Future Generation Computer Systems 28 (2012) 9–23

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Scheduling efficiency of resource information aggregation in grid networks

P. Kokkinos *, E.A. Varvarigos

Department of Computer Engineering and Informatics, University of Patras, Greece
Research Academic Computer Technology Institute, Patras, Greece

ARTICLE INFO

Article history:
Received 1 March 2011
Received in revised form
23 June 2011
Accepted 30 June 2011
Available online 22 July 2011

Keywords:
Grid networks
Information aggregation
Scheduling
Stretch factor

ABSTRACT

We consider information aggregation as a method for reducing the information exchanged in a Grid network and used by the resource manager in order to make scheduling decisions. In this way, information is summarized across nodes and sensitive or detailed information can be kept private, while resources are still publicly available for use. We present a general framework for information aggregation, trying to identify issues that relate to aggregation in Grids. In this context, we describe a number of techniques, including single point and intra-domain aggregation, define appropriate grid-specific domination relations and operators for aggregating static and dynamic resource information, and discuss resource selection optimization functions. The quality of an aggregation scheme is measured both by its effects on the efficiency of the scheduler's decisions and also by the reduction it brings on the amount of resource information recorded, a tradeoff that we examine in detail. Simulation experiments demonstrate that the proposed schemes achieve significant information reduction, either in the amount of information exchanged, or in the frequency of the updates, while at the same time maintaining most of the value of the original information as expressed by a stretch factor metric we introduce.

1. Introduction

Grids consist of geographically distributed and heterogeneous computational and storage resources that may belong to different administrative domains, but are shared among users. The Grid resource manager (scheduler) receives user requests and assigns tasks to resources so as to optimize some objective function. Scheduling decisions are made based on static and dynamic resource information, including the computation and storage capacities, their availability, the number of tasks queued and other parameters of interest, which are usually collected by information services.

In this work, we study the operation of resource information aggregation in Grids. Through information aggregation, the resource characteristics are summarized before being sent to scheduling or other mechanisms (e.g., data managers), while Grid resources are still distributedly controlled. Resource-related information size and dynamicity grows rapidly with the size of the Grid, making the aggregation and use of this massive amount of information a challenge for the resource management system. In addition, as computation and storage tasks are conducted increasingly non-locally and with finer degrees of granularity, the flow of information among different systems and across multiple domains will increase. Information aggregation techniques are important in reducing the amount of information exchanged and the frequency of these exchanges, while at the same time maximizing its value to the Grid resource manager or to any other desired consumer of the information. An additional motivation for performing information aggregation is confidentiality and interoperability, since as more resources or domains of resources participate in the Grid, it is often desirable to keep sensitive and detailed resource information private, while resources are still being publicly available for use. For example, it will soon become necessary for the interoperability of the various cloud computing services (e.g., Amazon EC2 and S3, Microsoft Azure) that the large quantity of resource-related information is efficiently abstracted, before it is provided to the task scheduler. In this way, the task scheduler will be able to use efficiently and transparently the resources, without requiring services to publish in detail their resource characteristics. In any case, the key to information aggregation is the degree to which the summarized information helps the scheduler make efficient use of the resources, while coping with the dynamics of the Grid and the varying requirements of the users.

We propose several information aggregation techniques, which are presented in a general way, so as to permit their adaptation to specific situations, or their combination for the creation of new aggregation schemes. We are mainly interested in resource information that influences the decisions made by the Grid scheduler, while we do not consider parameters that cannot be directly aggregated (e.g., operating system version). Even

* Corresponding author at: Department of Computer Engineering and Informatics, University of Patras, Greece.
E-mail address: kokkinop@ceid.upatras.gr (P. Kokkinos).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.06.008

though the proposed information aggregation techniques are in line with the current hierarchical structure of Grid Networks and the related monitoring systems [1,2], our work follows a more theoretical approach, trying to identify generic issues that relate to aggregation, instead of studying how aggregation affects Grid Networks with specific characteristics. Specifically, we assume that resource sites are grouped into domains and the characteristics of the sites in each domain are aggregated before being sent to the resource scheduler. For aggregating the information, we use concepts and ideas derived from multi-criteria optimization [3]. In particular, each site is characterized by a vector of cost parameters that record its computation and storage capacity, their availability with time, and other parameters of interest, handling in this way the multi-dimensionality of the sites' characteristics. The cost vectors of the sites in a given domain are aggregated into a single (single point) or multiple (intra-domain aggregation) cost vectors for the entire domain, by performing appropriate associative operations to the site cost parameters. The domain cost vectors are collected from the monitoring system. We also introduce so-called *domination relations* that aim at reducing the number of vectors aggregated and stored. A resource site is said to be dominated by another one, when it is inferior to that with respect to all the resource parameters of interest. When a task request arrives, the scheduler applies an optimization function to the domain cost vectors in order to select the domain where the task will be executed. The task is then transferred to the selected domain and is assigned to a site in it, using exact resource information.

The undesirable side of information aggregation is that the efficiency of a scheduler using such information can be negatively affected. Even though there are obvious benefits in reducing through aggregation the amount of information exchanged and stored in a Grid, the job scheduler has less detailed and accurate information for decision making. This introduces an interesting *tradeoff between the amount and frequency of information exchanges and the value this information has for making efficient scheduling decisions*. We propose information aggregation schemes that produce aggregated (summarized) information of different quantity, granularity and therefore quality, improving or deteriorating scheduling decisions.

As a metric of the quality of the aggregated information we introduce 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. We perform a large number of experiments to evaluate the proposed aggregation techniques, using a relatively simple simulation environment. Towards this end, we designed the simulations scenarios so that they are generic, evaluating various different scenarios: grids with many or few sites and domains, tasks with small or large workloads, resources with small or high computation capacities, etc. We also measure the number of resource information updates triggered by each aggregation scheme and the amount of information transferred. The simulation results show that the proposed schemes achieve significant information reduction, both in terms of size and of frequency of updates, while maintaining good scheduling quality. The uniformity or lack of uniformity of the sites' and tasks' characteristics is found to significantly affect the quality of the aggregation. In addition, we observe that that the type of parameters aggregated and the operators used for their aggregation play a significant role.

The present work handles information aggregation in Grids as a separate and important issue, attempting to identify the main issues, parameters, dependencies and side-effects related to the aggregation operation. Even though there are other works in Grids that consider aggregation as an available mechanism, they usually consider specific policies and scenarios, and they do not address through appropriate metrics the effect information aggregation has on the quality of the scheduling decisions made. In addition, other previous works focus on Data Networks where information aggregation is an old and well known mechanism used for making routing scalable and efficient. We use the same idea, making possible for a task scheduler to use transparently the globally available resources without knowing their full details, in the same way a router forwards a packet to the next gateway, without knowing all the intermediate nodes the packet will pass through, towards its destination.

The remainder of the paper is organized as follows. In Section 2 we report on previous work. The information aggregation problem is formulated in Section 3. Section 4 introduces the proposed aggregation techniques. In Section 5 we present an example of applying the techniques introduced in a hierarchical Grid Network. In Section 6 we experimentally evaluate the proposed techniques. Finally, conclusions are presented in Section 7.

2. Previous work

Information aggregation has been previously studied mainly in the context of hierarchical Data Networks [4], where it is performed on network-related parameters in order to facilitate routing. Resource information aggregation in Grids has not been studied in detail, despite its practical importance and its impact on the efficiency of the scheduling decisions. Actually, most scheduling algorithms proposed [5,6] make their decisions using exact resource information. In the following, we present prior work on the relevant topics.

Task scheduling is usually performed at two levels [7,8]; at the higher level, a central scheduler decides the site or domain a task will be executed on, while at the lower level a local scheduler selects the exact machine where the task will be executed. Most scheduling algorithms try to minimize the total average task delay [9] or maximize resource utilization, even though other performance metrics can also be used. The authors in [10] incorporate Grid economic models and propose scheduling algorithms that support deadline and budget constraints. Another criterion used in comparing scheduling schemes is the degree of fairness achieved among users in the use of the resources [11,12]. Finally, scheduling with advance reservations of computational and network resources has also received a great deal of attention in Grid and Data Networks [13,14]. A taxonomy of Grid resource schedulers is presented in [5,6]. In Grid Networks, resource information collection is performed by the monitoring systems. In [15] a number of monitoring systems are presented and categorized based on their architecture, as defined in the Global Grid Forum's Grid Monitoring Architecture (GMA) [16].

In existing Grid and Data Networks, sites/machines/nodes are organized in hierarchical structures (e.g., the EGEE [17] and the Internet). A node in a domain communicates with nodes belonging to other domains using specific border nodes. Hierarchical routing is a major issue for Data Networks, and is important for reducing the memory requirements at the routers (border nodes) for the very large topologies encountered in the Internet's infrastructure. A topology is broken down into several layers of hierarchy, thus downsizing the routing tables required, but this comes at the expense of an increase in the average path length. [18] is one of the first works investigating hierarchical routing, where clustering structures are introduced to minimize the routing tables required. Bounds are also derived on the maximum increase in the path length for a given table size. A central issue in hierarchical routing is topology information aggregation [4,19]. Aggregation techniques in hierarchical topologies try to summarize and compress the topology information advertised at higher levels. In order to perform routing and network resource allocation efficiently, the aggregated information should adequately represent the topology and the characteristics/metrics of the network.

Delay and bandwidth are network-related metrics that are usually aggregated along with the topology. In [4] a number of topology aggregation techniques are presented: In the Symmetric approach a set of nodes (a domain) is collapsed into a single virtual node. In the Full-Mesh approach a logical link between each pair of border nodes is used to build the aggregated topology. The Star approach is an intermediate scheme, where a virtual node is connected over logical links to border nodes. An important issue in topology aggregation is the choice of the parameters that are aggregated along with the topology. In [19] these parameters are distinguished into three classes: additive, min–max and the combination of additive and min–max metrics (multi-criteria). Topology aggregation based on such metrics is investigated in [20] using mainly the Star approach, and considering only network-related parameters (namely, delay and bandwidth). In the same context [20] presents a topology aggregation scheme that is subject to multi-criteria (delay and bandwidth) constraints. Specifically, a transition matrix is constructed containing the minimum information that describes exactly the traversing characteristics of a domain, that is, the characteristics of the paths connecting border (ingress–egress) node pairs of the domain.

