J Supercomput (2007) 42: 59–82
DOI 10.1007/s11227-006-0036-x

Dimensioning and on-line scheduling in Lambda Grids using divisible load concepts

Pieter Thysebaert · Bruno Volckaert · Marc De Leenheer · Filip De Turck · Bart Dhoedt · Piet Demeester

Published online: 10 March 2007

Abstract Due to the large amounts of data required to be processed by the typical Grid job, it is conceivable that the use of optical transport networks in Grid deployment (hence the term "Lambda Grid") will increase. The exact topology of the interconnecting network is obtained by solving a dimensioning problem, and the outcome of this strongly depends on both the expected workload characteristics and Grid scheduling policy. Solving this combined scheduling and dimensioning problem using straightforward ILP modelling is cumbersome; however, for steady-state Grid operation, Divisible Load Theory (DLT) can yield scalable formulations of this problem.

In this paper, the on-line hierarchical scheduling on a lambda Grid of workload approaching the Grid's capacity in a two-tier Grid mode of operation is studied. A number of these algorithms are goal-driven, in the sense that target per-resource goals are obtained from the off-line solution to the Divisible Load model. We compare these on-line multiresource scheduling policies for different workloads, Grid interconnection topologies and Grid parameters. We show that these algorithms perform well in the studied scenarios when compared to a fully centralized scheduling algorithm.

Keywords Lambda Grids · Optical Transport Networks · Optimization · Divisible Load

1 Introduction

The term Grid [1, 2] has been coined to refer to an aggregation of heterogeneous, possibly geographically dispersed resources of various types. These resources can be

P. Thysebaert (✉) · B. Volckaert · M. De Leenheer · F. De Turck · B. Dhoedt · P. Demeester
Department of Information Technology, Ghent University—IBBT—IMEC, Gaston Crommenlaan 8, 9050, Gent, Belgium
e-mail: pieter.thysebaert@intec.ugent.be

co-allocated to a single application; therefore, the total processing capacity of such a Grid can easily surpass the offerings of any local supercomputer or cluster.

This idea of a high-capacity wide-area distributed computing platform is very attractive to large parts of the research community dealing with highly complex computational problems involving large amounts of data to be processed. These large amounts of data associated with certain classes of Grid applications have generated an interest in optical transport networks as interconnection means between geographically separated resources participating in the same Grid, as these types of networks can sustain high bandwidths over large distances. Grids using this type of interconnection networks are often dubbed *Lambda Grids* [3, 4].

A major property of Grids is that resources of different types (e.g. computing power, network bandwidth, storage, etc.) can be co-allocated for the execution of a single application. Since Grid resources are scattered over different sites, the exact Grid workload scheduling policy not only determines computational resource usage, but also influences e.g. network load. The scheduling problem is thus entangled with the problem of deriving suitable Grid resource dimensions in an off-line fashion, based on the expected Grid's workload. Scheduling problems with predetermined resource allocations have been studied extensively (see Sect. 2); in this paper we model resource allocation interdependencies into a scalable (due to the use of divisible load theory) combined off-line dimensioning and scheduling problem and feed the results into on-line Grid scheduling policies.

We evaluate several on-line Grid workload scheduling policies; the actual workload scheduled resembles the worst case workload for which the Grid has been dimensioned. In Sect. 2, we compare our work to existing relevant research in the fields of job scheduling and optical network dimensioning. Section 3 describes in detail the application and resource models, as well as the scheduling metrics used throughout this work. A scalable off-line model for workload scheduling in Lambda Grids is presented in Sect. 4. Several on-line scheduling policies based solely on resource state information are discussed in Sect. 5; in addition, we present an alternative policy which uses information from the combined off-line dimensioning and scheduling problem in addition to resource state information.

These different scheduling policies have been compared for different scenarios; these scenarios are described thoroughly in Sect. 6, while the results are presented in Sect. 7. Finally, some concluding remarks are drawn in Sect. 8.

2 Related work

Scheduling individual jobs on a set of processing elements (e.g. a cluster or a multiprocessor machine) has been studied extensively in literature [5–7]. The quality of on-line scheduling heuristics has been analyzed and compared extensively to optimal (off-line) algorithms.

These types of scheduling problems are special instances of (Multi-Modal) Resource Constrained Project Scheduling Problems (MMRCPSP) as described in [8]. In this class of problems, a workload with fixed a priori resource requirements (for different types of resources) is scheduled on a set of such resources.

The problem of scheduling workload on a Grid (featuring a large number of jobs and resources and interdependent resource co-allocations), however, is more difficult

to capture in a scalable mathematical model. Ranganathan et al. [9] have studied independent CPU-allocation and data set replication through simulation. In a similar way, replica optimization is the main topic of the research described in [10]. That work focuses on the location of data sets, but does not address network resource allocations that may be needed to access the data within a deterministic time frame. In [11], the effects of data needing to be transferred across the network is not expressed using resource allocations but is condensed into a single *overhead* parameter describing the slowdown of a job relative to the situation where the data is locally available. A similar parameter (called *slowdown factor*) appears in [12, 13], where queueing model analysis is performed to deduce schedule quality in a purely space-shared multicomputer system.

Market-driven and incentive-based parameter sweep application scheduling on computational resources has been studied extensively by Buyya et al. [14], where resource selection policies depending on the notions of *budgets* and *deadlines* are investigated. These notions influence the amount of work performed by the Grid. In contrast, our work focuses on performing all jobs submitted to the Grid (the workload demand) in a steady-state, and dimensioning the Grid accordingly. We allow for interdependent resource allocations (e.g. allocated CPU slices and network bandwidth availability can influence each other) and use the concept of Divisible Load (see below) to keep the combined dimensioning and scheduling problem manageable.

The effects of co-allocating CPU and network resources to a single job have been studied in [15]. Grid sites are connected through a VPN in which fine-grained bandwidth pipes can be set up. In reality, it is difficult to imagine a scenario in which such pipes with guarantees concerning delay, jitter and bandwidth availability can be set up over e.g. the Internet. In contrast, the use of optical transport networks does offer the reality of high-capacity bandwidth pipes between the various Grid sites and is therefore a focal point in this paper.

In recent years, Divisible Load Theory [16–18] has been used successfully in modelling steady-state off-line Grid scheduling problems. Scheduling decisions concerning both computational and network resources (not necessarily optical) are obtained from a scalable linear program. In [18] network resource allocations are in fact fixed-bandwidth TCP connections, supplemented by a somewhat artificial maximum number of TCP connections allowed per network element. In our work, we apply Divisible Load Theory to lambda grids. In this case, data is transferred over (virtual) wavelength paths in an Optical Transport Network. These wavelength paths directly map onto the concept of a finite network resource, which eliminates the need to introduce artificial constraints in the scheduling model.

For the on-line Grid scheduling problem, a framework has been presented in [19]. This framework supports multiple resource types by construction. The authors have used the framework to evaluate on-line scheduling algorithms based on economic principles, making use of the "marginal cost" concept. The cost functions presented in [19] are geared towards dealing with multiple resource types within a single machine, while in this work, we consider resources such as the interconnecting network on-par with Computing Elements. In addition, we focus on two-level hierarchical Grid scheduling policies, which are more scalable—and thus more realistic to be implemented in operational Grids.

Scheduling requests in lambda Grids has been addressed in [20]. The emphasis is on dynamic on-demand lightpath provisioning schemes, whereas in our work steady-state scenarios (featuring long-lived pre-established lightpaths) are the major point of interest.

