Scheduling network requests with transmission window

Loris Marchal, Yves Robert, Pascale Vicat-Blanc Primet, Jingdi Zeng

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

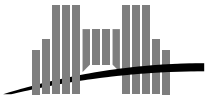
Scheduling network requests with transmission window

Loris Marchal,
Yves Robert,
Pascale Vicat-Blanc Primet,
Jingdi Zeng

July 2005

Research Report N° 2005-32

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : lip@ens-lyon.fr



Scheduling network requests with transmission window

Loris Marchal, Yves Robert, Pascale Vicat-Blanc Primet, Jingdi Zeng

July 2005

Abstract

We consider the problem of bulk data transfers and bandwidth sharing in the context of grid infrastructures. Grid computing empowers high-performance computing in a large-scale distributed environment. Network bandwidth, which makes the expensive computational and storage resources work in concert, plays an active role on performance. Due to specific traffic patterns, network topology and application scenarios, bandwidth sharing encounters new challenges. From this perspective, this research report looks at bulk transfers among computing and storage elements. Referred to as short-lived, transfer requests with transmission window and volume are scheduled in the network. By manipulating the transmission window, the request accept rate and network resource utilization are to be optimized. The formulated optimization problem is proven NP-complete. Associated with proposed heuristics, simulations are carried out to study each bandwidth sharing strategy and its application scenarios. A tuning factor, that allows adaptation of performance objective, is introduced to adjust network infrastructure and workload.

Keywords: grid computing, network bandwidth sharing, online scheduling, optimization, transmission window, scheduling window

Résumé

Nous considérons le problème du transfert de données de grande taille et du partage de bande passante dans les grilles de calcul. L'utilisation de telles grilles permet de déployer des calculs dans un environnement distribué à grande échelle pour obtenir de grandes performances. La bande passante du réseau qui interconnecte les ressources de calcul et de stockage a un impact critique sur les performances. À cause de la spécificité des transferts, de la topologie du réseau et des applications sous-jacentes, le partage de bande-passante doit s'adapter à de nouveaux défis. Nous nous concentrons ici sur les transferts de données de grande taille entre éléments de calcul et de stockage. Nous cherchons à ordonnancer sur le réseau des requêtes de tels transferts munies d'une fenêtre de transmission. Nous exprimons le problème d'optimisation correspondant et montrons qu'il est NP-complet. Nous proposons des heuristiques pour le résoudre, et menons à bien des simulations pour étudier chaque politique de partage de bande passante. Nous introduisons un coefficient de calibrage qui permet d'adapter l'objectif de performances pour ajuster l'ordonnancement à l'infrastructure réseau et à sa charge.

Mots-clés: Calcul sur la grille, ressources de communication, partage de ressources, ordonnancement à la volée, optimisation, fenêtre de transmission, fenêtre d'ordonnancement

1 Introduction

As large-scale computing and data management emerges in scientific and business areas, cost-effective, scalable, high performance computing infrastructures are called for. Moving one step further from processor clusters, grid computing is a promising technology that brings together large collection of geographically distributed resources (e.g., computing, visualization, storage, information, etc.) to build a very high-performance computing environment for data-intensive or computing-intensive applications [1]. In data grid applications, like experimental analyzes and simulations in high-energy physics, climate modeling, earthquake engineering, and astronomy, massive datasets must be shared by a community of hundreds or thousands of researchers distributed worldwide. These researchers transfer large subsets of these datasets to local sites or other remote resources for processing. Consequently, it has been recognized that high-performance, distributed data-intensive applications require reliable, efficient, and predictable data transfer as a fundamental service.

Data transfer protocols [2], which extend the standard FTP protocol to include a superset of the features offered by various grid storage systems currently in use, and that provide grid security and parallel, striped, partial, and third-party transfers have been proposed. However, when bottleneck are tight and when links present large bandwidth delay product, these solutions suffer heavily from TCP limitations. Apart from poor and unpredictable performance, they have also to deal with failed transfers that may be very frequent in congested situations. While TCP/IP protocols are improved to work with grid applications, they still conserve the max-min fair bandwidth sharing philosophy of the Internet that does not completely fit in the grid context.