The idea of aggregation has also appeared in sensor networks [21,22], where it is often called data fusion, as a way to reduce the amount of data transferred towards the sink node. Relatively recently, information aggregation appeared as an interesting topic in Peer-to-Peer (P2P) [23] and P2P Grid systems [24–26]. In [23] the authors present a distributed information management system called Astrolabe, which continuously computes summaries of the data collected from various resources (machines, sensors, other devices), using on-the-fly aggregation. In this way the rate of information flow at each participating node is bound, and independent of system size. P2P Grids belong to the class of Desktop Grids, and extend the applicability of Desktop Grid computing by adding high parallel efficiency. In [24] the authors present an information aggregation system for P2P Grids, focusing on the system's architecture and the aggregation tree management mechanisms. In [25] the authors are interested in trusted relations in P2P Grids and for this purpose they propose a trust overlay network that models these relations. Through this network the peer trust scores are collected and aggregated so as to yield a global reputation. In [26] a scalable grid monitoring architecture is proposed that builds distributed aggregation trees on a structured P2P network. In general the works [23–26] focus on the architecture and on the mechanisms of the system performing the aggregation, while in the current work we assume that such a mechanism/system is in place and examine the aggregation operations performed for different parameters, the applied aggregation policies and the effects they have on scheduling efficiency. Also, our work applies to all kinds of Grids and not specifically to Desktop Grids.

In addition, resource information aggregation and task scheduling issues have been investigated in a few works, which focus, however, on particular systems and under specific assumptions [27–29]. In the Monitoring and Discovery System 2 (MDS2) [27,28] resource management system, information from individual information providers (e.g., resources) is aggregated into collections, where each collection may correspond to a different virtual organization (VO). [29] is a quite recent work where aggregated resource information and scheduling issues are considered. It proposes two aggregation policies (simple and categorized) and two scheduling policies that use the aggregated information. In the current work we are interested more in the parameters aggregated, in the operators used and in the policies applied to summarize them, while we use a simple scheduling policy to evaluate the effects of the aggregation on the scheduling efficiency. We also investigate the tradeoffs between the amount of information exchanged, the frequency of updates required, and the quality of the aggregated information. Our work is more general, and attempts to fill the gap between the topology information aggregation works presented in [4,19] and Grid computing.

Finally, we have presented part of this work in [30]; however, here we extend it significantly by considering additional parameters (e.g., time availability, network related parameters), by providing examples of using the aggregated information and by running all the experiments in more realistic simulation conditions, measuring additional metrics and providing new insights.

3. Problem formulation

Our aim is to design efficient and flexible aggregation schemes that can be applied to Grids without a significant penalty on the quality of the scheduling (and probably other) decisions taken with the aggregated information. We formulate the problem in a generic way, without assuming Grid Networks with specific characteristics. The scheduling policies we assume are relatively simple, since combining the aggregation schemes with a large set of scheduling policies would obscure the focus on the aggregation issues.

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 an *information matrix* whose rows are the resource site 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 $1, 2, \ldots, |D_j|$ are the sites contained in domain D_j. By performing appropriate operations, to be discussed later, on the information vectors contained in the information matrix, M_j is transformed into the *aggregated information matrix* $\hat{M}_j$.

The Grid scheduling problem is usually viewed as a two-level hierarchical problem. At the first level, called *meta-scheduling*, a meta-scheduler allocates tasks to sites, while at the second level, called *local scheduling*, each site schedules the tasks assigned to it on its local computing elements. This is the approach followed by many grid middleware systems, including gLite [31], where the Workload Management System (WMS) selects the site where a task will be executed and the local scheduler chooses the Working Node (WN) it will be executed on after reaching that site. Similarly, in this paper scheduling is performed at 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 (Fig. 1). Information collection and aggregation is performed, similarly, by a two level monitoring system, consisting of a central monitor CM and the domain monitors $DM_j,\ j = 1, 2, \ldots, L$.

A user located at some site generates tasks $T_m,\ m = 1, 2, \ldots,$ with computational workload W_m, that have to be scheduled and then executed at a resource site.

4. Information aggregation

In this section we present the resource information parameters of interest, the operators applied and the proposed aggregation techniques.

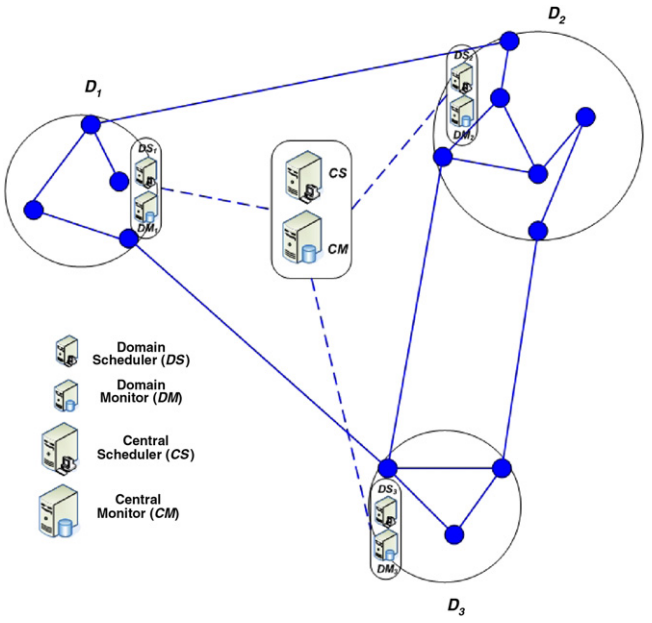

Fig. 1. A two-level hierarchical scheduling and monitoring system. Each domain j has a domain scheduler DS_j and a domain monitor DM_j, while there is also a central scheduler CS and a central monitor CM.

4.1. Operational framework

In Algorithm 1 we present, in pseudocode, the sequence of operations performed by the information collection mechanism and the proposed aggregation scheme.

Algorithm 1 Resource information collection & aggregation

1. Each site i, $i = 1, 2, \ldots, N$, belonging to some domain D_j periodically or reactively (driven by information changes) publishes its information vector V_i to the domain monitor DM_j.
2. Each domain monitor DM_j, $j = 1, 2, \ldots, L$, puts together these vectors to form the information matrix M_j.
3. Each domain monitor DM_j, $j = 1, 2, \ldots, L$, 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 operations performed by a task 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 request and selects the exact site the task will be scheduled on, using exact resource information.

Generally, as the number of sites in a domain D_j increases, the amount of information collected by the domain monitors DM_j also increases. Therefore, it is necessary for the information contained in each domain information matrix M_j to be aggregated/summarized. This is done by performing appropriate associative operations (addition, maximization, etc.) on the parameters of the sites' information vectors, in order to transform the information matrix M_j into the *aggregated information matrix* $\hat{M}_j$, which contains a smaller number of vectors than the original matrix. The operators used for summarizing the information depends on the types of the parameters involved.

Also, in Algorithms 1 and 2, we assume that resource information is pushed to the central system (CS and CM) by the local systems (DS and DM). Alternatively, the central system could also pull periodically this information. The pull method (and any similar periodic method) is generally less efficient, since in this way the various resource information changes are not immediately propagated to the system taking the scheduling decisions. However, the pull method (and any similar periodic method) results in smaller traffic, since there is not information exchange between the local and the central systems every time the information of the resources change. On the other hand, as we will later on exhibit (Section 5), by aggregating information it is possible to reduce the number of times this information is advertised—pushed to the central system after every change, reducing in this way the generated traffic.

In what follows, we elaborate on the resource parameters of interest, and the associative operators and aggregation techniques proposed. We also present optimization functions that can be applied by the CS to select the optimal information vector.

4.2. Information parameters and aggregation operators

The resource information parameters (both static and dynamic) of interest in this work, the operators used for their aggregation and the benefits we get for different choices of the operators are discussed next:

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

$$\hat{C}_j = \min_{i \in D_j} C_i, \qquad \hat{C}_j = \sum_{i \in D_j} C_i \quad \text{or} \quad \hat{C}_j = \operatorname*{avg}_{i \in D_j} C_i.$$

  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 minimization, the additive and the average operators (and possibly other operators, such as maximization) could all be used so that a number of capacity-related features of the domain are recorded.
- The available (free) computational capacities FC_i, measured in MIPS, of the sites in domain D_j can similarly be aggregated by performing a minimum representative or an additive operation:

$$\hat{FC}_j = \min_{i \in D_j} FC_i \quad \text{or} \quad \hat{FC}_j = \sum_{i \in D_j} FC_i.$$

  The storage capacities S_i of the sites, measured in MB, in a domain D_j can be aggregated by using the minimum representative or the additive operation:

$$\hat{S}_j = \min_{i \in D_j} S_i \quad \text{or} \quad \hat{S}_j = \sum_{i \in D_j} S_i.$$

  The first definition is useful when the data have to be stored at a single site, while the second when the data of a task

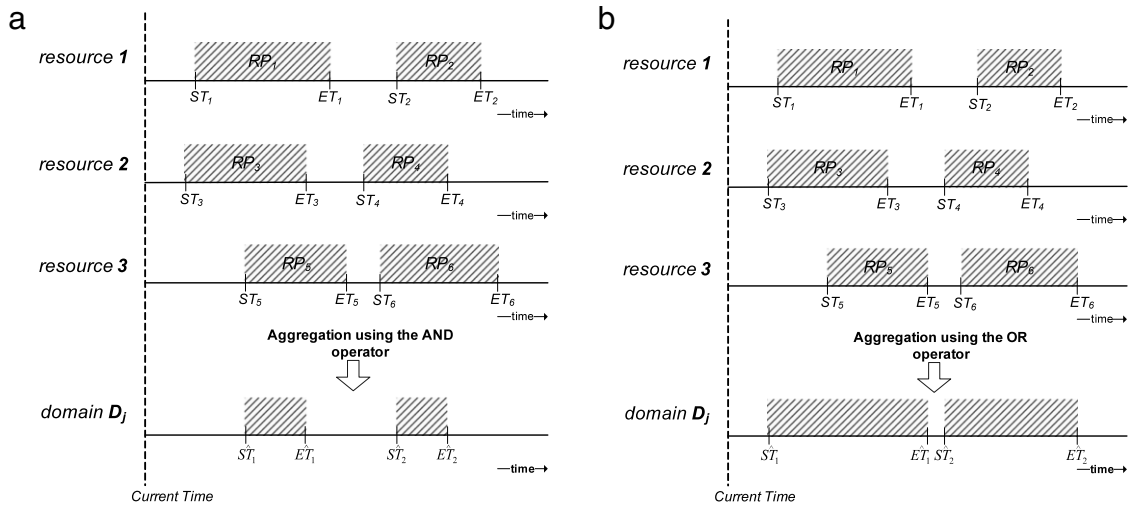

Fig. 2. The aggregation of reservation duration information (availability), where (a) we use the AND Boolean operator to calculate the time periods during which all the sites in domain D_j are reserved, (b) we use the OR Boolean operator to calculate the time periods during which at least on site in the domain is reserved.

can be stored in a distributed way across the domain. The averaging operation could also be used, giving an indication of the domain's site storage capacities.

