intra-clusters in each domain. The stretch factor metrics behave similarly, that is, their value first decreases up to some point, after which they start increasing towards 1. This is because when the number of domains is small, then the number of sites per domain is quite high (e.g., 200) increasing the $P_{multiple-best}$ probability. As the number of domains increases, $P_{multiple-best}$ decreases and the stretch factors also decrease. After some point, as the number of domains increases and the number of sites per domain decreases, the quality of information produced by the aggregation schemes improves. This is because when there are few sites per domain, the aggregated information better represents the characteristics of its sites.



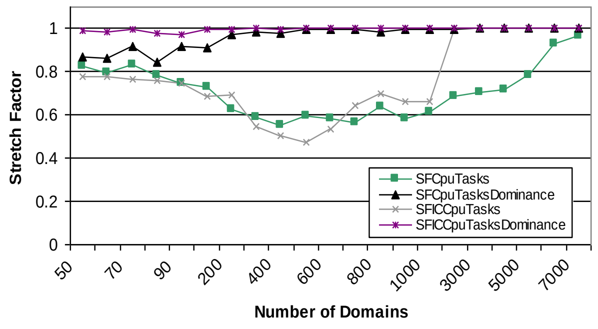


Figure 1.2: (a) The SFCpuFreeStart and the SFCpuTasks (b) the SFCpuTasks, SFCpuTasksDominance, SFICCpuTasks and the SFICCpuTasksDominance stretch factors, when 10000 Grid sites are clustered in a variable number of domains.

SFCpuFreeStart is generally larger than *SFCpuTasks* (**Figure 1.2**a), indicating that different parameters in the information vectors and different operators used for their aggregation result in different quality for the information provided to the scheduler. We also observe that *SFCpuTasks* and *SFICCpuTasks* (**Figure 1.2**b) take similar values; however, when 2000 domains are used the *SFICCpuTasks* metric reaches 1. This is because in this case each domain has 5 sites and 5 intra-domain cluster, and the aggregation scheme that produces 5 information vectors per domain, describes exactly the resources' information (in fact, no aggregation is performed in that case). We also observe that the *HierDominanceCpuTasks* and *HierDominanceICCpuTasks* aggregation schemes produce the best results. This indicates that the dominance operation, which discards dominated information vectors, improves the quality of the information provided to the scheduler. This is also confirmed when comparing the *HierDominanceICCpuTasks* and the *HierICCpuTasks* aggregation schemes. We should also note that the number of domains and sites used in our simulations, may be seem quite large in comparison to the usual values in existing Grid Networks. We took this decision in order to examine the full dynamics of the proposed aggregation techniques.

Moreover, the *HierDominanceICCpuTasks* scheme yields results that are very close to those obtained by the *FlatCpuTasks* scheme, while providing less information vectors to the central scheduler. Reducing the number of intra-domain clusters, reduces the number of information vectors produced, but also reduces the quality of the information provided, as measured by the corresponding stretch factor. **Table 1.1** shows the number of information vectors provided by each scheme when 10000 sites are clustered in 100 domains. Also, it is not only the amount of resource information transferred that it is reduced, but also the number of control messages exchanged, the computational overhead for processing the information and the storage overhead for storing it.

Aggregation Scheme	Number of Information Vectors
FlatCpuFreeStart	N = 10000
HierCpuFreeStart	$L = 100$
FlatCpuTasks	N = 10000
HierCpuTasks	$L = 100$
HierDominanceCpuTasks	$L = 100$
HierICCpuTasks (*h*=5 inter-domain clusters)	$L \cdot h = 500$
HierDominanceICCpuTasks (*h*=5)	$L \cdot h = 500$

Table 1.1: The number of information vectors produced by each aggregation scheme, when N = 10000 sites are clustered in L = 100 domains.

Figure 1.3 shows the *SFCpuTasks*, *SFCpuTasksDominance*, *SFICCpuTasksDominance,* and *SFICCpuTasks* stretch factors, when a variable number of sites are clustered in 20 domains. The *SFs* initially decrease and then, as the number of sites increases further, start increasing towards 1. This is because, initially, having more sites per domain reduces the quality of information provided by the schemes to the central scheduler. The exact amount of this reduction depends on the aggregation operators applied and the aggregation scheme used. For this reason we observe that the *HierDominanceCpuTasks* and *HierDominance-ICCpuTasks* schemes outperform the *HierCpuTasks* and *HierICCpuTasks* schemes. However, after a point, when the number of sites in each domain becomes large, the probability $P_{multiple\text{-}best}$ that there is a site in the selected domain that can execute a task as fast as the “best” site, becomes large and the *SF*s increase towards 1. This is also related to the number of different values resource characteristics can take. **Figure 1.4** gives a better insight into this.

