Evaluation of a Job Admission Algorithm for Bandwidth Constrained Grids

Marc De Leenheer, Pieter Thysebaert*, Bruno Volckaert,
Filip De Turck†, Bart Dhoedt, Piet Demeester
Department of Information Technology, Ghent University - IMEC
Sint-Pietersnieuwstraat 41, B-9000 Gent, Belgium.
Tel.: +32 9 267 35 87, Fax: +32 9 267 35 99
E-mail: Marc.DeLeenheer@intec.UGent.be

Abstract *One of the goals envisioned by Grid computing is to make the execution of both computational and data-intensive problems possible. A key problem is finding the optimal set of resources to use for each submitted job, especially when we take network usage into consideration. To address this problem, we introduce an integer linear programming model, which incorporates the functions of job admission control and resource assignation. Scheduling decisions are based on a cost model for resources. We analyse our algorithm in terms of job acceptance rate, resource utilization and average job queueing time. The results show the correctness of the model, and prove the importance of network-awareness in grid scheduling algorithms.*

Keywords: Grid Computing, Integer Linear Programming, Scheduling Strategies.

1 Introduction

Grid computing [1] assembles geographically distributed computer resources into a virtual supercomputer, allowing co-allocation of computing power and data-sharing functionality. Research projects such as the European DataGrid [2] estimate that the data to be shared will soon reach Petabyte scale. This not only puts severe constraints on the computational and storage resources, but also on the network which transports these datasets.

A scheduling algorithm must answer the following fundamental question: Which resource set should be used when for each job? We will present an integer linear programming (ILP) model [3], which incorporates both *admission control* and *resource allocation* functions. We then investigate the behaviour of our algorithms under different grid and job loads, and pay special attention to the effects of network constraints in grids.

Existing grid scheduling algorithms show limited interest in bandwidth usage. For instance, James et al. present and evaluate several heuristics in [4], as do Hamscher et al. in [5], but none take network constraints into account. Recently, Ranganathan and Foster combined resource assignation and data replication algorithms in [6]. Our paper takes a different approach: we don't allow data replication, but instead consider the network topology a first-class entity of our scheduler.

The rest of this paper is organized as follows. We introduce our grid model, along with various notations, in Section 2. Then we present our scheduling algorithms, and proceed by describing our evaluation setup. Our results, together with their analysis, are given in Section 5. Finally, we conclude and discuss future work in Section 6.

2 Grid Model

2.1 Resources

A grid is composed of a set of computational resources $\mathcal{C}$ and a set of storage resources $\mathcal{S}$. Each element $c \in \mathcal{C}$ ($s \in \mathcal{S}$) has a processing (storage) capacity of P_c (D_s) units, and the cost for using the resource during one unit of time is given by C_c^p (C_s^d). Additionally, we introduce a set of router resources $\mathcal{R}$, whose elements perform the familiar store-and-forward function of a packet router. We will assume throughout this paper that the forwarding capacity of each router is sufficient for all traffic sent through it. Note that the computational and storage resources also have this router functionality. Since all these resources are connected over a network, we introduce a set of edge resources $\mathcal{E}$. Likewise, an edge resources offers a bandwidth capacity of B_e units, and has a usage cost of C_e^b. If

*Research Assistant of the Fund Of Scientific Research - Flanders (F.W.O.-V., Belgium)

†Postdoctoral Fellow of the Fund of Scientific Research - Flanders (F.W.O.-V., Belgium)

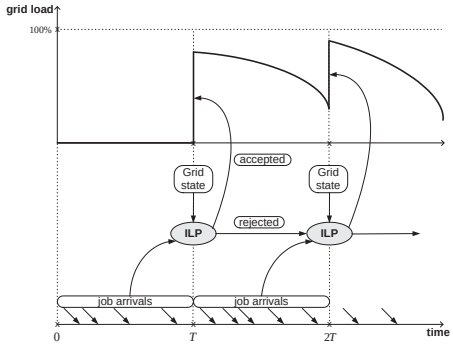

Figure 1: Periodic scheduler

we let $\mathcal{V} = \{\mathcal{C}, \mathcal{S}, \mathcal{R}\}$ be the set of nodes, and $\mathcal{E}$ be the set of edges, then it is straightforward to model a grid infrastructure as the directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