3 Models

A complete description of a Grid scheduling framework comprises 4 major components [21]:

- the application model
- the resource model
- the scheduling policy
- the performance model

The major properties of these components, as used in this work, are detailed in the sections below.

3.1 Application model

In this paper, we will use the term *job* when identifying atomic units of work. Thus, these jobs cannot be parallelized, nor can they be distributed across multiple Computational Elements. Jobs can be constrained by precedence relations, but do not depend on (and thus, do not communicate directly with) other jobs during their execution.

Each job j has a *length* l_j, which is a measure of that job's running time on a reference Computational Element, in absence of other workload on that element and in absence of bottlenecks created by other resources.

Jobs can process multiple input data sets and can generate multiple output data sets. These data sets are read and written in a number of equally sized *chunks*. The number of chunks into which input data set k for job j is divided is called n^i_{jk}; similarly, the number of chunks into which output data set k for job j is divided is called n^o_{jk}. We assume these input and output transfers occur in parallel with the execution of an instruction block.

This provides a generic job model supporting both data *streaming* and *pre-staging*. If for some data set k n^i_{jk} is big (or infinite), this data sets is almost continuously streamed to the job during its lifetime. On the other hand, if n^i_{jk} equals 1 for this data set, the entire data set is pre-staged to the Computational Element executing the job before it is started.

The job model clearly reflects the typical resource allocation interdependence: if an input data chunk is delivered late, the job's processing is suspended until the data chunk has been received. The occurrence of such processing stalls indicates that Computational Element allocation (the time share allocated to that job) and Network Element allocation (bandwidth allocated to the delivery of the offending data set) are not in ideal correspondence.

A sample visual interpretation of this job model is presented in Fig. 1. In this example, job j, executed on the reference Computational Element, needs 1 input data

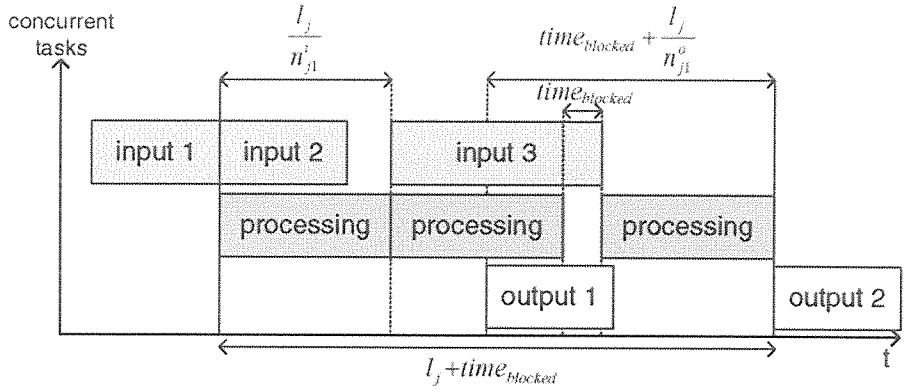


Fig. 1 Job model: illustration

set and produces 1 output data set; n^i_{j1} is 3 while n^o_{j1} is 2. Suspension of the job's computational progress is illustrated in case the last input data chunk is delivered late.

When evaluating scheduling policies in this paper, we limit ourselves to using jobs requiring a single input data set and emitting a single output data set.

3.2 Resource model

The Computational Elements in the Grids under investigation are assumed to be *time-shared* in nature. This means that they support dividing their processing power over multiple concurrent jobs. In addition, it is assumed that the exact size of the time share allocated to some job can be negotiated, guaranteed and allocated prior to assigning this job to that particular Computational Element—in other words, the Computational Elements provide a sufficient level of *CPU QoS*. In this light, the most important property of a Computational Element c is its capacity (i.e. total processing power) C_c.

In a similar way, we assume that Data Storage Elements can accommodate multiple concurrent allocations as well. As we will focus on Computational and Network Elements in the remainder of this paper, we will assume that each Storage Element provides sufficient space for the jobs writing output data to it.

Input data sets needed by jobs are stored on Information Elements. The distribution of the various data sets across these Information Elements is fixed and static. In particular, the effects of data replication (copying popular data sets to multiple Information Elements) and data scheduling (moving data closer to the location were it will be processed) are not investigated in this paper.

When dealing with lambda grids, the interconnecting network (between the participating Grid sites) comes in the shape of a Circuit-Switched Optical Transport Network. In order to transfer data between different sites, so-called *lightpaths* must be set up between these sites. The provisioning of an adequate number of lightpaths, the routing of lightpaths over the different network edges and Optical Cross Connects and the exact wavelengths allocated to a lightpath are decided upon when dimensioning this network. As this paper deals with scheduling on an already deployed Grid infrastructure, it can be assumed that lightpaths have been setup and routed. A fixed-bandwidth window on these lightpaths can be allocated to the jobs to be scheduled, much like the Computational Element allocation is performed.

3.3 Scheduling policies

The on-line scheduling policies detailed in Sect. 5 operate as two-level hierarchical scheduling algorithms and schedule arriving jobs as soon as possible on a suitable set of resources (the criteria used to rank resources and assign them a level of suitability is what the algorithms differ in). The algorithms do not operate in *batch* mode (in which scheduling is performed at regular time instants and all unscheduled jobs are scheduled together).

Jobs cannot be pre-empted: once a job is started on its allocated resources, it runs on those resources until completion. This means that jobs do not migrate and cannot be interrupted (checkpointed) and rescheduled at a later time, even if such action improves the resulting schedule.

The first level of the on-line scheduling framework concerns itself with resource *selection*. The metrics used to make this selection are described in the following section. Once a resource set has been selected, these resources are co-allocated to the job in order to minimize the job's *response time*. This resource *allocation* is performed by the local resource managers of the selected resources without involvement of the resource selection mechanism.

3.4 Performance metrics

Both the off-line and on-line scheduling algorithms described distribute the Grid's *load* across all participating Grid resources. The *load* of a resource over some period of time is taken to be the amount of work performed by that resource during the observed time period (see Fig. 2) divided by the resource's capacity, resulting in a number of "resource-seconds"—the minimal amount of time the resource needs to perform all of the work performed in the observed interval. Thus, the observed load is influenced by the time of observation, and the arrival and completion of jobs on the observed resource. Unless mentioned otherwise, in this paper, the time interval over which a resource's load has been calculated upon arrival of a new job starts at that job's arrival time and ends at the current schedule's makespan. Due to the nature of the workloads studied in this paper (with workload sizes approximating the Grid's capacity), distributing these workloads may lead to a near *load balancing* situation (which would be obtained exactly by minimizing the maximal load observed on any one resource in the Grid).

When observing the Grid in steady-state, the load on a resource can be defined as the instantaneous amount of work being performed by the resource per time unit divided by the resource's capacity. For all resources, this yields a real number in [0, 1] for its steady-state load.

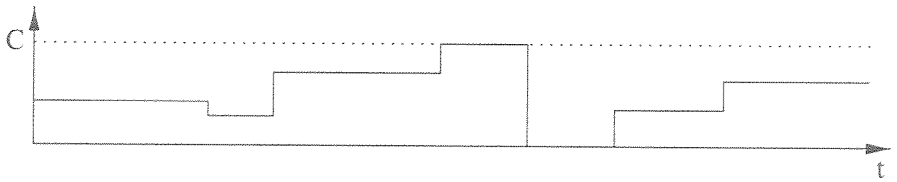


Fig. 2 Work performed by time-shared resource with capacity C over time

In both cases (i.e. with or without explicit time instants in the observations), these metrics allow us to compare (the load on) the various resources as these metrics are expressed in the same units (either time units or dimensionless numbers) regardless of the type of resource under investigation.