Indeed, bandwidth sharing and transfers scheduling in grids have to be coupled with the management of other types of grid resources. Effective scheduling of jobs in large scale distributed system is complex and network bandwidth has been identified as one of the primary parameters that affect the performance [3]. The completion time of typical datagrid applications is given by the sum of the execution time and the time taken to transfer the data they need. In most cases, data transfer time often dominate in completion time. For bulk data grid applications, moving terabytes (and sometimes even petabytes) of data in the shortest possible amount of time is then of great interest. Network bandwidth sharing then surfaces as a part of the grid resource management. In the rest of the paper, as moving data is easier than distributing and deploying application codes, it is assumed that scheduling algorithms allocate computing (i.e., CPUs) and storage (i.e., memory and disks) resources first and generate output as data transfer request with time window and volume.

This paper proposes original solutions to control bandwidth sharing considering the specificity of the data grid context. Grid network resources are managed to ensure bandwidth reservation of dataset movements. Three goals are pursued: 1) improving data transfer time predictability, 2) enhancing transfer reliability, and finally 3) improving transfer performance. A lightweight and easy-to-deploy control plane that is complementary to the data plane is introduced.

A data grid is modeled as a set of sites, each comprising a number of processors and storage. Sites are connected to an over-provisioned core interconnection network. Indeed, the underlying communication infrastructure of grids is a complex interconnection of enterprize domains and public networks that exhibit potential bottlenecks and varying performance characteristics. For instance, considering grid hosts may generate large flows through their gigabit interfaces, the interfaces between private domains and the network core may become bandwidth sharing bottlenecks. The sharing is carried out at network edge of a fully-meshed grid inter-network. The proposed system model hides the packet-level traffic dynamics inside discrete data transfer requests, thus greatly reduces the complexity of the system.

The rest of the paper is organized as follows. Section 2 summarizes and explains our results. Section 3 presents related work. Section 4 gives the system model and defines optimization problems of bandwidth sharing. Heuristics and simulation results are given in Sections 6 and 7, respectively. The report concludes in section 8.

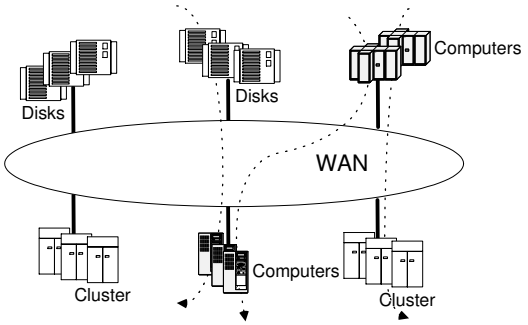


Figure 1: Ingress and egress points of the network as potential bottlenecks.

2 Our Results

We define the bandwidth sharing optimization problem. Each transfer request corresponds to a data flow with limited time duration. Given the capacity of the grid network access points and the maximum constant transfer rate an application can support, the overall requested bandwidth at any access point should be within the capacity of the natural aggregation point. Following the concept of "what enters the network shall be able to leave the network", the achieved results shall guarantee absolute bandwidth all along the path. Two optimization goals apply. One is the ordinary request accept rate, given the capacity of the network access points. The other is the refined request accept rate, provided with a tuning factor f that can reduce the time a request occupies the network resource. Between the originally requested bandwidth and the maximum transfer rate of the application, this factor reflects a margin the eventually assigned bandwidth has.

We analyze the complexity of the formulated optimization problem. From the reduction of the well-known NP-complete 3-Dimensional Matching (3-DM) problem, we show that the problem is NP-complete. Heuristics are then needed to obtain the solution.

Two types of online bandwidth sharing strategies are studied. Greedy heuristics follow the 'first come first serve' policy and are similar to a classical RSVP allocation strategy. Interval-based heuristics schedule bandwidth on a time interval basis. For both types of heuristics, ordinary request accept rate and a refined accept rate based on a bandwidth tuning factor are investigated through simulations.

Simulations show that greedy and interval based heuristics have similar performance when the network is not heavily loaded. Results show an average of an acceptance rate of 50% (ie. with bandwidth guarantee) for both strategies. The advance knowledge of requests does not improve the system. In a busy network as the one discussed in this report, interval-based approach improves a lot the accept rate while greedy strategy have an acceptance rate less than 20%. Furthermore, we show that the longer the interval, the better the accept rate. Longer intervals imply chances of having a better knowledge and more requests to schedule; it thus provides more room for the scheduling algorithm to optimize the bandwidth sharing. With large scheduling windows, more than 50% of bandwidth requests for bulk data transfers will be guaranteed, while it has been observed that in such overloaded context, concurrent high speed TCP flows have great difficulties in obtaining bandwidth and the largest flows suffer denial of service. In these conditions, bulk transfers often fails before ending. The proposed control may improve transfer reliability (goal 2) while insuring predictability of transfer time (goal 1).