- The available (free) storage capacities FS_i of the sites, measured in MB, in a domain D_j can similarly be aggregated using a maximization or an additive operation:

$$\hat{FS}_j = \max_{i \in D_j} FS_i \quad \text{or} \quad \hat{FS}_j = \sum_{i \in D_j} FS_i.$$

An averaging operation could also used for the free storage capacities.

- The number of tasks N_i already assigned to the sites can be aggregated over a domain D_j using an additive operation:

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

Other operators can also be used, such as the maximum representative or the average number of tasks in the sites of the domain.

- Some tasks require for their execution a number of datasets stored at specific sites. If a dataset k exists in site i, we set $I_{ik} = 1$; otherwise, we set $I_{ik} = 0$. This parameter is aggregated over all sites in a domain D_j using a Boolean OR operator:

$$\hat{I}_{jk} = \underset{i \in D_j}{OR}\{I_{ik}\}.$$

Thus, $\hat{I}_{jk} = 1$ means that there is at least one site in domain D_j that holds dataset k.

- The estimated time FT_i in the future when a computational 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 sites' reservation duration information (equivalently, their time availability) can also be aggregated; reservation periods are defined through the *start times* (ST) and *end times* (ET). In this case aggregation cannot be performed using operators like min, max, Σ on the reservation periods, since their aggregated result would have little meaning for the scheduling algorithms. Instead Boolean operators can be used in order to find the time periods where, e.g., all the sites in the domain are free, or at least one of them is free, etc.

In Fig. 2 we consider three sites, S_1, S_2 and S_3, belonging to a domain D_j, and several reservation periods RP_m, $m = 1, 2, \ldots, 6$, that are already scheduled with their corresponding start and end times. In Fig. 2(a) we aggregate the reservation durations information of the sites for the whole domain D_j, using the AND Boolean operator, so as to obtain the time periods [intervals $(\hat{ST}_1, \hat{ET}_1)$ and $(\hat{ST}_2, \hat{ET}_2)$ in Fig. 2(a)] during which all the sites in the domain are reserved. This means that during the remaining time periods, there is at least one resource that is idle and available (e.g., for reservation or for executing new tasks). This information may be useful for schedulers performing timed and advance resource reservations; the scheduler will avoid sending a reservation request with specific start and end time requirements to a domain D_j whose sites are all occupied during the corresponding period. In Fig. 2(b) we aggregate the reservation durations information of all the sites of the domain D_j, using the OR Boolean operator, so as to obtain the time periods [intervals and in Fig. 2(b)] during which at least one site is reserved. This means that during the remaining time periods all the sites in the domain are free. It is evident, that the aggregation definition that uses the AND operator is more permissive than the one using the OR operator, meaning that it reports more free periods of time for the domain. With the second approach (OR operator), it is possible that a reservation request is blocked even though there are resources that can serve it.

Another issue regarding the sites' reservation periods is how this information can be efficiently recorded and the aggregation operation be efficiently performed. Having several sites in a domain with many reservation periods (STs and FTs) makes it difficult to perform the operations described in Fig. 2. A known approach for handling time is its discretization in constant slots, and the introduction of an indicator variable for each time slot, which is zero or one depending on whether the resource is free or not, for the corresponding time period. This is the approach also followed in [32] where the network resource's availability is captured through discretized availability information vectors, so as to perform efficient resource reservation using exact information. In particular, the algorithm calculates the availability information vectors of paths using the corresponding vectors of the constituent links. We can record the reservation duration information, or, equivalently, the time availability A_i of a site i, as a binary array:

$$A_i[k] = \left\{ \begin{array}{c} 1, \text{ if a } ST, FT \text{ period exists so that} \\ (k-1) \cdot s \geq ST \;\&\; k \cdot s \leq FT \\ 0, \text{ otherwise} \end{array} \right\},$$

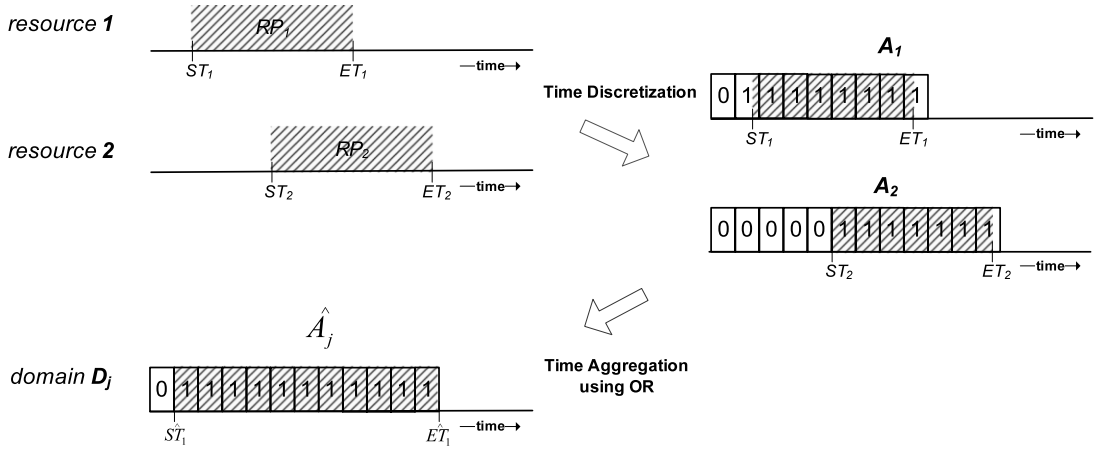

Fig. 3. Discretizing the availability information of two computational resources and then performing time aggregation using the OR operator.

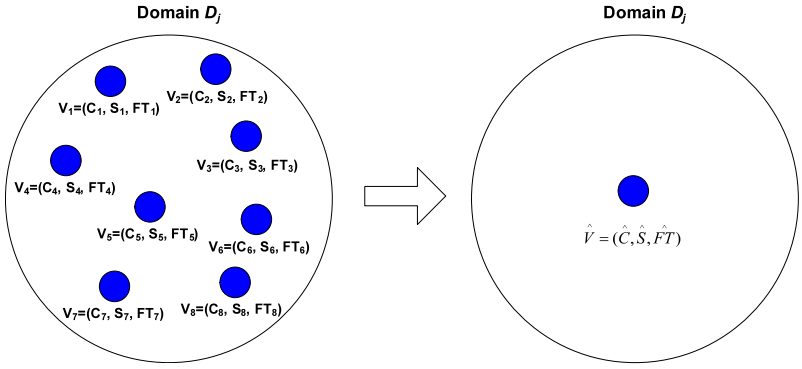

Fig. 4. The single point aggregation technique, where the site vectors $V_1, V_2, \ldots, V_8$ of the eight sites of domain D_j are aggregated into a single vector. After aggregation the domain D_j is described by a single vector $\hat{V}$.

whose size depends on the discretization step s. The time availability array of a site is included in the site's information vector. Fig. 3 presents an example of first discretizing the time availability information of two computational resources and then performing time aggregation using the OR operator. The Boolean operations are performed between the corresponding discrete time periods. The aggregated availability of a domain D_j is denoted by $\hat{A}_j$.

The discretization of the time axis in steps of duration s results in some loss of information (unless all durations are multiples of s). The choice of the discretization step s introduces a tradeoff between the amount of reservation information transferred, processed and stored, and the accuracy of this information. A small discretization step results in large vectors for the time availability of the sites or domains, but provides good accuracy of the time information represented. The opposite holds when the discretization step is large. Also, in this study we assume that all sites use the same discretization step, otherwise prior to the aggregation, the availability vectors should be homogenized. In addition, in order for time discretization and aggregation to operate efficiently it is necessary that all sites are time synchronized (e.g., using the Network Time Protocol, NTP) and that along with the availability vectors, the sites also advertise the exact reference times they use. Synchronization requirements of network equipment operating in a packet network, dealing with the distribution of time and frequency over a network of clocks, have been set by ITU-T [33].

In any case, the list of parameters and aggregation operators defined above is only indicative of the different options that can be used by the aggregation schemes. Other parameters and operators can also be defined, depending on the needs of the applications and the scheduling algorithms used.

4.3. Aggregation schemes

In this subsection we present aggregation techniques for reducing the number of vectors in the information matrix M_j of a given domain D_j.

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. Fig. 4 shows the application of the single point aggregation technique, where the information matrix M_j that has $|D_j|$ rows is reduced to an aggregated information matrix $\hat{M}_j$ that has only one row:

$$M_j = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_8 \end{bmatrix} = \begin{bmatrix} (C_1, S_1, \ldots) \\ (C_2, S_2, \ldots) \\ \vdots \\ (C_8, S_8, \ldots) \end{bmatrix}$$

$$\Rightarrow \quad \hat{M}_j = \left[\hat{V}\right] = \left[(\hat{C}, \hat{S}, \ldots)\right].$$

The information transferred to the higher levels is greatly reduced using this aggregation technique, but this happens at the expense of a degradation in the quality of the aggregated information and in the value it has for the resource scheduler.