On-line scheduling algorithms can be compared using the resource loads observed during the scheduling process and using the *response time* (the difference between completion and arrival times) experienced by the scheduled jobs making up the Grid's workload.

4 Offline multi-resource scheduling

4.1 Offline scheduling formulation

Off-line scheduling problems in which each scheduling decision involves the allocation of multiple resources have been studied extensively in literature. Most important classes of multicomputer and multiprocessor scheduling problems can be described using Linear Programming as a special instance of a Resource Constrained Project Scheduling Problem. This general case supports the description of workloads needing allocations with multiple resource types, where each resource requirements is known a priori.

While supporting multiple resource types is a fundamental requirement for every Grid scheduling model (as e.g. Computational Elements, Storage Elements and Network Links are all to be treated as first class resources in such an environment), Grid scheduling problems differ from Project scheduling problems in a few ways.

First, resource allocations for a single Grid job requiring multiple resources may not be independent: Grid jobs which have been allocated only a small amount of CPU time probably cannot make full use of high bandwidth network connections to Storage Elements, as they simply do not have time to process data arriving at full rate.

In addition, the exact size of a Grid job's resource requirements is part of the scheduling process and not fully known in advance. This is because the jobs can be scheduled on time-shared resources (the size of the share is determined in the scheduling process), and, because of the resource allocation interdependence mentioned before, this may affect all other resource allocations for that particular job as well.

In Sect. 4.2, we model the Grid scheduling problem (with resource and application models as described in Sect. 3) as an extended version of a Resource Constrained Project Scheduling Problem. The extensions have the explicit goal of taking into account the resource allocation interdependence and the dynamic nature of their sizes.

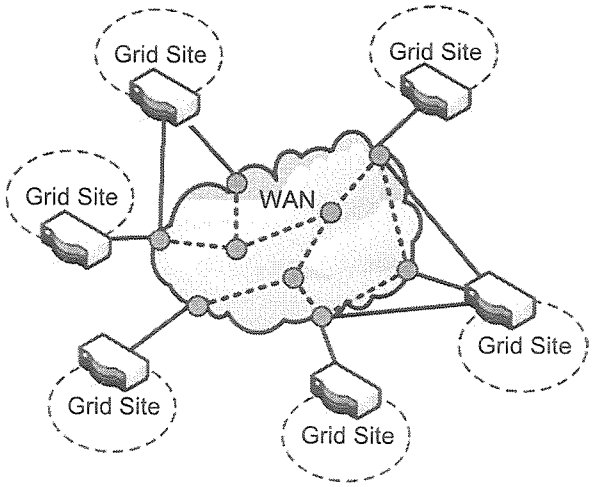
However, while this formulation does completely capture our model of the off-line Grid Scheduling problem in a Linear Program, we argue that this program quickly becomes intractable even for modest Grid sizes. Therefore, in Sect. 4.3, by making additional assumptions, we propose a more scalable version of this Linear Program by making use of Divisible Load Theory.

4.2 RCPSP model

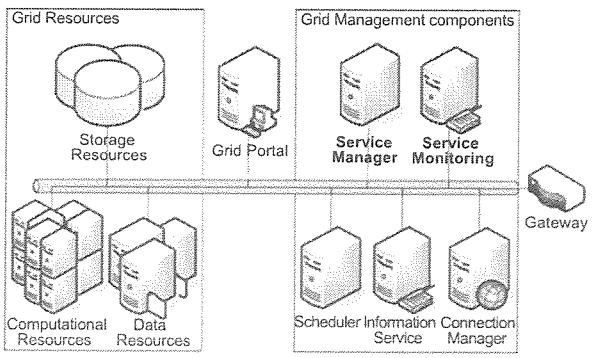
The off-line grid scheduling problem involving J jobs can be described as an extension of a Multi-Modal Resource Constrained Project Scheduling Problem (and thus, modeled into an Integer Linear Program), where each mode in which a job can be executed is the collection of resource allocations assigned to that job.

For this Linear Program, assume that each job j (which is submitted on its *home* Computational Element h_j) is executed on a single Computational Element and processes a single data stream, located at the site where this job was submitted. All output data generated by a job must be returned to its submission site, where it is presented to the user responsible for launching the job. The job j is launched at time r_j, its computational length is l_j, the amount of input data it processes is d_j^i bytes, and the amount of output data generated is d_j^o.

The Grid itself is modeled after Fig. 3 and adheres to the following properties:



(a) Grid model



(b) Individual grid site model

Fig. 3 Grid and grid site conceptual models

- The Grid is made up of several *sites*; these sites are interconnected through a Optical Circuit Switched Network, in which (virtual) lightpaths between sites have been established.
- Each site consists of a time-shared Computational Resource c with processing capacity C_c and a time-shared Storage Resource s with total storage capacity S_s, and is connected to the OCS network via through a gateway.
- The intra-site networks are assumed to have sufficient capacity in order not to create bottlenecks.
- The number of (virtual) wavelength paths that have been setup between the sites hosting Computational Elements c and c' is $\lambda_{cc'}$. Each wavelength paths offers data rate B. The total number of wavelength paths that have been set up in the network is denoted L.

The key concept in modeling the complete Grid scheduling problem (as opposed to a standard MMRCPSP) as an Integer Linear Program is to associate a set of *dummy jobs* with each job j; the number of dummy jobs needed equals the number of resource types that need to be allocated by this job. For instance, focusing on data streaming jobs, 3 dummy jobs are needed for each real job, all of which are to execute simultaneously: one for the input data transfer, one for the output data transfer and one for the processing (in contrast, for data staging jobs, we would need the same three dummy jobs but with precedence relations between them).

Directly related to the dummy jobs are the decision variables needed in the problem: each dummy job needs to be scheduled on an available resource of the appropriate type (identified by the type of dummy job).

Assuming a discrete time scale, with increasing time instants of interest labeled $0, 1, \ldots, t, \ldots, T$ (these time instants need not be equidistant but can, for scalability reasons, as well be (geometrically) increasing in distance—see [7] for instance), this leads us to introduce binary decision variables r_{jRbe} which equals one if and only if the dummy job of type r for job j is executed on resource R starting no earlier than time instant b and ending no later than time instant e.

If we want to obtain a load balancing schedule (we will not consider alternative factors such as resource usage cost), the following objective function may be used as a minimization criterium:

$$Objective = MaxLoad + P * Makespan \tag{1}$$

In this expression, "Makespan" refers to the resulting schedule's makespan, "MaxLoad" is the maximal load (over all resources) assigned to a resource and "P" is a penalty factor which, when taken large enough, enforces the selection of a load balancing schedule among all feasible schedules featuring minimal makespan by eliminating unnecessary idle time. These auxiliary quantities can be obtained from the constraints

$$\forall j.Makespan \geq \sum_{c=1}^{C} \sum_{b=0}^{T-1} \sum_{e=b+1}^{T} t_e r^p_{jcbe} \tag{2}$$

$$\forall c.MaxLoad \geq \sum_{j=1}^{J} \sum_{b=0}^{T-1} \sum_{e=b+1}^{T} \frac{r^p_{jcbe} l_j}{C_c} \tag{3}$$