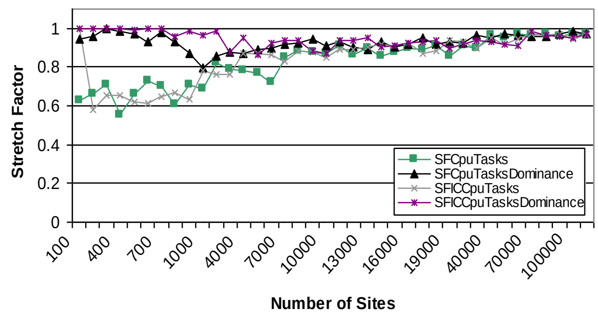


Figure 1.3: The SFCpuTasks, SFCpuTasksDominance, SFICCpuTasks, and the SFICCpuTasksDominance SFs, when a variable number of sites are clustered in 20 domains.

Figure 1.4 shows the results obtained for the *SFs* when changing the upper and lower limits of the uniform distributions assumed for the computational capacities and the number of tasks at the sites. The scenarios/probabilistic distributions used are presented in **Table 1.2**. In **Figure 1.4** we illustrate the *SFCpuTasks* stretch factors obtained for the case where a variable number of sites are partitioned into 20 domains. Note that the number of different information vectors that the *UD03* scenario can produce is larger than the ones produced by the *UD02* scenario and even larger than those produced by the *UD01* scenario. We observe that the *SFCpuTasks* values decrease as the number of distinct values the sites’ characteristics can take increase. This is because a large number of possible and different information vectors reduce the probability $P_{multiple\text{-}best}$ that more domains will have sites with information vectors similar to the “best” site. Corresponding experiments were performed for all the proposed aggregation schemes, producing similar results.

Scenario	Computational Capacity (max/min)
UD01	10000/1000
UD02	100000/100
UD03	1000000/10

Table 1.2: Scenarios UD01 UD02 and UD03 correspond to different choices for the upper/ lower limits of the uniform distributions assumed for the computational capacities and the number of tasks at the sites.

Finally, we should note that the number of different information vectors produced by the aggregation schemes depends on the aggregation operators used. Specifically, as stated in [2], when two additive parameters are used, the number of possible information vectors produced is exponential, while when two restrictive operators are used (as in the information vectors we use), the number of different information vectors is polynomial. This illustrates the importance of the resource parameters and the aggregation operators on the efficiency of the aggregation schemes.

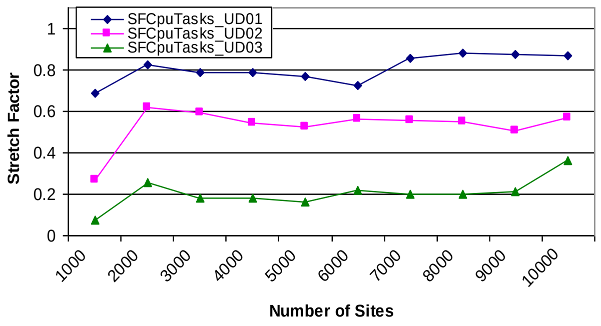


Figure 1.4: The SFCpuTasks SFs for the UD01, UD02 and UD03 scenarios (**Table 1.2**), when a variable number of sites are clustered in 20 domains.

1.7 Conclusions

We proposed several techniques for aggregating the resource information of the sites in hierarchical Grid domains and performing task scheduling using this information. We performed a number of simulation using the *Stretch Factor* (*SF*) as the main metric for measuring aggregation efficiency. The *SF* is defined as the ratio of the task delay when the task is scheduled using complete resource information over the task delay when an aggregation scheme is used. We observed that in many cases the proposed schemes achieve large information reduction, while enabling good task scheduling decisions as indicated by the *SF* achieved. We studied the trade-off between the amount of information exchanged (and used by the scheduler) and the scheduling efficiency. We also introduced domination relations and showed that they can increase the quality of the aggregated information. Finally, we observed that the uniformity of the sites' characteristics significantly affects the *SF*s achieved.

2 Scalability Study of Harmony Service Plane

Work Package 5 is responsible for the supporting studies to the other WP's in the Phosphorus project. The main purpose of the activities in WP5 consists of the design and evaluation of innovative architectures and algorithms to efficiently manage optical Grid infrastructures. This work is essential because it allows to effectively test interesting alternatives with short turn-around times. The Harmony – WP5 collaboration can be situated within these supporting studies activities.