We study the behavior of the tuning factor f through simulations. Between the value of 0 and 1, the tuning factor supplies requests an opportunity of actually obtaining more bandwidth for improving transfer duration (goal 3) and then application completion time. We understand that by pushing requests out of the network at an earlier time, the network may accommodate more requests in the future. Moreover, assigning more bandwidth to each request will certainly decrease the number of accepted requests. Simulation results presented in the report illustrate the dynamic between the tuning factor and the refined request accept rate. Both greedy and interval-based strategy take advantage of this factor under very overloaded conditions

as well as very underloaded conditions. In these conditions, allocating 80% of the maximum bandwidth improve the acceptance rate by a factor $0,2$ (linear with $(1-f)$) while a value of $f=0,5$ gives only a gain of $0,3$. This tuning factor enables the grid manager to adjust the global system with its own characteristics and the actual workload without modifying the bandwidth allocation strategy.

3 Related Work

How to allocate bandwidth to flows is a central issue in networking. Bandwidth sharing in IP networks has been well studied [4]. They are different from those of grid applications from several points of view. In the Internet, the source access rate are generally much smaller ($c=2Mbit/s$ for DSL lines) than the bottleneck capacity ($C=2,5Gbit/s$, say) and the link is not a bottleneck until demand attains around 99% of link capacity. In such an environment, max-min fairness, that is giving all flows the opportunity to make use of all available capacity in a "fair" way, is the goal of statistical bandwidth sharing strategies. But it has been known that in overloaded networks, performance deteriorates rapidly. Pro-active admission control is then prone to preserve performance. Admission control and reservation in the Internet have been studied for real-time traffic and generally with immediate reservation, that is, QoS takes effect immediately and remains in effect for an indefinite duration. Consequently, traditional admission control and reservation algorithms [5] adopt greedy strategies. In grid context, QoS guarantees is for TCP-dominated traffic and a certain degree of isolation is required between connections in order to support performance guarantees without precluding multiplexing. Grid context presents also a great difference in terms of transfer duration (hours, days...) that are bounded and are despite several order greater than that of the Internet. Flow transfer reliability is a very important issue as other grid resources (CPUs, disks) have been scheduled and a large amount of resources could be wasted when long transfer failure occurs. Finally, the grid network exhibits a specific topology, that is, heterogeneous and highly hierarchical. Ingress/egress links act both as natural aggregation points and constitute expected overloaded points as their capacity is in the same order of the access rate of the sources. Similar topology and bandwidth sharing problem are analyzed at different rate scales, in radio access networks. But in grids we have to consider both sides of the network and the load matrix is given.

The fairness issue between short and long TCP flows [6], that is, between mice and elephants, gained wide attention. Besides, the sharing is closely coupled with routing path search. The work in this report, however, assumes that grid bulk data are separated from the rest of the traffic (mice). TCP/IP protocols [7] have been adapted to carry high-volume grid data applications over long distance. The reliability, RTT fairness, TCP-friendliness issues are in the center of the investigation. These TCP enhancements for large bandwidth-delay product paths focus on Internet context. However, we consider that the work on end-to-end protocol improvement could be of great interest in this controlled context. The routing path search has also been integrated into the picture. This report looks at grid network access points where the traffic enters and leaves the network. The request have a predefined route from source to destination in this topology. The performance predictability is of more interest here.

In grid community, the Globus Architecture for Reservation and Allocation (GARA) introduced the idea of advance reservations and end-to-end management for QoS on different type of resources (bandwidth, storage, and computing) [8]. In line with this direction, the report further explores the optimization of bandwidth sharing given the specified grid network topology and traffic pattern. A similar bandwidth sharing problem has been investigated in [9]. Although both targeting at resource requests with transmission time windows, this report tackles optimal resource sharing, instead of investigating on impacts of the percentage of book-ahead periods and that of malleable reservations on the system. The physical characteristics of optical medium makes it an excellent candidate for supporting grid bulk data applications [10]. Existing work centers on the feasible network architectures. Assuming a system model that complies to optical domain topology and conditions, however, the bandwidth sharing mechanisms of this report can be deployed as part of the optical intelligent management system.