4.3.2. Intra-domain clustering aggregation scheme

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

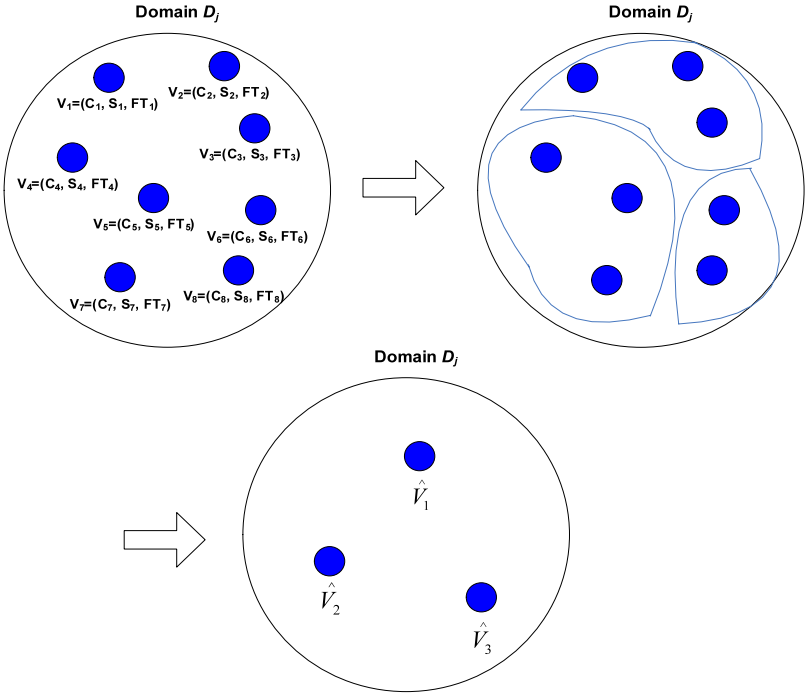

Fig. 5. The intra-domain clustering aggregation technique, where all site vectors belonging to an intra-domain cluster are aggregated into a single vector. The size of the information matrix M_j is reduced from $|D_j| = 8$ vectors to $h_j = 3$ vectors in this example.

information vectors $\hat{V}_l$, $l = 1, 2, \ldots, h_j$, of the clusters, is sent to the higher levels. Various approaches can be used for clustering the sites of a domain:

- An appropriate clustering function can be applied to each site's information vector and the sites that yield closer values are grouped together. The objective is to obtain intra-domain clusters consisting of sites with similar characteristics so that the aggregated information vector better represents the sites in the intra-domain cluster. A special case of this is to define the clusters based on a particular property or feature some sites have.
- For Grids that support timed and advance reservations, the clustering can be performed so as to maximize the time periods during which the sites belonging to a given cluster are (un)available (as indicated by their availability vector A_i). In this way the aggregated availability vector $\hat{A}$ of a cluster will better describe the actual availability of the sites in that cluster. In [34] a scheduling method is presented that increases the time overlapping of the tasks (or reservation requests) 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.
- The sites can be clustered randomly or based on other known clustering approaches.

Fig. 5 shows the application of the intra-domain clustering aggregation technique, where $h_j = 3$ clusters are created in the given domain D_j.

$$M_j = \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_7 \\ V_8 \end{bmatrix} = \begin{bmatrix} (C_1, S_1, \ldots) \\ (C_2, S_2, \ldots) \\ \vdots \\ (C_7, S_7, \ldots) \\ (C_8, S_8, \ldots) \end{bmatrix}$$

$$\Rightarrow \quad \hat{M}_j = \begin{bmatrix} \hat{V}_1 \\ \hat{V}_2 \\ \hat{V}_3 \end{bmatrix} = \begin{bmatrix} (\hat{C}_1, \hat{S}_1, \ldots) \\ (\hat{C}_2, \hat{S}_2, \ldots) \\ (\hat{C}_3, \hat{S}_3, \ldots) \end{bmatrix}.$$

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.

4.3.3. Reducing aggregated information using domination relations

In this subsection we introduce the concept of *dominated resources* to further prune the number of information vectors processed by the domain monitors or the number of aggregated information vectors processed by the central monitor. In particular, we say that the information vector V_1 *dominates* information vector V_2, if V_1 is better than V_2 with respect to all the cost parameters. The term "better" is interpreted differently based on the parameters of interest.

For the sake of being specific, 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:

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

Information vector V_2 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: it has smaller computational and storage capacity and larger future time at which the resource will be available for executing a new task. We should also note that in the case of the time availability parameter (Section 4.2), a site A is better than a site B, if A is available, for serving reservation requests or tasks, for larger periods of time than B.

The domination relation assumes that all tasks generated have the same notion about what is "good" and what is "bad" in terms of resource characteristics. In other words, and more formally, the only requirement is that the optimization function (to be discussed in Section 4.4) that selects a resource for a task, based on the parameters aggregated, is monotonic (increasing or decreasing, it does not matter) with respect to the parameters involved. For example, it would be reasonable to expect that tasks prefer resources with large computational and storage capacity, and large availability. In this case the application of such domination relations is as described, without any assumptions about how

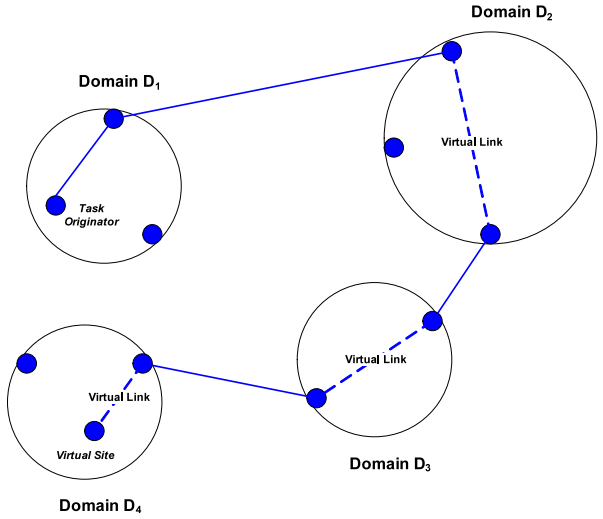

Fig. 6. The border node–site pairs aggregation scheme, considering network, computational and storage resources.

tasks are scheduled. Even if a class of tasks prefer, say, resources with small storage capacity (for whatever reason), this case could also be treated by defining differently the optimization criterion to be used (which in this case would been the minimization of storage capacity as opposed to maximization), and consequently the domination relation should be modified.

Also, the domination relations can either be applied in the vectors of a domain without any further processing, leading to a standalone aggregation technique, or they can be applied along with the single point and intra-domain aggregation techniques presented earlier.

4.3.4. Border node–site pairs scheme

In this subsection we consider the integration of network and grid information aggregation schemes. In general, there are several differences between these schemes: The parameters of interest are different (e.g., link delay versus computational capacity). Also, in the network aggregation schemes each domain is represented by a number of virtual links connecting physical border, ingress and egress, nodes. In this way the transition matrix of the domain is created, which contains the aggregated costs of traversing the corresponding border node pairs [20]. On the other hand, in the grid information aggregation schemes, virtual links between resource and border nodes have also to be created, since a domain can be the final destination of a task, where it is executed.

The *border node–site pairs aggregation scheme*, presented next, incorporates both network (as in [19,20]) and resource (as in this work) aggregation techniques. In this scheme, we first perform resource information aggregation in the Grid, using one of the techniques presented in Sections 4.3.1–4.3.3. Next, using concepts similar to [19,20], we record virtual links between border nodes and virtual sites (such a virtual link is presented in D_4 of Fig. 6). In the case of the single point aggregation technique, there will be only one virtual site per domain, with virtual links connecting each virtual site to border nodes. In the case of the intra-domain clustering aggregation technique, there will be one virtual site per intra-domain cluster, with virtual links connecting each virtual site to border nodes. Each border node–site pair is characterized by a vector consisting of the virtual site's aggregated resource characteristics and the virtual link's network related parameters. For example, a border node–site pair can be characterized by the following vector:

$$\hat{V}_{u,v} = (\hat{C}_v, \hat{S}_v, \ldots, \hat{d}_{u,v}, \hat{b}_{u,v}),$$

where $\hat{d}_{u,v}$ is the delay of the virtual link (u, v) and $\hat{b}_{u,v}$ is its bandwidth. The delay $\hat{d}_{u,v}$ of the virtual link can be defined as the maximum delay of any path connecting border site u to any site v in the domain (or any site v in the corresponding intra-domain cluster), while its bandwidth $\hat{b}_{u,v}$ can be defined as the minimum bandwidth of any path connecting the border node u to any site v of the domain (or any site v in the corresponding intra-domain cluster).

In this way every domain is characterized by two matrices: Matrix $\hat{M}$ contains network information on the virtual links connecting the border nodes of the domain with its virtual sites. It also contains resource information (computational and storage capacity, etc.) on these virtual sites, representing the aggregated resources in the domain. Matrix $\hat{B}$ contains network information (delay and the bandwidth) on the virtual links connecting the border nodes of that domain.

$$\hat{M} = \left[\hat{V}_{u,v}\right] = \left[(\hat{C}_u, \hat{S}_u, \ldots, \hat{d}_{u,v}, \hat{B}_{u,v})\right],$$

and

$$\hat{B} = \left[(\hat{d}_{u,u'}, \hat{B}_{u,u'})\right].$$

Each candidate path–domain pair is viewed as a sequence of consecutive border–border node pairs until the final pair consisting of a border node–site pair, belonging to the domain where the task will be executed. Fig. 6 shows the route followed by a task and its accompanying data from the task's originating user to the virtual site where the task will be executed. For the intermediate domains D_2 and D_3, border–border node pairs are selected, while for the final destination domain D_4 a border node–site pair is selected.

4.4. Domain selection optimization functions

Upon the arrival of a new task, the central scheduler *CS* selects a proper domain D_j for its execution, and forwards the task request to the corresponding domain scheduler DS_j who assigns it to a specific site in that domain. The *CS* uses aggregated information collected by the central monitor *CM*, while DS_j uses exact resource information in order to make its corresponding scheduling decisions. The *CS* performs the following operations in order to select the appropriate domain for a task's execution (similar are the operations performed by a *DS* to select the appropriate site for a task's execution):

1. It discards all aggregated information vectors that do not satisfy the task requirements. For example, if the task T_m at hand requires more storage, or if its starting time is earlier that the finish time $\hat{FT}$ specified in an aggregated information vector $\hat{V}$, then this vector is discarded.
2. An optimization function is applied to the remaining vectors $\hat{V}$ and the domain giving the optimum value (minimum or maximum, depending on the function's definition) is selected.

A number of different optimization functions can be applied in the second step. For example, consider the aggregated information vector

$$\hat{V}_j = (\hat{C}_j, \hat{S}_j, \hat{N}_j) = (1000, 200, 5)$$

belonging to domain D_j, which was created using the single point aggregation scheme.