The main objective of the collaboration between WP1 and WP5 is to build a simulation environment to evaluate the different architectural approaches of the Network Service Plane in Harmony. Thus, the results of the simulations executed by WP5 aim to assess the performance and scalability of the service plane as well as to determine which architecture performs optimally under different scenarios.

As the prototype and testbed infrastructure is quite limited for extensive evaluation of the Harmony service plane, WP5 has built a simulation environment which is able to simulate much larger topologies to allow performance evaluation under realistic circumstances. Also, simulation allows a wide range of service and control algorithms to be rapidly tested under various network and traffic conditions. As such, the main purpose of this collaboration is to provide feedback to the Harmony designers and developers about the high-level and large-scale behaviour of their software stack through simulation analysis.

2.1.1 Simulator

The simulator was built using the Python programming language. As our aim was to use discrete event simulation (DES), we selected the SimPy library which offers an object-oriented and process-based DES language. The fundamental entity of the simulator is the Resource object provided by the SimPy package, which simulates a queueing system and corresponds to an IDB in the Harmony system.

Furthermore, to incorporate features for the creation, manipulation and study of complex networks, we required an additional Python library. For this, we selected the NetworkX package for its advanced support in network generation capabilities and various network algorithms such as routing and topology evaluation. In particular, the NetworkX package was used for random graph generation using the Barabasi-Albert preferential attachment model. Also routing functions and extraction of various topology parameters were extensively used.

The simulator itself makes a distinction between the three basic architectural approaches, more specifically:

- Centralized: a single IDB functions as point of entry, and has a direct connection to all HNAs.
- Hierarchical: multiple levels of IDBs are assembled in a tree structure. Each IDB can function as point of entry, but only the leaf nodes in the tree can directly control the HNAs. Thus, when a connection request arrives at a point of entry that is not in direct or indirect control of the relevant HNAs, the request is propagated upwards in the tree until an IDB is found that has indirect control of all HNAs that are part of the connection request. Refer to Phosphorus Deliverable D1.4 “Definition and development of the Network servie Plane and northbound interfaces development” for the full description on how the hierarchical approach functions.
- Distributed: Each HNA has a single IDB that it controls directly, while the IDB network is assumed to be a full mesh (other scenarios are studies in D5.5 addendum).

Further explanation is required for the hierarchical approach, as it is not directly obvious how to construct the tree structure on top of the HNA network. For this, we chose an easily implementable technique for which we cannot prove any optimality properties. On the contrary, our results indicate that the technique can be easily parameterized and this may lead to large variations in the behavior of the system, ultimately implying that the construction of such an IDB hierarchy corresponds to an optimization problem. Our proposed algorithm is based on the grouping degree, which indicates how many lower level entities will be controlled by an upper level IDB. More specifically, we start from the HNA network, and introduce IDB’s with each IDB controlling the exact number of HNA nodes that correspond to the grouping degree. Any HNA’s that remain are then controlled by an additional IDB. We repeat this process for the IDB that were introduced, and create new IDB’s on higher levels until a single root IDB is created.

For more details with regard to the design and implementation of this simulation environment, refer to Deliverable D5.9 “Extended Simulation Environment”.

2.1.2 Simulated scenarios

In this deliverable, we will only show results from a single scenario to demonstrate the validity of our simulation environment, and sketch some preliminary conclusions with respect to the scalability behavior of the various approaches. More specifically, we will discuss how the average duration of a connection request behaves for the different Harmony architectures.

Analysis of a working Harmony IDB controller implementation has shown that the processing duration for a forwarded request is 20 ms, while a request that must be processed (leading to individual HNA reservations) has a duration of 25 ms. Likewise, the response time of a HNA controller is 7 ms.

We generated a random 100 node HNA network, using the Barabasi-Albert preferential attachment model. The Barabasi-Albert parameter, which corresponds to the number of edges that are attached from a new node to an existing node, was configured to 3. This led to an average degree of 5.58 for the HNA network.

The arrival process is the typical Poisson process, i.e. with exponentially distributed inter-arrival times. The point of entry IDB is chosen ad random, thus leading to a uniform traffic pattern. All processing queues in the IDB's and HNA's have a first-come first-serve policy,

Furthermore, the grouping degree in the hierarchical approach was varied to study its influence on the scalability properties. For the given network size of 100 nodes, we chose grouping degrees of 2, 5, 10, 25 and 50.