4 System Model and Problem Definition

The system is a collection of grid sites interconnected over a well-provisioned wide area network. The network core is assumed to have ample communication resources [10]. The aggregated capacity of a site is larger than the capacity of its access point (i.e., the router), and the capacity of the network core is larger than the aggregated capacity of all access points.

Given a set of transmission requests, an ingress point is where the traffic requires to enter the network, and an egress point is where the traffic requires to leave the network. These points, as depicted in Fig. 1, are where potential resource bottlenecks present.

Different from classical concept of flows that lasts indefinite time, flows in this report correspond to finite-size large data transfer. Flows arrive at the network edge according to a Poisson distribution, and each flow is associated with a source and a destination. Flows are unidirectional, given the fact that grid traffic volume between two users is often asymmetrical.

4.1 Resource requests

Bandwidth requests can be long-lived or short-lived. Long-lived requests correspond to indefinite flows between grid users, and short-lived requests represent discrete data transfer tasks. The scheduling of short-lived requests can be difficult, due to their flexible time windows and thus flexible bandwidth assignments.

We use the following notations:

- a set of requests $\mathcal{R} = \{r_1, r_2, \ldots, r_K\}$.
- a set of ingress points $\mathcal{I} = \{i_1, i_2, \ldots, i_M\}$, with $B_{in}(i)$ as the capacity (i.e., bandwidth) of ingress point $i \in \mathcal{I}$.
- a set of egress points $\mathcal{E} = \{e_1, e_2, \ldots, e_N\}$, with $B_{out}(e)$ as the capacity (i.e., bandwidth) of egress point $e \in \mathcal{E}$.
- each request has a required transmission window of $[t_s(r), t_f(r)]$, and an assigned transmission window of $[\sigma(r), \tau(r)]$ when accepted.
- each request has its volume $vol(r)$ specified either in Bytes or other meaningful units.
- each request has the transmission limit of its attached host $MaxRate(r)$.
- each request, if accepted, has an assigned bandwidth $bw(r)$.

If request r is accepted at time $\sigma(r) = t$, both points $ingress(r)$ and $egress(r)$ devote a fraction of their capacity, that is, $bw(r)$, to request r from time t to time $\tau(t) = t + \frac{vol(r)}{bw(r)}$.

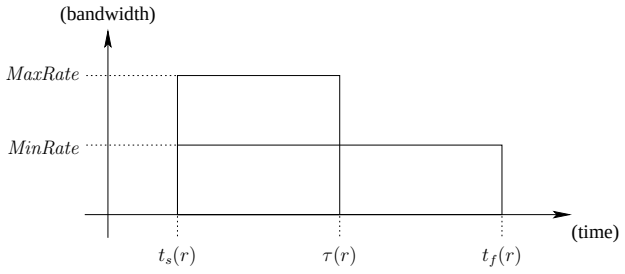


Figure 2: The flexible bandwidth assignment.

The flexibility of bandwidth assignment is illustrated as in Figure 2. Without the loss of the generality, the requested transmission starting time is kept as it is, that is, the assigned starting time is $\sigma(r) = t_s(r)$.

Provided with a manipulatable finishing time $t_f(r)$, the assigned bandwidth $bw(r)$ lies in the interval of $[MinRate(r), MaxRate(r)]$. $MinRate(r)$ is determined by the requested time window, that is,

$$MinRate(r) = \frac{vol(r)}{t_f(r) - t_s(r)}.$$

Obviously, the assigned finishing time $\tau(r)$ should not exceed the value of the requested finishing time $t_f(r)$. Accordingly, we have

$$\tau(r) = \sigma(r) + \frac{vol(r)}{bw(r)} = t_s(r) + \frac{vol(r)}{bw(r)} \leqslant t_f(r)$$

and

$$bw(r) \geqslant MinRate(r)$$

Moreover, the capacity of ingress or egress points pose a limit on the number of scheduled requests. The resource sharing constraints are then stated as the following:

$$\begin{aligned}
&\forall t, \quad \forall i \in \mathcal{I}, \sum_{\substack{r\in\mathcal{R},\ ingress(r)=i,\\ \sigma(r)\leqslant t<\tau(r)}} bw(r) \leqslant B_{in}(i)\\
&\forall t, \quad \forall e \in \mathcal{E}, \sum_{\substack{r\in\mathcal{R},\ egress(r)=e,\\ \sigma(r)\leqslant t<\tau(r)}} bw(r) \leqslant B_{out}(e),\\
&\forall r, \quad MinRate(r) \leqslant bw(r) \leqslant MaxRate(r)
\end{aligned} \tag{1}$$

where $ingress(r) \in \mathcal{I}$ and $egress(r) \in \mathcal{E}$ are the ingress and egress point of request r, respectively.