Using the optimization function

$$f(V) = \frac{N}{C \cdot S},$$

we favor domains with large computational and storage capacities, and small numbers of already assigned tasks, getting:

$$f(\hat{V}_j) = \frac{5}{1000 \cdot 200}.$$

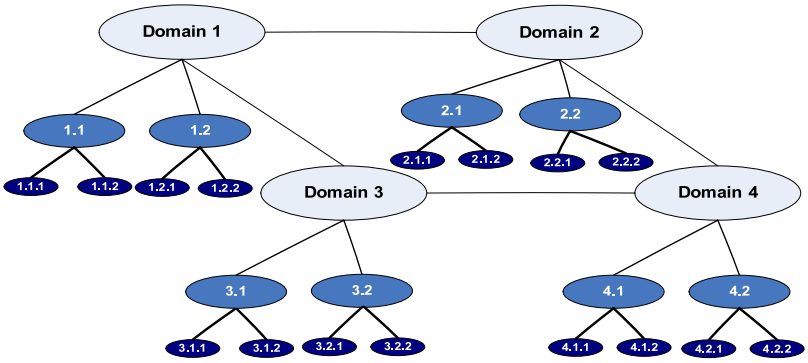

Fig. 7. A multi-level hierarchical network.

This value is then compared to the values produced by the other candidate aggregated information vectors. Note that for different users (or VOs) we can use different optimization functions for selecting the domain or site where tasks will be executed based on their quality of service requirements.

In addition if the optimization function used is monotonic in each of its parameters, it implicitly defines a domination relationship (see Section 4.3.3) that helps reduce the amount of information kept after aggregation. For example, if the information vector is $V_j = (C_j, S_j, FT_j)$ and $f(V_j)$ is an increasing function of the capacity C_j and the storage S_j and a decreasing function of the finish time FT_j, then we implicitly obtain the domination relationship given in Section 4.3.3, since there is no reason to keep information about sites (or domains) that are inferior than others with respect to *all* parameters of interest (because such sites are not going to be optimal for the given optimization function and, thus, they will not be selected). If different and contradictory optimization functions, however, are used for selecting the site on which different tasks will be executed, then a domination relationship is not defined. For example, if for some tasks we want to use a policy (equivalently, optimization function) that selects the site with the minimum finish time (FT), while for some other tasks we prefer to use a policy that selects the site with the maximum finish time (FT), then no site can be dominated and eliminated from further consideration due to its FT parameter. In this paper we do not consider contradictory optimization functions, assuming that is natural to want to minimize the delay incurred for serving a task, and therefore selecting a site with a small FT parameter, or equivalently, using an optimization function that is decreasing with the finish time.

5. An example of the applicability of the grid resource information aggregation

The proposed aggregation schemes have been presented assuming a simple two-level hierarchy (Fig. 1); however, they can also be applied in multi-level hierarchical Grid Networks, organized in domains and subdomains, like the one presented in Fig. 7. In such a network, the monitoring and scheduling process is performed in a distributed manner, and resource-related information is advertised from sub-subdomains (e.g., $D_{4.1.1}$ and $D_{4.1.2}$) to subdomains ($D_{4.1}$) where it is aggregated. Subdomains (e.g., $D_{4.1}$ and $D_{4.2}$) publish their aggregated information to domains (D_4), where again it is aggregated. Finally, domains advertise their aggregated information to each other. This is similar to the way routing information is collected in a hierarchical data network.

When a task is created in a domain (say, D_3), the corresponding domain scheduler (DS_3) uses the domain information matrices ($\hat{M}_1, \hat{M}_2, \hat{M}_3, \hat{M}_4$) to select a domain for the task's execution. Using an optimization function, as described in Section 4.4, the domain scheduler selects the domain (say, D_1) where the task will be executed and forwards it to the corresponding domain scheduler (DS_1). The domain scheduler then chooses in a similar way one of its subdomains (e.g., $D_{1.2}$) and forwards the task to the corresponding subdomain scheduler ($DS_{1.2}$). This procedure is repeated until the task reaches a site, and is reminiscent of the way packet routing is performed in hierarchical data networks. Also, we should note that it is possible, due to the aggregation of the resources information, for a task to reach a subdomain where the corresponding scheduler finds it impossible to schedule it at the lower levels of the hierarchy. This can be the case when the task's hard requirements cannot be satisfied. In this case the task can either be dropped, or it can backtrack up the hierarchy, searching for a different candidate subdomain.

Referring again to Fig. 7, we proceed with an example of information aggregation, to indicate that this approach can lead to significant information exchange reduction in the network. Specifically, we assume that each site i is characterized by an information vector, which records (among others) the availability of the data 1 in the site (as indicated by I_1) and that the OR Boolean operator is used for the aggregation of this information. If sub-subdomain $D_{1.2.1}$ "loses" for some reason dataset 1, resulting in $I_1 = 0$, while for subdomain $D_{1.2.2} I_1 = 1$, then when this information is propagated to domain monitor $DM_{1.2}$, the corresponding information vector $\hat{V}_{1.2}$ does not change ($I_1 = 1$) and, consequently, it does not need to be propagated to the network. In the performance evaluation section that follows, we measure the frequency of information vector changes/updates that result from the changes in the corresponding resources' dynamic characteristics.

In this paper we assume that the resources in all domains use the same information vectors, the same operators for their aggregation, and the same optimization metric for selecting, first, the domain (using aggregated information) and, next, the resource in that domain (using exact information). In a multi-domain environment (like the multi-level hierarchical network presented in Fig. 7), these assumptions do not always hold, and the resources in each domain (cluster, organization, virtual organization, etc.) may use different information parameters, or different operators for their aggregation and/or apply different optimization metrics for the selection (using exact information) of a site in the domain. For example, consider the scenario where the computational capacities of the resources in a domain are aggregated using the addition operator, while in another domain they are aggregated using the max operator. In this case, the scheduler will use an optimization metric for deciding the domain where a task will be executed, even though the aggregated information of the various domains is of a different type (sum or max). Aggregation can still be performed in this case (using the same methods, e.g., single point, as the ones proposed here), but with less efficiency. This shows the

need for open and broadly acceptable standards that would agree on the parameters, the operators and methods to be used. It is also possible to add another level of hierarchy (e.g., in Fig. 1), clustering together domains that follow the same aggregation approach (i.e., using the same standard). Another approach (see the related Fig. 7) would be to use the same aggregation operators, parameters, methods for subdomains belonging to the same hierarchical level (e.g., 5.1.x), of the same or not main domain (e.g., 5). In any case, we expect, that this diversity (of parameters, operators, methods) will degrade the quality of the aggregated information (in terms of the efficiency of the scheduling decisions taken) in comparison to the case (assumed in the paper) where there is uniformity in the aggregation operators, parameters and methods used. This is also supported by the performance results to be given in the next section.

6. Performance evaluation

We performed simulations in order to evaluate the proposed aggregation operators and techniques, and to examine the effects of information aggregation on the quality of the scheduling decisions

6.1. Simulation environment

In the experiments we did not use an existing simulator but created a new simple simulation environment, tailored to the evaluation needs for the aggregation schemes. The simulation framework attempts to abstract the characteristics of any distributed computation environment. In this way, we concentrate on the important issues of aggregation, without having to deal with the full complexity of a Grid system that would involve many more parameters and would obscure the picture. Through this simulation environment we create several computational resources, with different characteristics that belong to different domains. We assume that there is a central scheduler for all the Grid and local schedulers for each domain. After defining the exact characteristics of the Grid Network, a large number of tasks with varying characteristics are created and are submitted to the central scheduler. The rest of the operations performed by the simulator are the same as those described in Section 4.1.

In the proposed aggregation schemes the information considered for aggregation can be any set of parameters and aggregation operators. For the simulations, however, we had to choose only a few representative parameters so that the results are easier to present and interpret. The three parameters (computational capacity, number of tasks queued and the time availability of the resources) chosen for the simulations are representative of other Grid related parameters. The computation capacity is a static characteristic and is similar in nature to other static parameters, such as the number of CPUs of a resource, its storage capacity, its cost of use, etc. The number of tasks queued is a dynamic characteristic, and is similar to other dynamic parameters, such as the percentage of CPU used, the free storage capacity and other. The time availability is also a dynamic characteristic that is, however, more difficult to deal with and perhaps more interesting than the other two parameters. This parameter is important in data and Grid Networks when performing in advance reservation of resources; a large research field where aggregation can have many applications.

We consider a number of sites that are randomly grouped into domains, each having an approximately equal number of sites. Site i is characterized by its computational capacity C_i, measured in MIPS and chosen from a uniform distribution UC. In our simulations, tasks (or reservation requests) are created with exponentially distributed interarrival times with mean I, while their workload (or duration) follows a uniform distribution UW. A task's execution time and reservation duration of the corresponding resource, depend on the resource's computational capacity and the task's workload. Also, in this paper, each reservation request is made for the whole resource and it has a certain duration specification, independently of the resource it is assigned on. The tasks (or reservation requests) are submitted to the central scheduler that makes its decisions using either complete or aggregated resource information. In the simulation results we do not consider the effects of the information propagation delay, assuming that resource information (aggregated or not) is not outdated by the time it is used by the scheduler due to the delay between measuring some parameter and the moment the measured value is available to the scheduler. However, we believe that in practice the aggregation operation can reduce the negative effects of this delay, since summarized information is usually less affected by the change in the values of the parameters measured. Also, in each scenario examined, we alter a number of parameters, such as the number of domains, the number of sites, the minimum or maximum values of the uniform distributions, and we average the measured values over a number of simulation rounds using independent random seeds. Network related issues are not considered in these simulation experiments.

Users generate either tasks or reservation requests. We consider two cases respectively, regarding the sites' information vectors:

- The information vector V_i of site i contains its computational capacity C_i and the number of tasks N_i queued at it:

  $V_i = \{C_i, N_i\}$.