Table 1 Off-line grid scheduling as an extension of MMRCPSP: linear program size

# Variables	# Constraints
$\frac{(C+2L)JT(T+1)}{2}$	$3J + C + (T+1)(C + L + JCT)$

$$\forall \lambda . MaxLoad \geq \frac{\sum_{j=1}^{J} \sum_{b=0}^{T-1} \sum_{e=b+1}^{T} d_j^i r_{j\lambda be}^i + d_j^o r_{j\lambda be}^o}{B} \tag{4}$$

Capacity constraints on the various resources are given by

$$\forall c . \forall t . \sum_{j=1}^{J} \sum_{b=0}^{t} \sum_{e=t+1}^{T} \frac{l_j r_{jcbe}^p}{t_e - t_b} \leq C_c \tag{5}$$

$$\forall \lambda . \forall t . \sum_{j=1}^{J} \sum_{b=0}^{t} \sum_{e=t+1}^{T} \frac{d_j^i r_{j\lambda be}^i + d_j^o r_{j\lambda be}^o}{t_e - t_b} \leq B \tag{6}$$

$$\forall s . \sum_{j : s \in h_j} d_j^o \leq S_s \tag{7}$$

Since a job j only arrives at time r_j, it cannot be started before that time:

$$\forall j . \sum_{c=1}^{C} \sum_{b<r_j} \sum_{e=b+1}^{T} r_{jcbe}^p = 0 \tag{8}$$

A unique schedule is obtained if

$$\forall j . \sum_{c=1}^{C} \sum_{b=0}^{T-1} \sum_{e=b+1}^{T} r_{jcbe}^p = 1 \tag{9}$$

$$\forall j . \forall c . \forall b . \forall e . r_{jcbe}^p = \sum_{\lambda : h_j \to c} r_{j\lambda be}^i \tag{10}$$

$$\forall j . \forall c . \forall b . \forall e . r_{jcbe}^p = \sum_{\lambda : c \to h_j} r_{j\lambda be}^o \tag{11}$$

The previous Linear Program's computational complexity is summarized in Table 1.

4.3 DLT model

Two obvious causes for the rapid intractability (for increasing Grid sizes) of the previous Linear Program are the number of jobs involved and the number of discrete time instants of interest, as the program's complexity increases proportionally to these parameters.

The concept of Divisible Load [16, 17] explicitly deals with these hurdles in the context of off-line scheduling problems by making the following assumptions:

- the workload to be scheduled is assumed to be arbitrarily divisible
- the Grid system, on which workload is to be scheduled, is analyzed in *steady-state*

The first assumption (restricting the scheduling problem's analysis to steady-state Grids) eliminates the need to investigate discrete time instants. Instead, only the continuous quantities representing the amount of workload *per unit of time* being moved around is of importance.

Assuming that the workload is arbitrarily divisible means that only the aggregate workload arriving over some time window is of importance and not the individual jobs it is made up of. This ensures that the scheduling problem can be described by real-valued variables $\alpha_{c'}^{c} \geq 0$, denoting the amount of computational workload arriving per time unit at Computational element c (in that workload's home site) which is ultimately processed by remote Computational Element c', and eliminates the need for per-job decision variables.

Again, we will not take into account resource costs, but rather attempt to find a load balancing schedule. This is equivalent to minimizing the maximal load found on any one resource, and this maximal resource load (relative to each resource's capacity) can be obtained from constraints of the form

$$\forall c.Load \geq \frac{\sum_{c' \neq c} \alpha_c^{c'}}{C_c} \tag{12}$$

$$\forall c.\forall c' \neq c.Load \geq \frac{\alpha_{c'}^{c} d_c^i + \alpha_c^{c'} d_{c'}^o}{B\lambda_{cc'}} \tag{13}$$

The resource capacity constraints now become

$$\forall c. \sum_{c'} \alpha_c^{c'} \leq C_c \tag{14}$$

$$\forall c.\forall c' \neq c.\alpha_{c'}^{c} d_c^i + \alpha_c^{c'} d_{c'}^o \leq B\lambda_{cc'} \tag{15}$$

In the last equation, d_c^i and d_c^o represent the average input and output data set sizes for jobs arriving at Computational Element c, divided by the average computational length of these jobs.

A meaningful schedule is obtained if

$$\forall c. \sum_{c'} \alpha_{c'}^{c} = \alpha^c \tag{16}$$

with α^c denoting the total workload arriving at Computational Element c per time unit.

Modelling the off-line Grid Scheduling using the divisible load approach yields a Linear Program with lower computational complexity, as shown in Table 2.

Table 2 Off-line grid scheduling using divisible load: linear program size

# Variables	# Constraints
$C(C-1)$	$C(2C+1)$

5 Two-tier online scheduling algorithms

5.1 Online scheduling framework

Off-line scheduling models like the ones presented in Sect. 4 commonly appear in the scope of a Grid dimensioning problem [22]. In such an off-line dimensioning problem, each resource's capacity is a decision variable rather than an input constant. The exact computational capacity and (in a lambda grid) number of lightpaths to be installed between sites depends on the envisioned job schedule, as these scheduling decisions directly influence the necessary (remote) processing power and network bandwidth.

Once a Grid has been deployed, arriving jobs will of course be scheduled onto it following an *on-line* scheduling policy. While on-line scheduling algorithms and heuristics have been studied extensively in literature [5–7], most concentrate on problems with a single resource type (i.e. CPU time).

Since multiple resource types are a key element in every Grid, scheduling heuristics taking into account one resource at a time are not the most appropriate for the Grid scheduling problem. Moreover, many of these approaches lack a sound mathematical foundation as resource usage and cost for different resource types are incompatible and involve different units of measurement.

To tackle this problem, a *unified* approach to modelling resource assignment cost was proposed by Keren et al. in [19]. They describe a on-line scheduling framework which allows for the inclusion of different resource types by using dimensionless quantities, in particular the work assigned to a resource divided by that resource's capacity. In addition, an optimization goal based on economic principles and marginal cost analysis is proposed as an improvement of a greedy list scheduling algorithm.

Because of the sheer size of typical Grids, it is unrealistic to envision a fully centralized scheduling system responsible for managing and co-allocating resources to every job. Instead, as explained in Sect. 3.3, it is more reasonable to assume a hierarchical two-tier scheduling model where a top level scheduling system makes use of local resource schedulers.

The online load-balancing algorithms examined in Sect. 5 are situated at this top level scheduling system. First, we will show how the unified online framework can be used within the Grid scheduling model described above. In particular, we describe how the greedy algorithm and the marginal cost based algorithm have been implemented for our simulations. These algorithms are purely state-based: they do not rely on information concerning resource usage history or future job properties.

The last class of algorithms we present not only uses resource state information, but also employs results obtained by solving the offline Grid scheduling problem developed in Sect. 4.3. These results represent the optimal steady-state Grid workload distribution for the workload for which the scheduling problem was solved (in the context of a dimensioning problem, this workload would commonly represent the

most stressing load the Grid needs to be able to handle). This optimal steady-state off-line workload distribution is used in the algorithm as a *target* load, and the algorithm's optimization goal is to mimic this target load in an on-line fashion.

5.2 Greedy scheduling algorithm

The *greedy* on-line scheduling algorithm schedules jobs as soon as they arrive; suitable resources are selected by attempting to minimize the resulting schedule's (i.e. the schedule consisting of the newly arrived job and all jobs which have already been allocated) maximal load on any single resource.