2.1.3 Preliminary results

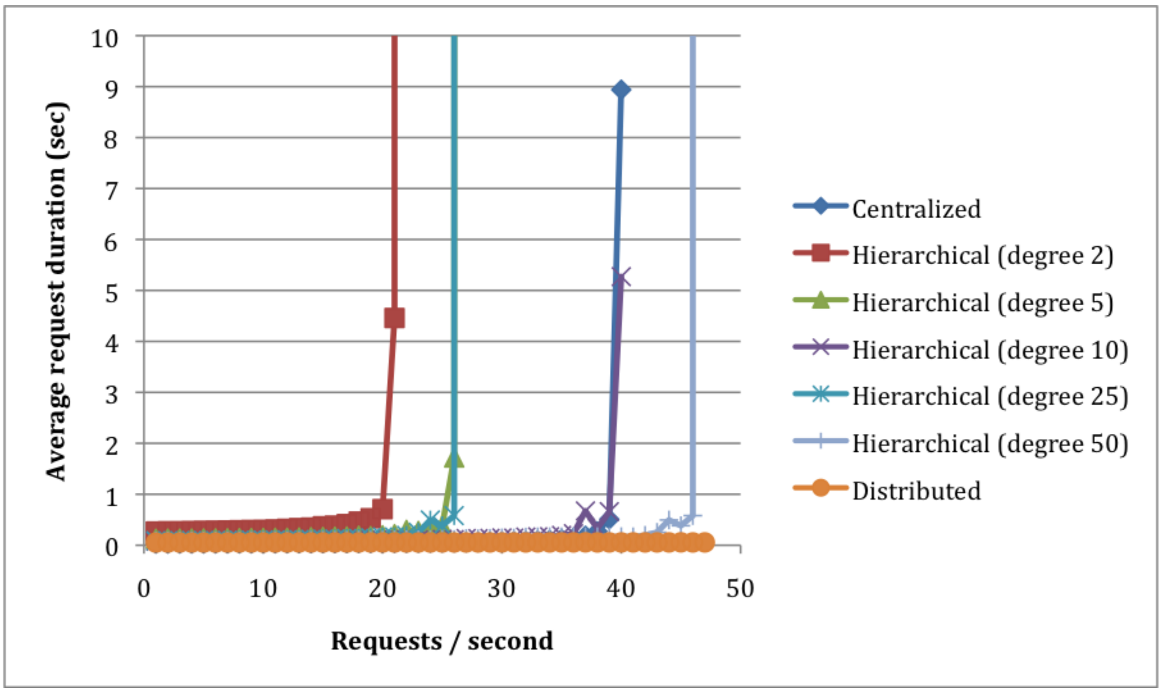


Figure 2.1: Scalability of average request duration for different Harmony architectures

Figure 2.1 shows the average request duration for various Harmony architectures. A number of interesting conclusions can be drawn from this figure. First, every architecture has a certain arrival rate (requests per second) at which the average duration increases very quickly. This is the point where the system becomes unstable as more arrivals are generated than the system can process. Obviously, this should be avoided at all cost, and this point is highly dependent on the arrival pattern (care must be taken for non-uniform loads as this may overload only certain parts of the system), the architecture, the processing speed of the IDB/HNAs (more efficient implementation can improve scalability), and other parameters.

It is also clear that the hierarchical system only performs better than the centralized under certain circumstances (i.e. grouping degrees 10 and 50 lead to similar or better performance, the other grouping degrees lead to much worse behaviour). This implies that the proposed algorithm for building the hierarchy of IDB's can be further improved. As such, this problem will be further studied in the future and will lead to a more intensive collaboration with Harmony's designers.

Finally, also note that the distributed system does not show an apparent instable point. However, this is mainly for clarity reasons, as we have seen (and we can also predict) that the distributed scenario becomes overloaded for an arrival rate over 220 requests per second: more than 5 times most other architectures can handle.

2.1.4 Conclusion

In this section, we described our simulation environment that was created to study scalability issues of the Harmony service plane solution. More specifically, we implemented a simulator that incorporates the different architectural approaches: centralized, hierarchical and distributed. We demonstrated some results related to the average process duration of a connectivity request, and showed the existence of large variations in the behavior of the various architectures. More specifically, we demonstrated the poor scalability of the centralized approach, and the very high scalability performance of the distributed approach. Likewise, we showed that the hierarchical approach is highly dependent on the structure of the Harmony service plane tree. This implies that further research will be focused on optimizing the location and interconnection of the service nodes.

3 Conclusions

We proposed several techniques for aggregating the resource information of the sites in hierarchical Grid domains and performing task scheduling using this information. We performed a number of simulation using the *Stretch Factor* (*SF*) as the main metric for measuring aggregation efficiency. The *SF* is defined as the ratio of the task delay when the task is scheduled using complete resource information over the task delay when an aggregation scheme is used. We observed that in many cases the proposed schemes achieve large information reduction, while enabling good task scheduling decisions as indicated by the *SF* achieved. We studied the trade-off between the amount of information exchanged (and used by the scheduler) and the scheduling efficiency. We also introduced domination relations and showed that they can increase the quality of the aggregated information. Finally, we observed that the uniformity of the sites' characteristics significantly affects the *SF*s achieved.