  In case no aggregation is used a new task T_m is assigned to the site i that minimizes

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

  In case where aggregation is used then the central scheduler CS uses only the aggregated domain information vectors, and assigns task T_m to the domain D_j that minimizes

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

  Next, 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 is again performed based on the same optimization function:

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

- The information vector V_i of site i contains only its time availability array A_i. In case no aggregation is used a reservation request *(ST, FT)* is assigned to first the site $i$found, which can satisfy it. By the term "satisfy" we mean that the resource is free at the corresponding time period. In case where aggregation is performed then the central scheduler CS assigns, using only the aggregated domain information vectors, a reservation request *(ST, FT)* to the domain D_j that can satisfy it. Next, the selected domain's scheduler DS_j, receives the request, and, having complete knowledge of the information vectors of all the sites in the domain, assigns it to any domain site that can satisfy it.

6.2. Aggregation schemes evaluated

We implemented and evaluated the performance of the following schemes:

- *FlatCpuTasks*: In this scheme no information aggregation is performed of the sites' computational capacity and number of tasks parameters.

- *MinCpuSumTasks*: In this scheme the information vectors of the sites belonging to the same domain are aggregated (single point aggregation—Section 4.3.1) using the minimum representative and the additive operators, respectively:

  $$\hat{C} = \min_i C_i \quad \text{and} \quad \hat{N} = \sum_i N_i.$$

- *MinCpuMaxTasks*: This scheme is similar to the *MinCpuSumTasks*, except that the maximum representative operator is used for aggregating the number of tasks in the sites of a domain:

  $$\hat{N} = \max_i N_i.$$

- *MinCpuAvgTasks*: This scheme is similar to the *MinCpuSumTasks*, except that the average number of tasks in the sites of the domain, is used as the domain's aggregated value for the corresponding property:

  $$\hat{N} = \text{avg}_i N_i.$$

- *DomMinCpuSumTasks*: This scheme is similar to the *MinCpuSumTasks*, except that domination relations (Section 4.3.3) are applied to the vectors of the sites of a domain, before they are aggregated using the single point aggregation scheme.
- *ICMinCpuSumTasks*: This scheme is similar to the *MinCpuSumTasks*, except that the intra-domain clustering aggregation scheme (Section 4.3.2) is applied, instead of the single point one, where sites are randomly clustered into intra-domain clusters.
- *FlatTime*: In this scheme no information aggregation is performed on the sites' time availability parameter.
- *AndTime*: In this scheme the sites' time availability arrays are aggregated using the AND Boolean operator:

  $$\hat{A} = \underset{i}{AND}(A_i).$$

- *OrTime*: In this scheme the time availability arrays are aggregated using the OR operator:

  $$\hat{A} = \underset{i}{OR}(A_i).$$

6.3. Performance metrics

We are interested in evaluating the degree to which the information produced by the proposed aggregation schemes leads to efficient scheduling decisions, while the size and the frequency of the information updates is kept low.

For the evaluation of the MinCpuSumTasks, MinCpuMaxTasks, MinCpuAvgTasks, DominanceMinCpuSumTasks, and ICMinCpuSumTasks aggregation schemes we use the *Stretch Factor (SF)*, as the metric that measures the scheduling efficiency and in practice the quality of the aggregated information. The Stretch Factor is defined as the ratio of the task delay D when the task is scheduled using complete resource information (FlatCpuTasks) over the task delay when an aggregation scheme is used (MinCpuSumTasks, MinCpuMaxTasks, MinCpuAvgTasks, DominanceMinCpuSumTasks, or ICMinCpuSumTasks). The task delay is defined as the time that elapses from the task's submission to the Grid until the completion of its execution at a site. A stretch factor metric is also encountered in the hierarchical networks related literature [4], where it is defined as the ratio of the average number of hops (or average delay) between a source and a destination when flat routing is used over the corresponding value when hierarchical routing is used. The *Stretch Factor (SF)* metrics we use in this paper are calculated using the following expression:

$$SF_{\text{aggregation\_scheme}} = \frac{D_{\text{FlatCpuTaks}}}{D_{\text{aggregation\_scheme}}}.$$

In all cases $SF \leq 1$, since when a scheduler has complete resource 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.

For the evaluation of the AndTime and OrTime aggregation schemes, we use the number of reservation requests that were not scheduled (blocked), as the metric that characterizes the quality of the aggregated information. The measured values are of course compared with the ones produced in the case where no aggregation is performed (FlatTime). A reservation request is blocked when no free time space is found either because there is no available resource or because the aggregation scheme led to the selection of an overused domain.

In the experiments, for all the aggregation schemes, we also measure the frequency of changes/updates caused to the aggregated information vectors (of the domains) due to the scheduling of new tasks. This metric is measured as the number of aggregated information vector changes over the number of tasks scheduled; its value is always ≤ 1, where a value close to 1 means that the aggregated information vectors change frequently. Since the evaluated aggregation schemes (Section 6.2) consider only one dynamic parameter (either the number of tasks scheduled in each site or the site's time availability), the information vector of a site changes only if a new task or reservation request is scheduled on it. In general, we expect that by using aggregated resource information, the changes in the dynamic characteristics of the resources will not always propagate to the central monitoring system, since aggregated information vectors will sometimes absorb these changes. In the experiments, we do not directly count the number of information vector changes due to tasks completing execution (or reservation); these are counted indirectly when a new task is scheduled and the information vectors are checked for changes. An additional metric used for the evaluation of the aggregation schemes is the number of information vectors produced and sent to the central scheduler in order to make its decisions. In this case, we assume that the central scheduler has no a priori knowledge of the sites/domains information and all the corresponding vectors (changed or not) are sent to it.

6.4. Aggregating computational capacity and number of tasks

6.4.1. Aggregation schemes

Fig. 8 presents the *Stretch Factor (SF)* for the MinCpuSumTasks, the DomMinCpuSumTasks and the ICMinCpuSumTasks aggregation schemes when 1000 Grid sites are clustered in a variable number of domains and 25 000 tasks are created and scheduled. The sites' computational capacity and the tasks' workload follow uniform distributions (UC min/max = 10/10000 MIPS and UW min/max 1000/10000000 MI respectively). In general, all the stretch factor metrics measured behave similarly, that is, their value first decreases up to some point, after which it starts increasing towards 1. This is because when the number of domains is small, then the number of sites per domain is quite high, increasing the probability that more than one "best" sites or sites similar to the "best" site exist in different domains. This increases the probability that a domain with such a site will be chosen, even if aggregation is used. Next, as the number of domains increases, this probability 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.

Comparing the different aggregation policies we observe that the ICMinCpuSumTasks produces the best results, followed by the DomMinCpuSumTasks and the MinCpuSumTasks aggregation policies. The ICMinCpuSumTasks aggregation scheme uses $h = 5$ intra-clusters in each domain and its use leads to better scheduling decisions (as measured by the corresponding stretch factor), however this comes at the cost of increased number of information

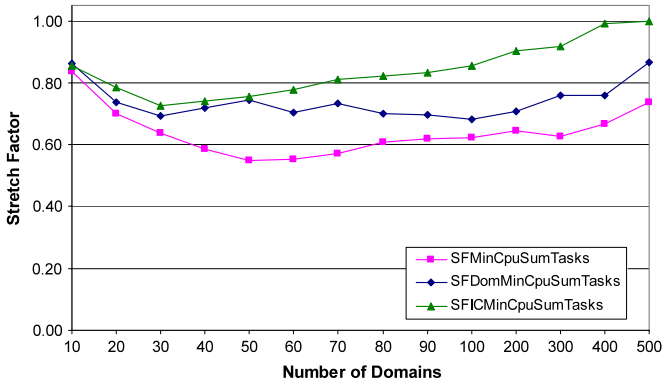

Fig. 8. The *Stretch Factor (SF)* for the MinCpuSumTasks, the DomMinCpuSumTasks and the ICMinCpuSumTasks aggregation schemes, when 1000 Grid sites are clustered in a variable number of domains and 25000 tasks are created and scheduled.

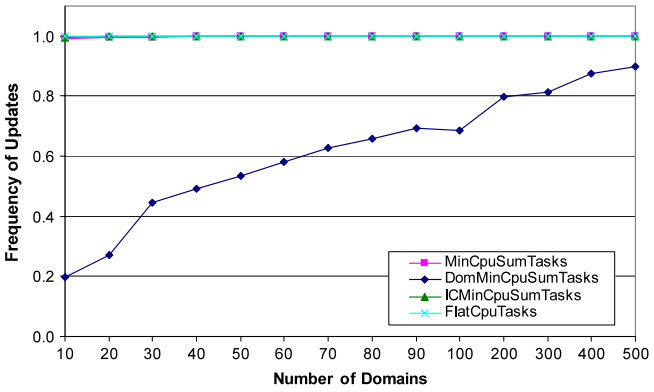

Fig. 9. The frequency of information vector updates for the FlatCpuTasks, the MinCpuSumTasks, the DomMinCpuSumTasks and the ICMinCpuSumTasks aggregation algorithms, when 1000 Grid sites are clustered in a variable number of domains and 25 000 tasks are created and scheduled. The lines corresponding to MinCpuSumTasks, ICMinCpuSumTasks and FlatCpuTasks are indistinguishable, since they fall on top of each other.

Table 1
The number of information vectors produced by each aggregation scheme for the central monitor (Fig. 1), when $N = 1000$ sites are clustered in $L = 100$ domains.

Aggregation scheme	# of information vectors
FlatCpuTasks	$N = 1000$
MinCpuSumTasks	$L = 100$
DomMinCpuSumTasks	$L = 100$
ICMinCpuSumTasks ($h = 5$ intra-domain clusters)	$L \cdot h = 500$

vectors advertised (Table 1). 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). In addition, it seems that the domination operation, which discards dominated information vectors, improves the quality of the information provided to the scheduler. Also, we observe that the average task delay results, using the MinCpuSumTasks, the DomMinCpuSumTasks and the ICMinCpuSumTasks aggregation algorithms, are in accordance with the results presented in Fig. 8, that is they are essentially each other's inverse. A large task delay indicates that the information produced by the corresponding aggregation scheme, leads to wrong task scheduling decisions. Table 1 shows the number of information vectors provided by each scheme when 1000 sites are clustered in 100 domains.

Fig. 9 illustrates the frequency with which the aggregated information vectors change. Since, the evaluated aggregation schemes (Section 6.2) consider only one dynamic parameter (that is, the number of tasks scheduled in each site), this means that the information vector of a site changes only if a new task is scheduled into it. We observe that the MinCpuSumTasks and ICMinCpuSumTasks aggregation schemes result in a large number of updates, almost equal, in most cases, to the maximum one. On the other hand using the DomMinCpuSumTasks aggregation scheme, we observe that when the number of domains is small, and as a result many sites exist in each domain, the domination operation achieves in absorbing a large percent of the changes in the sites' information vectors. As the number of domains increases and fewer sites exist in each domain, this capability declines almost linearly.