Formally, the greedy algorithm attempts to obtain a load-balancing schedule by selecting the resource set $\mathcal{R}_j$ for a newly arriving job j such that the quantity

$$\max_{r \in \mathcal{R}} l_r^{\mathcal{R}_j} \tag{17}$$

is minimized, where $l_r^{\mathcal{R}_j}$ denotes the load on resource r given that job j is scheduled on the resources contained in the set $\mathcal{R}_j$. If this maximal resulting resource load is equal for two different resource assignments, the load of the resource having the second highest load is compared and so on. This approach can be used (and is implemented as such) as a comparison operator for the resource load *vectors* obtained with each resource assignment.

The load for a single resource is calculated as specified in Sect. 3.4, where the observed time interval is taken to start at the new job's arrival time and ends at the schedule's makespan.

5.3 Opportunity cost based scheduling algorithm

The opportunity cost algorithm also schedules jobs as soon as they arrive, but uses an objective function based on marginal cost analysis to select a suitable resource set. The previous algorithm, greedy, essentially tracks the resulting increase in resource load induced by every possible scheduling decision.

In the opportunity cost algorithm, the quantity to be minimized (the marginal cost of the resource assignment) is given by

$$\sum_{r \in \mathcal{R}} a^{l_r^{\mathcal{R}_j}} - a^{l_r} \tag{18}$$

where l_r now denotes the load on resource r *before* job j had been scheduled on the resources contained in set $\mathcal{R}_j$ and a is a constant > 1.

Note that this cost function is not only sensitive to the increase in resource load, but also in the size of the increase relative to the current load allocated to the resource.

5.4 DLT based scheduling algorithms

The DLT based on-line scheduling algorithms get additional input (the steady-state *target* load for each resource) from the solution to the off-line scheduling problem (e.g. performed when the Grid is being dimensioned) as described in Sect. 4.3.

The cost function associated with a particular resource assignment for job j can take on the form

$$\max_{r \in \mathcal{R}} d_r^{\mathcal{R}_j} \tag{19}$$

or, using the marginal cost approach,

$$\sum_{r \in \mathcal{R}} a^{d_r^{\mathcal{R}_j}} - a^{d_r} \tag{20}$$

In the former case, as with the Greedy algorithm, the quantity of interest is actually a *vector* rather than a scalar.

In the above expressions, d_r is the *deviation* from the target load observed on resource r before job j has been scheduled on resource set $\mathcal{R}_j$, and $d_r^{\mathcal{R}_j}$ is the deviation from resource r's target load observed on it in the resulting schedule.

Given the target steady-state load t_r for resource r and currently observed load l_r, the deviation d_r can be calculated using one of the following alternatives:

$$d_r = \frac{l_r}{t_r} \tag{21}$$

$$d_r = |t_r - l_r| \tag{22}$$

$$d_r = \max(0, l_r - t_r) \tag{23}$$

Again, the proposed definitions and equations are mathematically sound in the sense that only dimensionless numbers are used when calculating costs involving different resource types.

Note that, given the previous definitions of the deviation concept, a resource's target load deviation does not necessarily increase when assigning extra load to this resource—in contrast to the total workload assigned to the resource. Therefore, for the rest of this paper, we will not pursue the use of the marginal cost approach when resource target load deviation is the metric of choice.

As with the previously presented algorithms, the scheduling algorithm elects the resource set yielding minimal cost.

It is worth noting that, although the divisible load based on-line algorithms use more information when compared to the other algorithms, this does not result in increased computational complexity as the required information has been calculated and made available from the off-line combined scheduling and dimensioning problem.

6 Evaluation: setup

The main experiments performed and described in this paper pertain to the evaluation and comparison of the hierarchical two-level on-line scheduling algorithms presented in Sects. 5.2–5.4 and their comparison to the fully centralized single-level algorithm. Due to the nature of the Divisible Load based algorithms, the experiments consist of two phases:

- Solving an off-line steady-state scheduling problem (in the context of a Grid dimensioning problem)
- Simulating the on-line scheduling of the appropriate workload (approximated in the first phase) on the Grid used in the first phase

Our experiments have been performed for a wide range of parameter variations: for our simulations, we have varied the Grid interconnection network connectivity, the stochastic workload used during the dimensioning and scheduling phases and—for the divisible load based algorithms—the definition and implementation of the divisible load cost function and deviation metric.

6.1 Simulated topologies

Our simulations have been performed for different Grid interconnection (i.e. Optical Transport Network) topologies, inspired by the sample European network shown in Fig. 4.

We generated sets of 10 random networks (all having 13 nodes) by constructing a connected set of 13 nodes through repeated addition of node-link pairs, and then added in extra links following a probability p in $[0, 1]$. This parameter was varied to generate networks with different average connectivity values. All Grid topologies have been dimensioned so that excess load generated at one site can be handled by the aggregation of all remote sites.

6.2 Simulated scenarios

The scheduling scenarios studied are those found in two-tier Grids. In particular, we study scenarios where excess workload arrives at one site (the top tier) which must then be distributed to the other, remote sites (the second tier).

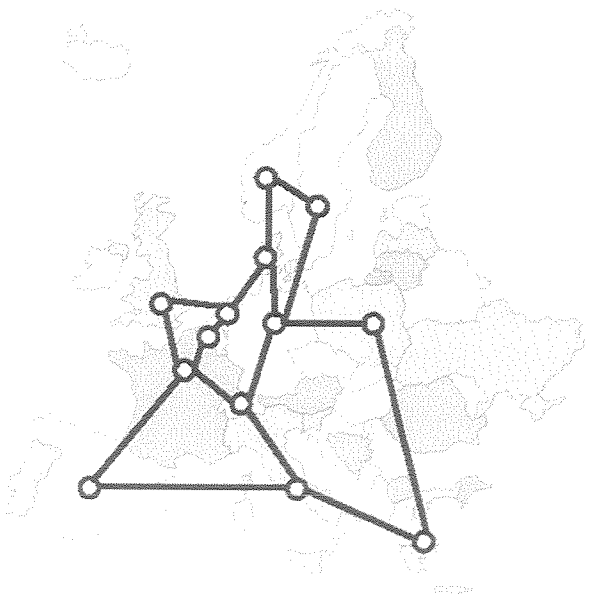


Fig. 4 13-node European network

In the dimensioning phase of our experiments, each Grid topology (as described in the previous section) has been dimensioned in order to support all possible two-tier excess load scenarios with a single source. This means that the off-line combined Grid dimensioning and scheduling problem solved describes, in our case, thirteen workload scenarios. The simulation results presented below were obtained by scheduling the same workload in these thirteen scenarios on the resulting Grid.

In order to measure the relevant metrics in steady-state, transient effects occurring in the start and end phases of each simulation (corresponding to workload build up and workload draining mechanics in the Grid) have been eliminated from our calculations.

6.3 Simulated workload

The workload used in the scheduling simulations was chosen in such a way that the average computational load arriving per time unit in each scenario equalled 90% of the Grid's computational capacity. An on-off distribution was used to model the intermittent arrival of large and small jobs.

In our scenarios, the average job interarrival time was taken to be 0.5 s. Each job reads its input from and returns it output to its submission site. The Grid's network has been dimensioned (see Sect. 7) to support the average expected network load resulting from this.

7 Evaluation: results and discussion

7.1 Grid interconnection network dimensioning

For different average job interarrival times, the resulting network cost of the dimensioned Grid has been shown in Fig. 5 for two different interconnection topology average connectivities. As stated in Sect. 6.3, for our experiments we have chosen an average job interarrival time of 0.5 s.