Also, we described our simulation environment that was created to study scalability issues of the Harmony service plane solution. More specifically, we implemented a simulator that incorporates the different architectural approaches: centralized, hierarchical and distributed. We demonstrated some results related to the average process duration of a connectivity request, and showed the existence of large variations in the behavior of the various architectures. More specifically, we demonstrated the poor scalability of the centralized approach, and the very high scalability performance of the distributed approach. Likewise, we showed that the hierarchical approach is highly dependent on the structure of the Harmony service plane tree. This implies that further research will be focused on optimizing the location and interconnection of the service nodes.

4 References

[1] S. Zanikolas, R. Sakellariou, A taxonomy of grid monitoring systems, FGCS, Vol. 21, pp. 163-188, 2005.

[2] Z. Wang, J. Crowcroft, Quality-of-Service Routing for Supporting Multimedia Applications, JSAC, Vol. 14, No. 7, pp. 1228-1234, 1996.

[3] R.Wolski, N. Spring, J. Hayes, The network weather service: a distributed resource performance forecasting service for metacomputing, FGCS, Vol. 15 , pp. 757-768, 1999.

[4] L. Kleinrock, F. Kamoun, Hierarchical routing for large networks. Performance evaluation and optimization, Computer Networks, Vol. 1, No. 3, pp. 155-174, 1977.

[5] W. C. Lee, Topology aggregation for hierarchical routing in ATM networks, ComCom Review, Vol. 25, No. 2, pp 82-92, 1995.

[6] P. Van Mieghem, Topology information condensation in hierarchical networks, Journal of Computer and Telecommunications, Vol. 31, No. 20, pp. 2115–2137, 1999.

[7] D. Bauer, J. Daigle, I. Iliadis, P. Scotton, Topology aggregation for combined additive and restrictive metrics, ComNet, Vol. 50, No. 17, pp. 3284-3299, 2006.

[8] I. Ahmad, Y.-K. Kwok, M.-Y. Wu, K. Li, Experimental Performance Evaluation of Job Scheduling and Processor Allocation Algorithms for Grid Computing on Metacomputers, IPDPS, US, pp. 170-177, 2004.

[9] T. Braun et al , A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems, JPDC, Vol. 61, No. 6, pp. 810-837, 2001.

[10] S. Zhuk et al, Comparison of Scheduling Heuristics for Grid Resource Broker, Mexican International Conference in Computer Science, pp. 388-392, 2004.

[11] Y. Cardinale, H. Casanova, An evaluation of Job Scheduling Strategies for Divisible Loads on Grid Platforms, HPC&S, Germany, 2006.

[12] R. Buyya, et. al, Scheduling Parameter Sweep Applications on Global Grids: A Deadline and Budget Constrained Cost-Time Optimisation Algorithm, SPE, Vol. 35, No. 5, pp. 491-512, 2005.

[13] W. Smith, I. Foster, V. Taylor, Scheduling with advanced reservations, IPDPS, pp 127-132, 2000.

[14] K. Christodoulopoulos, N. Doulamis, E. Varvarigos, Joint Communication and Computation Scheduling in Grids, CCGrid, pp. 17-24, 2008.

[15] C. Mendes, D. Reed, Monitoring large systems via statistical sampling,. High Performance Computing Applications, Vol. 18, No. 2, pp. 267-277, 2004.

[16] N. Doulamis, P. Kokkinos, E. Varvarigos, Spectral Clustering Scheduling Techniques for Tasks with Strict QoS Requirements', Europar, pp. 478-488, 2008.

[17] R. Renesse, K. Birman, W. Vogels, Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining, ACM Trans. on Computer Systems, Vol. 21, No. 2, pp. 164-206, 2003.

[18] C. Intanagonwiwat, et. al, Directed diffusion for wireless sensor networking. ToN, Vol. 11, pp. 2-16, 2003.

5 Acronyms

[CM]	Central Monitor
[CS]	Central Scheduler
[ET]	End Times
[GNS]	Grid Network Services
[HNA]	Harmony NRPS Adapter
[IDB]	Inter-Domain Broker
[MI]	Million Instructions
[MIPS]	Million Instructions Per Second
[NRPS]	Network Resource Provisioning Systems
[P2P]	Peer to peer
[SF]	Stretch Factor
[**ST]**	Start Times