2.2 Jobs

Let $\mathcal{J}$ be the set of jobs. Each job $j \in \mathcal{J}$ is characterized by its processing rate p_j, the size of the input (output) datasets d_j^r (d_j^w), and the input (output) bandwidth b_j^r (b_j^w). Also, each job has a budget w_j, which will be payed when the job is accepted for execution. If we suppose that the input (output) bandwidth suffices to complete the input (output) data transfer, we are able to deduce the running time of the job as $t_j = max(\frac{d_j^r}{b_j^r}, \frac{d_j^w}{b_j^w})$.

3 Scheduling Algorithms

3.1 Operation

Fig. 1 details the operation of our scheduler. Scheduling is performed at periodic instants (period length T) in time, which allows us to react to the grid's changing state. The ILP model incorperates two functions: job admission control and resource assignation. Thus, a solution of the ILP model tells us whether or not a job is accepted for execution, and if so, which resources it may use. Accepted jobs are all started immediately and at the same time; rejected jobs are transferred automatically to the next scheduling round. Since no jobs can be started in between scheduling rounds, a job will occupy its assigned resources during a whole number of period lengths. In effect, resources are reserved for the duration of $k_j = \lceil \frac{t_j}{T} \rceil$ periods, and not for the real running time of a job.

3.2 Variables

In each scheduling round, we introduce three sets of binary variables:

1. $x_j = 1 \iff$ job j is executed
2. $y_{jc}^p = 1 \iff$ job j uses CR c
   $y_{js}^r = 1 \iff$ input of job j stored on SR s
   $y_{js}^w = 1 \iff$ output of job j stored on SR s
3. $z_{je}^r = 1 \iff$ input of job j uses edge e
   $z_{je}^w = 1 \iff$ output of job j uses edge e

3.3 Objective Function

The following function expresses the cost associated with the execution of a task:

$$\begin{aligned} C_j \quad = \quad & \alpha \cdot k_j \cdot \sum_{c \in \mathcal{C}} (C_c^p \cdot p_j y_{jc}^p) + \\ & \beta \cdot k_j \cdot \sum_{s \in \mathcal{S}} (C_s^d \cdot (d_j^r y_{js}^r + d_j^w y_{js}^w)) + \\ & \gamma \cdot k_j \cdot \sum_{e \in \mathcal{E}} (C_e^b \cdot (b_j^r z_{je}^r + b_j^w z_{je}^w)) \end{aligned}$$

The parameters α, β and γ ($0 \leq \alpha, \beta, \gamma \leq 1$) allow us to assign priorities to specific resource types. However, we kept all parameters constant and equal to 1 in our simulations. We now define the objective function:

$$\sum_{j \in \mathcal{J}} (w_j \cdot x_j - C_j).$$

The proposed objective function expresses the *profit* a scheduler will make; clearly, our scheduler is interested in maximizing this function.

3.4 Constraints

The first set of constraints (Eq. 1) express that an accepted job will use *exactly* one computational resource, while Eq. 2 limits the amount of usable capacity (analogous constraints are necessary for storage resources):

$$\forall j \in \mathcal{J} : \sum_{c \in \mathcal{C}} y_{jc}^p \quad = \quad x_j \tag{1}$$

$$\forall c \in \mathcal{C} : \sum_{j \in \mathcal{J}} p_j y_{jc}^p \quad \leq \quad P_c \tag{2}$$

Edge resources require an analog to Eq. 2, and a constraint which states that an accepted job uses *at least* one edge resource (Eq. 3). Additionally, the association between edges and nodes is made in Eq. 4 (analogous equations for z_{je}^w):

$$\forall j \in \mathcal{J}, \forall e \in \mathcal{E} : z_{je}^r \quad \leq \quad x_j \tag{3}$$

$$\forall j \in \mathcal{J}, \forall u \in \mathcal{V} :$$

$$\sum_{\substack{v \in \mathcal{V} \\ (u,v) \in \mathcal{E}}} z_{j(u,v)}^r - \sum_{\substack{v \in \mathcal{V} \\ (v,u) \in \mathcal{E}}} z_{j(v,u)}^r = \begin{cases} -y_{ju}^p, & \text{if } u \in \mathcal{C} \\ +y_{ju}^r, & \text{if } u \in \mathcal{S} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