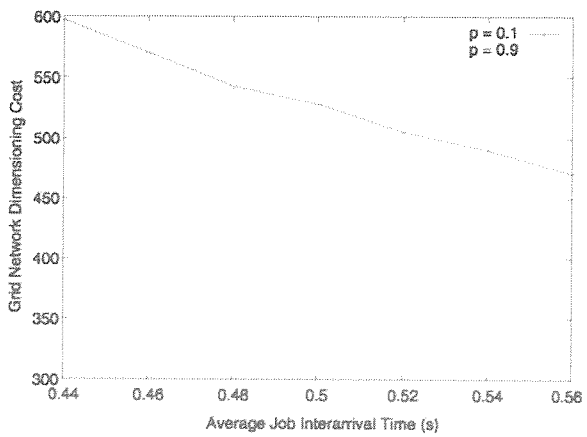
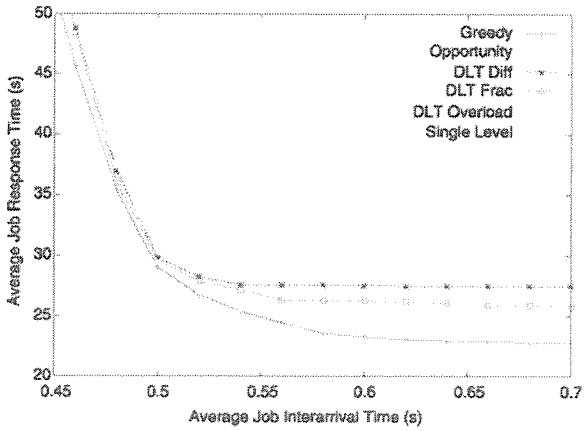


Fig. 5 Resulting grid dimensioning cost

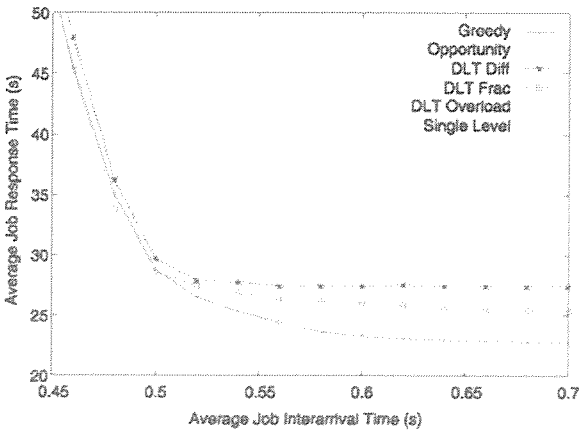
7.2 Job response time

In Fig. 6 the resulting schedule's average Job Response Time in steady state has been plotted for two sets of random networks (ten in each set), having connectivity parameter p set to 0.1 and 0.9, respectively. Along the x-axis is the average job interarrival time; as discussed in Sect. 6.3, the average load obtained when the average interarrival time is 0.5 s equals 90% of the Grid's computational capacity. As expected, the fully centralized single-level algorithm (performing an exhaustive search) does the best job in minimizing the average job response time, but is followed closely by the hierarchical algorithms.

From the figure, it is clear that no single two-level algorithm outperforms the others by a significant margin for the workloads of interest (around the average job interarrival time of 0.5 s). For the workload on which the Grid dimensioning was performed (around an interarrival time of 0.5 s), the greedy, opportunity cost and DLT scheduling algorithms (using the fractional load deviation calculation) show the same performance. For small workloads (i.e., high interarrival times), the DLT-based



(a) $p = 0.1$



(b) $p = 0.9$

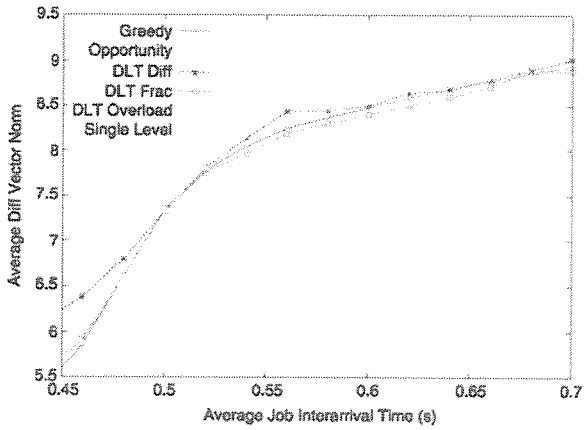
Fig. 6 Job response times: single level vs. hierarchical algorithms

scheduling algorithms perform worse when compared to the greedy and opportunity cost algorithms, due to their preference to fill slower resources first in order to minimize total target load deviation. As can be expected because the interconnecting topology is only of importance during the dimensioning phase (during the scheduling phase, only lightpaths are of importance, not their routing over the underlying optical transport network), a similar behavior is observed in both cases ($p = 0.1$ and $p = 0.9$) presented.

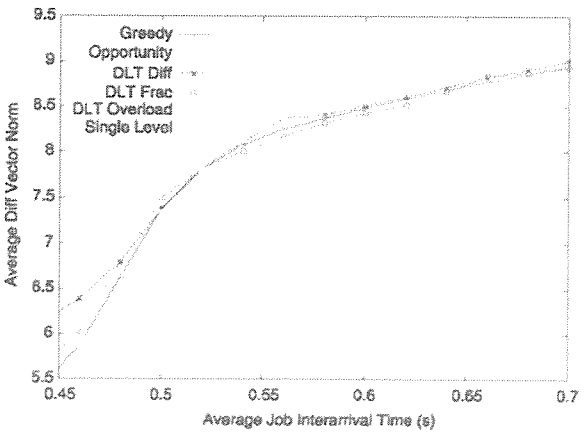
7.3 Resource target load difference

Further comparing the different algorithms, at each job's arrival we have constructed a vector containing, for each resource, the *deviation* (defined in Sect. 5.4) of the resource's current load from that resource's steady-state target load. The average norm of this vector in steady state has been plotted in Fig. 7 for the different algorithms.

Again, the two-level scheduling algorithms exhibit the same values for this metric around the main point of interest, which is the workload obtained for an average



(a) $p = 0.1$



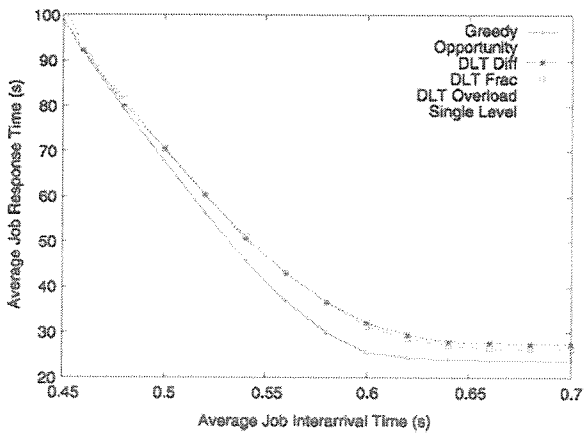
(b) $p = 0.9$

Fig. 7 Diff vector norm at job schedule time: single level vs. hierarchical algorithms

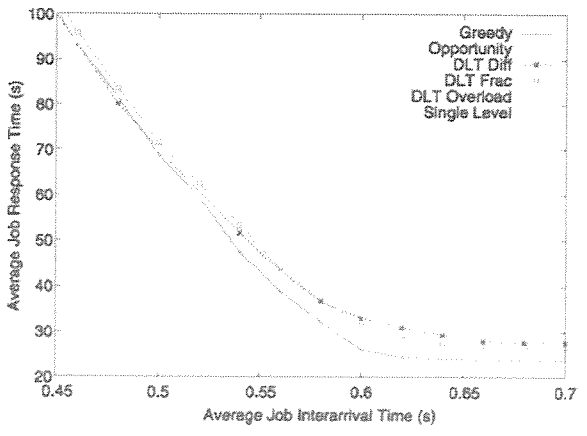
interarrival time of 0.5 s. The single-level algorithm, which attempts to minimize each job's response time will make more use of faster computational resources than prescribed by the off-line target loads, and vice versa for the slower computational resources. This explains why it generates a schedule deviating more from the off-line calculated resource target loads than the schedules calculated with the other algorithms. Around 0.5 s, the DLT-based algorithms follow the prescribed target load more closely, which is important for steady-state operation, as the steady-state is guaranteed when the prescribed target load is matched exactly.