We also performed several other experiments altering the number of sites, the number of tasks created or their creation pattern. We omit presenting these results, since the conclusions and observations obtained were similar to the above.

6.4.2. Distribution of values

Fig. 10(a) shows the results obtained for the MinCpuSumTasks aggregation scheme when changing the upper and lower limits of the uniform distributions assumed for the computational capacities of the sites (*UC*) and for the tasks' workloads (*UW*). This way we investigate the effects that the uniformity of the resources (or lack of it) have on performance. The scenarios/probabilistic distributions used are presented in Table 2; in our previous experiments we have assume the *UC*4/*UW*4 scenario. In general, all the stretch factor metrics measured behave similarly, that is, their value first decreases up to some point, after which it starts increasing towards 1. This was also observed and explained previously for Fig. 8. The main observation in Fig. 10(a) is that the stretch factors measured increase along with the contraction of the *UC* and *UW* distributions. This is because as the these distributions contract, the number of possible information vectors and their differences also decrease; for example the number of different information vectors that the *UC*1/*UW*1 scenario produces is smaller than the ones produced by the *UC*2/*UW*2 scenario and so on. When the number of possible information vectors is small and their differences are also small, then the aggregation policies applied do not distort much the value of the information of the sites in the domain. Fig. 10(b) shows the frequency of updates for the MinCpuSumTasks aggregation algorithm and for the uniform distribution scenarios presented in Table 2. Again, when the number of possible information vectors is small (e.g., for the *UC*1/*UW*1 scenario) then the number of updates is also small. Corresponding experiments were performed for all the proposed aggregation schemes, producing similar results.

6.4.3. Aggregation operators

We also preformed experiments, evaluating different operators for the aggregation of the number of tasks parameter. The results illustrate the importance that the resource parameters and the aggregation operators have on the efficiency of the aggregation schemes. In our previous experiments, we were using the Sum operator (MinCpuSumTasks policy), while next we present experiments also using the Avg (MinCpuAvgTasks policy) and the Max (MinCpuMaxTasks policy) operators (Fig. 11). We observe that the MinCpuSumTasks and the MinCpuMaxTasks aggregation policies, produce the best results (in terms of the stretch factors achieved). On the other hand the MinCpuAvgTasks policy results in a smaller frequency of information vector updates than the other policies. We should note that the average number of tasks, aggregated parameter, is rounded to an integer value and this is the main reason for the small number of updates achieved. Also, the MinCpuMaxTasks policy also achieves a small frequency of information vector updates. In any case, the frequency of updates increases along with the number of domains, since in this way fewer sites exist in each domain.

Table 2
The scenarios *UC*/*UW*, correspond to different choices for the upper and lower limits of the uniform distributions assumed for the computational capacities of the sites and the tasks' workload.

Scenario	Computational capacity (min/max)	Task workload (min/max)
UC1/UW1	10/10	1000/100000
UC2/UW2	10/100	1000/1000000
UC3/UW3	10/1000	1000/10000000
UC4/UW4	10/10000	1000/100000000
UC5/UW5	10/100000	1000/1000000000

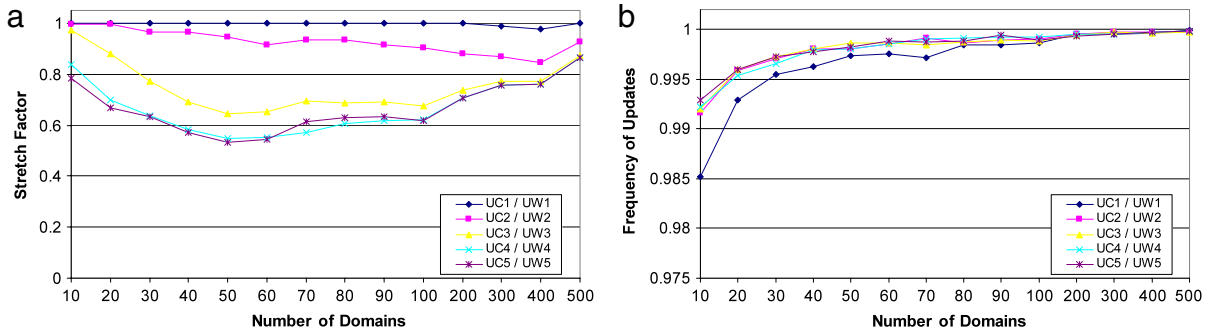

Fig. 10. The *Stretch Factor (SF)*, (b) the frequency of updates for the MinCpuSumTasks aggregation algorithm and for the uniform distribution scenarios presented in Table 2, when 1000 Grid sites are clustered in a variable number of domains and 25 000 tasks are created and scheduled. In (b) the lines corresponding to all scenarios, except UC1/UW1, are indistinguishable, since they fall on top of each other.

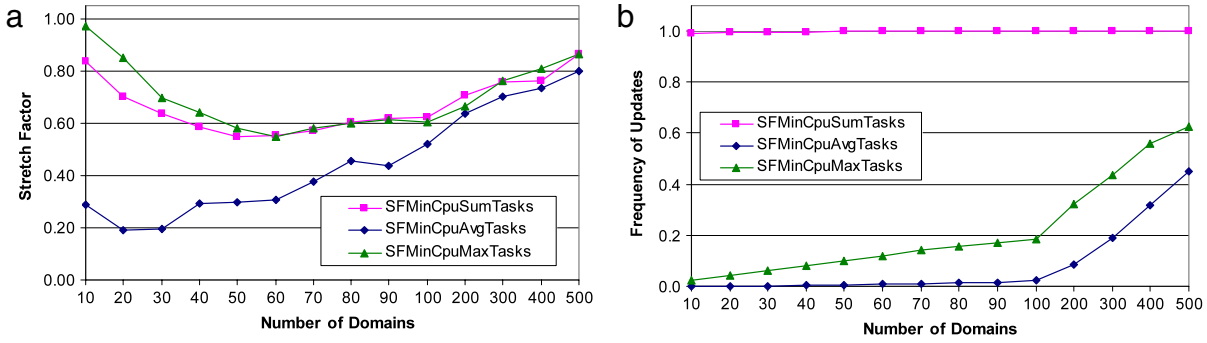

Fig. 11. (a) The *Stretch Factor (SF)*, (b) the frequency of updates for the MinCpuSumTasks, the MinCpuAvgTasks and the MinCpuMaxTasks aggregation algorithms, when 1000 Grid sites are clustered in a variable number of domains and 25 000 tasks are created and scheduled.

6.4.4. Experiments performed with real traces

In order to verify our findings we also performed experiments using real traces found in Grid Workloads Archive [35], and collected from the Grid'5000 infrastructure. Grid'5000 is an experimental Grid platform consisting of 9 sites geographically distributed in France. Each site comprises one or several clusters, for a total of 15 clusters inside Grid'5000. There are mainly two kinds of traces provided, both containing the same (for our purposes) information and referring to the same time period of about 1 year and a half, of Grid Network operation. The first contains detailed information on the tasks' execution, such as the site where the execution took place, the submission time, the queuing time, the execution time, the average CPU time used, the memory used and other parameters. The second trace file contained information on the computational resource reservations performed by the users, that is, their start and finish times. From these traces we extracted information regarding the submitted tasks, their order and their duration, which was converted to workload measured in MIPS based on a baseline computational resource. We believe this kind of information is sufficient to provide a more realistic setting for the simulations.

Fig. 12 presents the stretch factor obtained for the MinCpuSumTasks, the DomMinCpuSumTasks and the ICMinCpuSumTasks aggregation algorithms when 1000 Grid sites are clustered in a variable number of domains. We observe that our findings with real traces are similar qualitatively to those presented in Fig. 8. All the stretch factor metrics measured behave similarly, that is, their value first decreases up to some point, after which it starts increasing towards 1. The ICMinCpuSumTasks aggregation policy

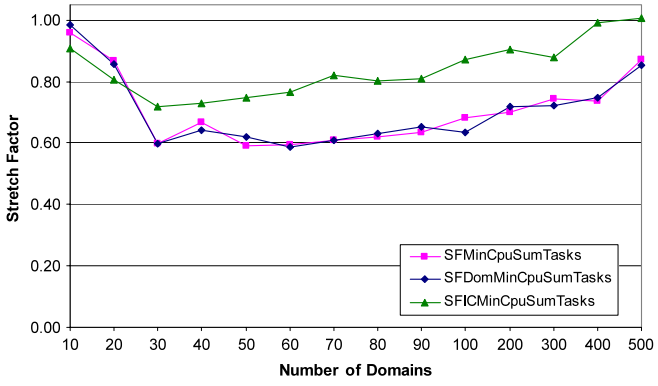

Fig. 12. The *Stretch Factor (SF)* for the MinCpuSumTasks, the DomMinCpuSumTasks and the ICMinCpuSumTasks aggregation algorithms, when 1000 Grid sites are clustered in a variable number of domains and tasks based on data collected from the Grid'5000 infrastructure are created and scheduled.

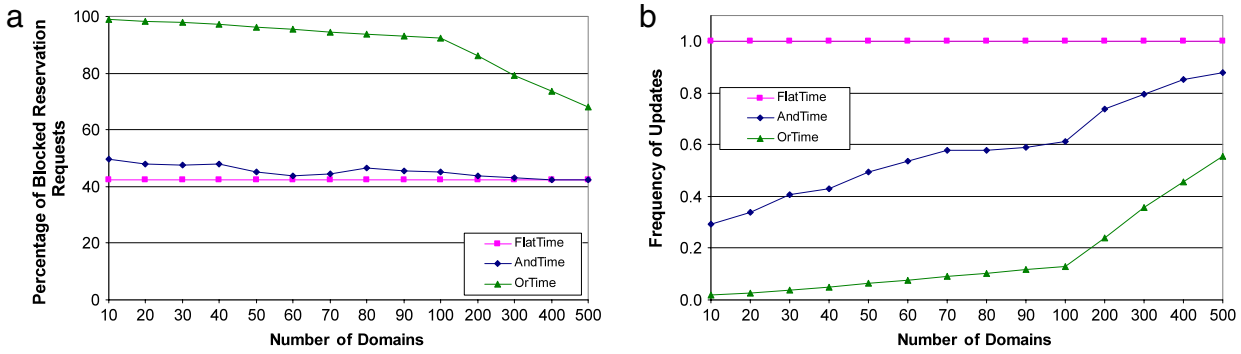

Fig. 13. (a) The percentage of blocked/failed reservation requests, (b) the frequency of updates for the FlatTime, AndTime and OrTime aggregation algorithms, when 1000 Grid sites are clustered in a variable number of domains and 25 000 reservation requests are created and scheduled.

produces the best results, while the MinCpuSumTasks and DomMinCpuSumTasks similar (probably due to the tasks' characteristics, e.g., workload distribution).