We used the well-known dual simplex method to obtain solutions of our ILP-formulated problem.

resource type	*capacity*	*cost per period*
computational	500 MFLOPS	1
storage	6 Gb	1
network	155 Mbps	1

Table 1: Resource capacities and costs

parameter	*input*	*output*
processing power (MFLOPS)	50	
storage space (Mb)	300	300
bandwidth (Mbps)	20	20
budget	1000	

Table 2: Job requirements

4 Evaluation Setup

4.1 Topology

We used the Waxman model to generate 10 random network topologies. A network consists of 15 nodes, each node having a connectivity of 2. We then randomly selected 5 computational, 5 storage and 5 router resources. All resource characteristics are held constant; Table 1 summarizes their capacities and costs.

4.2 Jobs

All submitted jobs have identical characteristics; these are detailed in Table 2. It follows that the execution time of a job is $t_j = \frac{300}{20} = 15$ time units (see Section 2.2). Also note that the assigned budget is sufficient for execution, and that each computational resource can run at most 10 jobs concurrently. The same number of jobs can also be supported by one storage resource if both input and output of all jobs is assigned to the resource.

Job requests are generated at computational resources, and the arrivals follow a Poisson process. As it is undesirable to assign the same average number of arrivals to all arrival sites[1], we proceeded as follows. First, we established the maximum average number of arrivals $(\lambda T)_{max}$ for all sites. Then, we assigned each site its average number of arrivals by selecting uniformly from the interval $[2, (\lambda T)_{max}]$. Finally, each site generated job requests at their assigned average during each period.

4.3 Scheduler

We performed evaluations for 10 random topologies, executing 5 runs for each topology. All evaluations (except those in Section 5.3) have a scheduling period of $T = 20$ time units, which means that all jobs can run to completion within the period they were started. Simulation time is 10 scheduling periods in all cases. Jobs which are not accepted at the end of these 10 periods, are not taken into consideration for the calculation of the average queueing time.

[1] this would give rise to a symmetrical scheduling problem

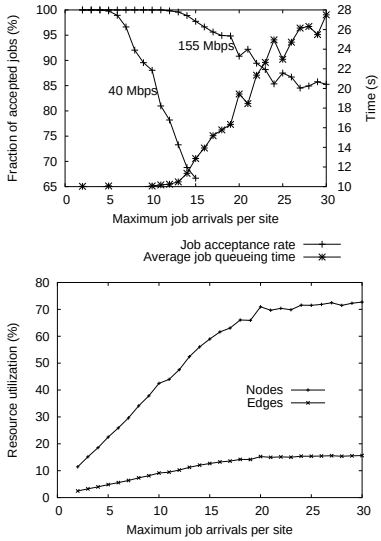

Figure 2: Influence of $(\lambda T)_{max}$

5 Simulation Results

5.1 Task Arrivals

Fig. 2 shows the acceptance rate for different values of $(\lambda T)_{max}$ (as defined in Section 4.2). Satisfying all job requests is possible for much higher values of $(\lambda T)_{max}$ in the case of $B_e = 155$ than for $B_e = 40$ Mbps. The average queueing time (taken for $B_e = 155$) shows an increasing trend for larger values of $(\lambda T)_{max}$. For a low number of arrivals, we get an average queueing time of 10 time units (50% of the period length), which means that all jobs get executed at the first possible scheduling round. Fig. 2 also shows the utilization of both node and edge resources; the node utilization for $B_e = 155$ Mbps approaches 75%, which is exactly the ratio of the job execution time and the period length.

5.2 Network Bandwidth

We start by establishing $(\lambda T)_{max} = 10$. Fig. 3 shows both the acceptance rate and the average job queueing time. We see here that a minimum of 40% of 155 Mbps is necessary to support the given load; going below this value drastically reduces performance. Also, the node utilization is constant and a little over 40% (see also Fig. 2 for $(\lambda T)_{max} = 10$) when at least 40% of 155 Mbps is available.