7.4 Job length distribution

The above results were obtained for a stochastic job generating process. In this section, we repeat the simulations performed earlier for a different job generating process. The resulting job set features the exact same *average* length and interarrival time, but the distributions used now have double the standard deviation as before. The results for these new simulations are shown in Figs. 8 and 9. Again, the single

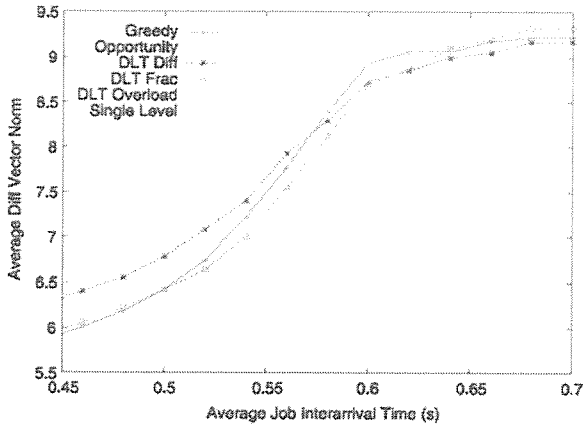


(a) $p = 0.1$

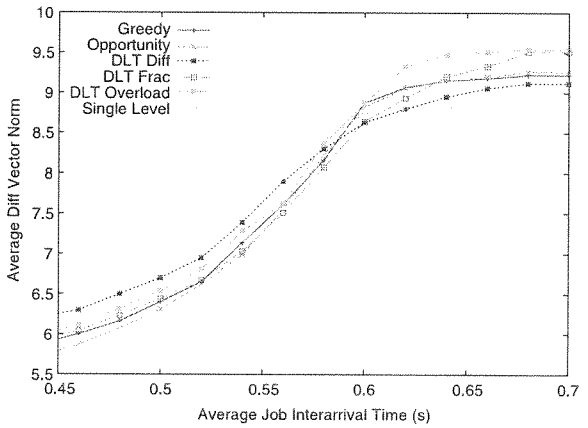


(b) $p = 0.9$

Fig. 8 Resulting job response times: increased job variability



(a) $p = 0.1$



(b) $p = 0.9$

Fig. 9 Diff vector norm at job schedule time: increased job variability

level scheduling algorithm provides the smallest average job turnaround times, but does not necessarily mimic the calculated target load distribution in doing so.

8 Conclusions

In this paper, we have introduced the use of Divisible Load Theory in a combined lambda grid dimensioning and scheduling problem. We have concentrated on two-tier Lambda Grids featuring a single excess load source, and have provided arguments in favour of our DLT-based approach.

We have assessed the gain obtained by using the off-line calculated workload distribution as target in an on-line hierarchical scheduling mechanism, in which the workload to be scheduled closely resembled the worst-case workload used to derive suitable Grid dimensions

For different Grid interconnection topologies, job generating processes and load metrics, we compared these algorithms to other (such as greedy and opportunity cost

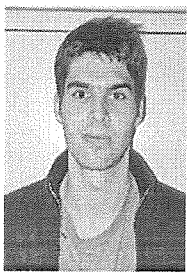
based) hierarchical scheduling algorithms (which do not explicitly take into account the off-line calculated workload distributions) as well as a single-level scheduling policy.

We found that, in the studied scenarios, no significant advantage is gained by using the off-line calculated workload distributions in an on-line algorithm in the way we described. In all of our experiments, a standard greedy resource selection policy is able to provide a good average job response time (and thus, keep the Grid operation in steady-state) which does not differ much from the job response times obtained by using a single-level algorithm which, due to scalability and computational constraints, is unlikely to be implemented in an operational Grid. Similar observations can be made when comparing the algorithms using deviation from the off-line calculated steady-state workload distribution as a metric. Our DLT-based algorithms, however, incorporate the additional workload information in the scheduling process without increasing its computational complexity, and offer reasonable performance (when measured using job response times and target load deviation) when compared to the optimum delivered by the single level scheduling algorithm.

References

1. Foster I, Kesselman C (eds) (1999) The grid: blueprint for a new computing infrastructure. Morgan Kaufmann
2. Foster I, Kesselman C (eds) (2003) The grid: blueprint for a new computing infrastructure, 2nd edn. Morgan Kaufmann
3. Smarr L, Ford J, Papadopoulos P, Fainman S, DeFanti T, Brown M, Leigh J (2005) The optiputer, quartzite, and starlight projects: a campus to global-scale testbed for optical technologies enabling lambdagrid computing. In: Optical fiber communication conference & exposition and the national fiber optic engineers conference (OFC/NFOEC) 2005, 2005
4. DeFanti T, de Laat C, Mambretti J, Neggers K, Arnaud BS (2003) Translight: a global-scale lambda-grid for e-science. Commun ACM 46(11):34–41
5. Hall L, Schulz A, Shmoys D, Wein J (1997) Mathematics of Operations Research. Scheduling to minimize average completion time: off-line and on-line approximation algorithms. Math Oper Res 22(3)
6. Feitelson D, Rudolph L, Schwiegelshohn U, Sevcik K, Wong P (1997) Theory and practice in parallel job scheduling. Springer
7. Sgall J (1998) On-line scheduling—a survey. Lecture notes in computer science, vol 1442
8. Kolish R, Padman R (2001) An integrated survey of project scheduling. OMEGA Int J Manag Sci 29(3)
9. Ranganathan K, Foster I (2003) Simulation studies of computation and data scheduling algorithms for data grids. J Grid Comput 1(1)
10. Cameron D, Carvajal-Schiaffino R, Millar A, Nicholson C, Stockinger K, Zini F (2003) Evaluating scheduling and replica optimisation strategies in optorsim. In: 4th International workshop on grid computing (Grid2003), 2003
11. Ernemann C, Hamscher V, Streit A, Yahyapour R (2002) Enhanced algorithms for multi-site scheduling. In: 3rd International workshop on grid computing (Grid2002), 2002
12. Bucur A, Epema D (2002) An evaluation of processor co-allocation for different system configurations and job structures. In: 14th IEEE symposium on computer architecture and high performance computing, 2002
13. Bucur A, Epema D (2000) The influence of the structure and sizes of jobs on the performance of co-allocation. In: 6th Workshop on job scheduling strategies for parallel processing, 2000
14. Buyya R, Murshed M, Abramson D (2002) A deadline and budget constrained cost-time optimization algorithm for scheduling task farming applications on global grids. In: The 2002 International conference on parallel and distributed processing techniques and applications (PDPTA'02), 2002