6.5. Aggregating reservation periods

Fig. 13(a) presents the percentage of blocked/failed reservation requests for the FlatTime, AndTime and OrTime aggregation schemes, when 1000 Grid sites are clustered in a variable number of domains and 25 000 reservation requests are created and scheduled. The FlatTime scheme's results, where no aggregation is actually performed, are our baseline results; all the reservation request blockings under this scheme are because no available resources were found to serve them. We observe that the AndTime aggregation scheme produces very good results, close to those achieved by the FlatTime, while the OrTime scheme exhibits a very large number of blockings. This is mainly because the OR operator, used in the aggregation of the sites' availability information, is quite restrictive, since fewer free periods of time are reported for a domain than those actually existing in its sites. As a result, requests are easily blocked even though there are available resources to serve them. As the number of domains increases and the number of sites per domain decreases, the quality of information produced by the aggregation schemes improves, and fewer requests are blocked. Fig. 13(b) presents the frequency of information vectors (and in particular of the availability arrays) updates due to the service of new reservation requests. This number is constant for the FlatTime scheme, where the set of vectors are updated every time a new request is served. Also, we observe that the OrTime scheme results in fewer updates than the AndTime scheme, mainly due to the nature of the AND and OR operators.

7. Conclusions

We proposed several techniques for resource information aggregation in Grid networks. Each site is assigned a parameter/information vector that records its computation and storage capacity, its time availability, the number of tasks queued and other parameters of interest. The information vectors of the sites belonging to a given domain are aggregated into a small number of vectors, using appropriate associative operations. We performed several simulation experiments using the Stretch Factor (*SF*) as the main metric for judging the extent to which the proposed aggregation schemes preserve the value of the information aggregated and assist the scheduler in making efficient decisions. The *SF* is defined as the ratio of the task delay incurred when scheduling based on complete resource information over that incurred when an aggregation scheme is used. We observed that in many scenarios the proposed aggregation schemes enabled efficient task scheduling decisions as indicated by the *SF* achieved, while achieving large information reduction. We looked into the frequency of information vector updates resulting from the aggregation schemes, and observed that the changes in the dynamic characteristics of the resources will not always propagate to the central monitoring system, since aggregated information vectors sometimes absorb these changes. We also observed that the uniformity of the sites' and the tasks' characteristics significantly affects the quality of the aggregated information. Finally, we performed experiments aggregating the time availability information of the sites and concluded that using the AND operator for the aggregation can be quite beneficial.

References

[1] S. Zanikolas, R. Sakellariou, A taxonomy of grid monitoring systems, Journal of Future Generation Computer Systems 21 (1) (2005) 163–188.
[2] R. Wolski, N. Spring, J. Hayes, The network weather service: a distributed resource performance forecasting service for metacomputing, Journal of Future Generation Computer Systems 15 (1999) 757–768.
[3] Z. Wang, J. Crowcroft, Quality-of-service routing for supporting multimedia applications, Journal on Selected Areas in Communications 14 (7) (1996) 1228–1234.
[4] W.C. Lee, Topology aggregation for hierarchical routing in ATM networks, Computer Communication Review 25 (2) (1995) 82–92.
[5] K. Krauter, R. Buyya, M. Maheswaran, A taxonomy and survey of grid resource management systems for distributed computing, Software: Practice and Experience 32 (2) (2002) 135–164.
[6] A. Kertesz, P. Kacsuk, A Taxonomy of Grid Resource Brokers, in: Distributed and Parallel Systems, Springer, 2007.
[7] 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, in: IPDPS, US, 2004, pp. 170–177.
[8] T. Braun, H. Siegel, N. Beck, L. Boloni, M. Maheswaran, A. Reuther, J. Robertson, M. Theys, B. Yao, D. Hensgen, R. Freund, A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, Journal of Parallel and Distributed Computing 61 (6) (2001) 810–837.
[9] Y. Cardinale, H. Casanova, An evaluation of job scheduling strategies for divisible loads on grid platforms, HPC&S, Germany, 2006.
[10] R. Buyya, M. Murshed, D. Abramson, S. Venugopal, Scheduling parameter sweep applications on global grids: a deadline and budget constrained cost-time optimisation algorithm, International Journal of Software: Practice and Experience (SPE) 35 (5) (2005) 491–512.
[11] K. Kim, R. Buyya, Fair resource sharing in hierarchical virtual organizations for global grids, in: International Conference on Grid Computing, 2007.
[12] N. Doulamis, A. Doulamis, E. Varvarigos, T. Varvarigou, Fair QoS resource management in grids, IEEE Transactions on Parallel and Distributed Systems 18 (11) (2007) 1630–1648.
[13] W. Smith, I. Foster, V. Taylor, Scheduling with advanced reservations, in: IPDPS, 2000, pp. 127–132.
[14] R. Guerin, A. Orda, Networks with advance reservations: the routing perspective, in: IEEE INFOCOM, 2000.
[15] S. Zanikolas, R. Sakellariou, A taxonomy of grid monitoring systems, Future Generation Computer Systems 21 (2005) 163–188.
[16] B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, R. Wolski, A grid monitoring architecture, in: GWDPerf-16–3, Global Grid Forum, August 2002.
[17] Enabling grids for E-science, EGEE: http://www.eu-egee.org/.
[18] L. Kleinrock, F. Kamoun, Hierarchical routing for large networks. Performance evaluation and optimization, Computer Networks 1 (3) (1977) 155–174.

[19] P. Van Mieghem, Topology information condensation in hierarchical networks, The International Journal of Computer and Telecommunications 31 (20) (1999) 2115–2137.
[20] D. Bauer, J. Daigle, I. Iliadis, P. Scotton, Topology aggregation for combined additive and restrictive metrics, Computer Networks 50 (17) (2006) 3284–3299.
[21] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva, Directed diffusion for wireless sensor networking, IEEE Transactions on Networking 11 (2003) 2–16.
[22] L. Chitnis, A. Dobra, S. Ranka, Fault tolerant aggregation in heterogeneous sensor networks, Journal of Parallel and Distributed Computing 69 (2) (2009) 210–219.
[23] R. Renesse, K. Birman, W. Vogels, Astrolabe: a robust and scalable technology for distributed system monitoring, management, and data mining, ACM Transactions on Computer Systems 21 (2) (2003) 164–206.
[24] S. Schulz, W. Blochinger, H. Hannak, Capability-aware information aggregation in Peer-to-Peer grids, Journal of Grid Computing 7 (2) (2009) 135–167.
[25] R. Zhou, K. Hwang, Trust overlay networks for global reputation aggregation in P2P grid computing, in: IPDPS, 2006.
[26] M. Cai, K. Hwang, Distributed aggregation algorithms with load-balancing for scalable grid resource monitoring, in: IEEE International Parallel and Distributed Processing Symposium, 2007.
[27] K. Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, Grid information services for distributed resource sharing, in: IEEE International Symposium on High-Performance Distributed Computing, HPDC, 2001.
[28] K. Muraki, Y. Kawasaki, Y. Mizutani, F. Ino, K. Hagihara, Grid resource monitoring and selection for rapid turnaround applications, IEICE Transactions on Information and Systems 89 (9) (2006) 2491–2501.
[29] I. Rodero, F. Guim, J. Corbalan, L. Fong, S.M. Sadjadi, Grid broker selection strategies using aggregated resource information, Future Generation Computer Systems 26 (1) (2010) 72–86.
[30] P. Kokkinos, E. Varvarigos, Resource information aggregation in hierarchical grid networks, in: IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009, pp. 268–275.
[31] gLite: http://glite.web.cern.ch/glite/.
[32] E. Varvarigos, V. Sourlas, K. Christodoulopoulos, Routing and scheduling connections in networks that support advance reservations, Computer Networks 52 (15) (2008) 2988–3006.
[33] ITU-T G.8261, Timing and synchronization aspects of packet networks, Geneva, 2006.
[34] N. Doulamis, P. Kokkinos, E. Varvarigos, Spectral clustering scheduling techniques for tasks with strict QoS requirements, in: Europar, 2008, pp. 478–488.
[35] The Grid Workloads Archive: http://gwa.ewi.tudelft.nl/pmwiki/.

P. Kokkinos received his Ph.D. in 2010 from the Computer Engineering and Informatics Department (CEID) of the University of Patras, Greece, in the field of Optical Grid Networks, focusing on task scheduling and data routing algorithms. He also holds an M.Sc. degree (2006) and a Diploma (2003) from the same department. His current research activities are in the areas of distributed computing and middleware.

E. A. Varvarigos received a Diploma in Electrical and Computer Engineering from the National Technical University of Athens in 1988, and his M.S. and Ph.D. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology in 1990 and 1992, respectively. He has held faculty positions at the University of California, Santa Barbara (1992–1998, as an Assistant and later an Associate Professor) and Delft University of Technology, the Netherlands (1998–2000, as an Associate Professor). In 2000 he became a Professor at the department of Computer Engineering and Informatics at the University of Patras, Greece, where he heads the Communication Networks Lab. He is also the Director of the Network Technologies Sector (NTS) at the Research Academic Computer Technology Institute (RA-CTI), which through its involvement in pioneering research and development projects, has a major role in the development of network technologies and telematic services in Greece. Professor Varvarigos has served in the organizing and program committees of several international conferences, primarily in the networking area, and in national committees. He has also worked as a researcher at Bell Communications Research, and has consulted with several companies in the US and in Europe. His research activities are in the areas of protocols for high-speed networks, ad hoc networks, network services, parallel and distributed computation and grid computing.