5.3 Period Length

As before, we established $(\lambda T)_{max} = 10$. Fig. 4 shows the acceptance rate for different values of the

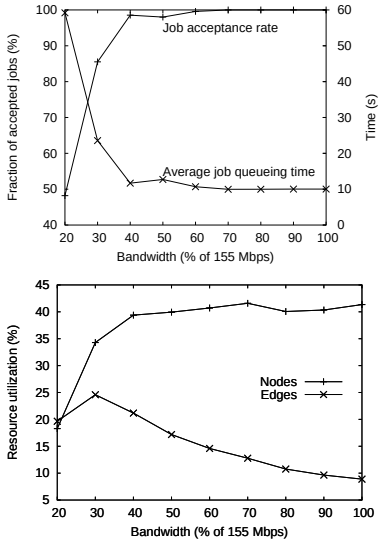

Figure 3: Influence of bandwidth

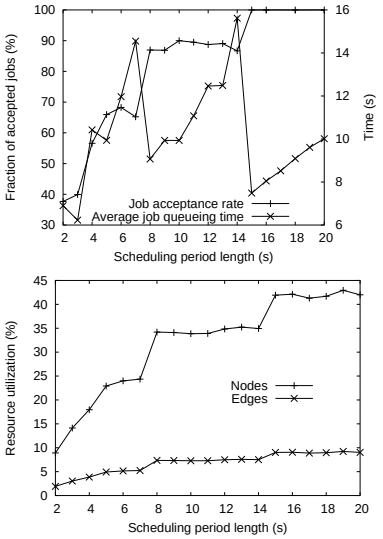

Figure 4: Influence of period length

period length, in which we see a step-like function emerge. As long as the period is greater than or equal to the running time of jobs, all jobs can be accepted. Taking the period length smaller than the job running time, implies resources will be reserved for two periods, although the resources are not utilized until the end (see also Section 3.1). We see the same behaviour when the period length equals whole fractions of the job running time. This also explains the spiked pattern of the average job queueing time, where optimal values are obtained by taking the period length equal to whole fractions of the job running time.

6 Conclusions and Future Work

In this paper we discussed the need for Grid scheduling algorithms which support network constraints. To this end, we proposed an Integer Linear Programming model, which accurately models compute, storage and network resources, and has support for economic parameters. We also showed how this ILP model can be incorporated in a Grid scheduling framework, and presented evaluations of our algorithms. We used well-known performance parameters, such as acceptance rate, average job queueing time and resource utilization.

Our results show that our scheduler behaves as expected under a broad range of job request loads. Still, even when much of the grid's resources are used, the scheduler optimally places new jobs. We also investigated the effect of available bandwidth in the network and determined a cut-off value; staying above this value wil always yield acceptable performance. Finally, we studied the effects of the scheduling period length, and clearly showed the existence of optimal values.

Further research will focus on the impact of changing job requirements, for instance by introducing several job request categories. Furthermore, we will investigate the choice of the scheduling instants, and try to gain insight in the effect of the economic parameters in our model. Given the limited scalability of ILP-formulated problems, we ultimately wish to extend our algorithms and develop a decentralized grid scheduler.

References

[1] I. Foster, C. Kesselman, "*The Grid: Blueprint for a Future Computing Infrastructure*", 1999.

[2] European DataGrid project (EDG), http://www.eu-datagrid.org.

[3] G.L. Nemhauser, L.A. Wolsey, "*Integer and Combinatorial Optimization*", John Wiley & Sons, 1988.

[4] H.A. James, K.A. Hawick, P.D. Coddington, "*Scheduling Independent Tasks on Metacomputing Systems*", Proc. of Parallel and Distributed Computing Systems, Aug 1999.

[5] V. Hamscher, U. Schwiegelshohn, A. Streit, R. Yahyapour, "*Evaluation of Job-Scheduling Strategies for Grid Computing*", Proc. of High Performance Computing, 2000.

[6] K. Ranganathan, I. Foster, "*Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids*", Journal of Grid Computing, Vol. 1, pp. 53–62, 2003.