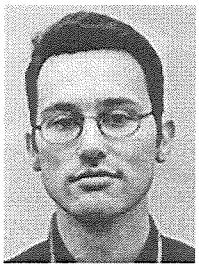
15. Thysebaert P, Volckaert B, Turck FD, Dhoedt B, Demeester P (2004) Network aspects of grid scheduling algorithms. In: 17th International conference on parallel and distributed computing systems (PDCS'04), 2004
16. Hung J, Kim H, Robertazzi T (2002) Scalable scheduling in parallel processors. In: 36th Annual conference on information sciences and systems (CISS'02), 2002
17. Yu D, Robertazzi, T (2003) Divisible load scheduling for grid computing. In: 16th International conference on parallel and distributed computing systems (PDCS'03), 2003
18. Marchal L, Yang Y, Casanova H, Robert Y (2005) A realistic network/application model for scheduling divisible loads on large-scale platforms. In: 19th IEEE international parallel and distributed processing symposium (IPDPS'05), 2005
19. Keren A, Barak A (2003) Opportunity cost algorithms for reduction of i/o and interprocess communication overhead in a computing cluster. IEEE Trans Parallel Distrib Syst 14(1):39–50
20. Farooq U, Majumdar S, Parsons E (2005) Dynamic scheduling of lightpaths in lambda grids. In: IEEE/create-net international workshop on networks for grid applications (GRIDNETS 2005), 2005
21. Zhu Y (2003) A survey on grid scheduling systems, Ph.D. Qualifying exam. Hong Kong University of Science and Technology
22. Thysebaert P, Leenheer MD, Volckaert B, Turck FD, Dhoedt B, Demeester P (2006) Scalable dimensioning of optical transport networks for grid excess load handling. Photon Netw Commun 12(2):117–132



Pieter Thysebaert received his M.Sc. degree in Computer Science Engineering from Ghent University, Belgium, in June 2001. He is now a research assistant and Ph.D. student affiliated with the Department of Information Technology at Ghent University and has received a scholarship by the FWO (Fund for Scientific Research—Flanders). His main interests include Grid simulation and modelling of Grid scheduling problems.

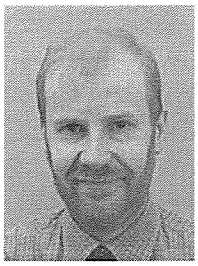
Bruno Volckaert received his M.Sc. degree in Computer Science Engineering from Ghent University, Belgium, in June 2001. He is now a research assistant and Ph.D. student affiliated with the Department of Information Technology at Ghent University and has received a scholarship by the IWT (Institute for Innovation in Science and Technology—Flanders). His main interests include Grid management architectural designs and Grid simulation.

Marc De Leenheer received his M.Sc. degree in Computer Science Engineering from Ghent University, Belgium, in June 2003. He is now a research assistant and Ph.D. student affiliated with the Department of Information Technology at Ghent University and has received a scholarship by the IWT (Institute for Innovation in Science and Technology—Flanders). His main interests include modeling and optimization of Grid management architectures, specifically in the context of photonic networks.



Filip De Turck received his M.Sc. degree in Electrical Engineering from Ghent University, Belgium, in June 1997. In May 2002, he obtained the Ph.D. degree in Electrical Engineering from the same university. From October 1997 to September 2001, he was research assistant with the Fund for Scientific Research—Flanders, Belgium (F.W.O.-V.). At the moment, he is a part-time professor and a post-doctoral fellow of the F.W.O.-V., affiliated with the Department of Information Technology at Ghent University. Filip De Turck is author or co-author of approximately 80 papers published in international journals or in the proceedings of international conferences. His main research interests include scalable software architectures for telecommunication networks, service management, performance evaluation and optimization of routing, admission control and traffic management in telecommunication systems.

Bart Dhoedt received a degree in Engineering from Ghent University, Belgium, in 1990. In September 1990, he joined the Department of Information Technology at the same university. His research, addressing the use of micro-optics to realize parallel free space optical interconnects, resulted in a Ph.D. degree in 1995. After 2 years of post-doctoral research in opto-electronics, he became professor at the Faculty of Engineering, Department of Information Technology. Since then, he is responsible for several courses on algorithms, programming and software development. His research interests are software engineering and mobile and wireless communications. Bart Dhoedt is author or co-author of approximately 100 papers published in international journals or in the proceedings of international conferences. His current research addresses software technologies for communication networks, peer-to-peer networks, mobile networks and active networks.



Piet Demeester received the M.Sc. degree in Electrical Engineering and the Ph.D. degree from Ghent University, Belgium, in 1984 and 1988, respectively. In 1992 he started a new research activity on broadband communication networks resulting in the IBCN group (INTEC Broadband Communications Network research group). He became professor at Ghent University in 1993 and is responsible for the research and education on communication networks. The research activities cover various communication networks (IP, ATM, SDH, WDM, access, active, mobile), including network planning, network and service management, telecom software, internetworking, network protocols for QoS support, etc. Piet Demeester is author of more than 400 publications in the area of network design, optimization and management. He is member of the editorial board of several international journals and has been member of several technical program committees (ECOC, OFC, DRCN, ICCCN, IZS).

Full Record

Record 1 of 1 (Set #1)

Title: Dimensioning **and on-**line scheduling **in** Lambda Grids using divisible load concepts

Author(s): Thysebaert P (Thysebaert, Pieter), Volckaert B (Volckaert, Bruno), De Leenheer M (De Leenheer, Marc), De Turck F (De Turck, Filip), Dhoedt B (Dhoedt, Bart), Demeester P (Demeester, Piet)

Source: JOURNAL OF SUPERCOMPUTING 42 (1): 59-82 OCT 2007

Document Type: Article

Language: English

Cited References: 22 **Times Cited:** 0

Abstract: Due to the large amounts of data required to be processed by the typical Grid job, it is conceivable that the use of optical transport networks in Grid deployment (hence the term "**Lambda** Grid") will increase. The exact topology of the interconnecting network is obtained by solving a **dimensioning** problem, and the outcome of this strongly depends on both the expected workload characteristics and Grid **scheduling** policy. Solving this combined **scheduling** and **dimensioning** problem **using** straightforward ILP modelling is cumbersome; however, for steady-state Grid operation, **Divisible Load** Theory (DLT) can yield scalable formulations of this problem.

In this paper, the on-**line** hierarchical **scheduling** on a **lambda** Grid of workload approaching the Grid's capacity in a two-tier Grid mode of operation is studied. A number of these algorithms are goal-driven, in the sense that target per-resource goals are obtained from the off-**line** solution to the **Divisible Load** model. We compare these on-**line** multiresource **scheduling** policies for different workloads, Grid interconnection topologies and Grid parameters. We show that these algorithms perform well in the studied scenarios when compared to a fully centralized **scheduling** algorithm.

Author Keywords: Lambda grids; optical transport networks; optimization; **divisible load**

Addresses: Thysebaert P (reprint author), Ghent Univ IBBT IMEC, Dept Informat Technol, Gaston Crommenlaan 8, B-9050 Ghent, Belgium
Ghent Univ IBBT IMEC, Dept Informat Technol, B-9050 Ghent, Belgium

E-mail Addresses: pieter.thysebaert@intec.ugent.be

Publisher: SPRINGER, VAN GODEWIJCKSTRAAT 30, 3311 GZ DORDRECHT, NETHERLANDS

Subject Category: Computer Science, Hardware & Architecture; Computer Science, Theory & Methods; Engineering, Electrical & Electronic

IDS Number: 202MV

ISSN: 0920-8542

Record 1 of 1 (Set #1)