4.2 Optimization objectives

Let $\mathcal{A}$ denote the set of accepted requests. The classical objective for performance optimization is the request accept ratio, i.e.,

$$\#accepted = \frac{|\mathcal{A}|}{|\mathcal{R}|}$$

Following this objective, scheduling algorithms grant accepted requests the requested and also the minimum bandwidth $MinRate(r)$. Grid computing applications, furthermore, brings new elements into the decision making procedure. To fulfill a grid computing task, the CPU, storage, and network bandwidth resources have to work in concert. If a transmission task gets served faster than what it originally requests, it implies the earlier release of computing and storage resources. These resources will be returned to the available resource pool and can be used for other application requests. The application scenario of grid computing, therefore, suggests that assigning more bandwidth than requested to requests will benefit grid applications. Nevertheless, given the same amount of network resource, assigning more bandwidth will perhaps decries the accept rate. What is the relationship between increased assigned bandwidth and decreased accept rate, and what is the traded regarding the performance gain, is of interest.

Instead of assigning the requested bandwidth of $MinRate(r)$ to an accepted request, one may grant a fraction of the maximum bandwidth $MaxRate(r)$ that a grid user can utilize. Let f denote this fraction, then the value $f = 0.8$ guarantees 80% of $MaxRate(r)$ for each accepted request. The number of accepted requests, provided with the factor f, is to be maximized as follows:

$$\#guaranteed = \max\Big(r \in \mathcal{R},$$

$$bw(r) \geqslant \max(f \times MaxRate(r), MinRate(r))\Big)$$

The factor f may be adopted as a single value or adopted as a set of values corresponding to each request. When a set of values are used, they can be associated with the pricing policy and indicate quality

of service (QoS) of individual requests. When a single value is used, however, the factor f gives a reference on how much faster all requests are transmitted through the network. From a customer and service provider relationship perspective, customers now have two choices: they can stick with their requested network resource, that is, $MinRate(r)$, and have a better chance of being accepted at this time. they can also stand the risk of being rejected and try later, but take the advantage of being transmitted more quickly. Obviously, how much more quickly a request gets transmitted depends on the system load.

5 Problem complexity

Scheduling problems are known to be difficult, and the one addressed in this paper is no exception. Scheduling long-lived requests has already been proven NP-hard [11], even in the case of an uniform network (same bandwidth for each ingress and egress port). However, it was also shown in [11] that the optimal solution for scheduling uniform long-lived requests ($bw(r) = b$ for all $r \in \mathcal{R}$) can be computed in a polynomial time.

In this section, we show that scheduling uniform short-lived requests is NP-hard. This clearly shows the combinatorial nature of the problem, and the intrinsic complexity added by the possibility to route a request at different time-steps. We state the decision problem formally:

Definition 1 (MAX-REQUESTS-DEC). *Consider a problem-platform pair $(\mathcal{R}, \mathcal{I}, \mathcal{E})$ with uniform (unit-size) requests:*

$$\forall r \in \mathcal{R},\ bw(r) = MaxRate(r) = 1$$

For each request $r \in \mathcal{R}$, the transmission window $[t_s(r), t_f(r)]$ is known at the beginning (off-line problem). Given a bound K on the number of requests to satisfy, is there a feasible solution to such that at least K requests are accepted?

Theorem 1. MAX-REQUESTS-DEC *is NP-complete.*

It is worth noting that if the platform reduces to a single ingress-egress pair, the problem is polynomial (a greedy algorithm is optimal).

Proof. Clearly, MAX-REQUESTS-DEC belongs to NP; we prove its completeness by reduction from 3-DM (3-Dimensional Matching), a well-known NP-complete problem [12]. Consider an instance B_1 of 3-DM: given $X = \{x_1, x_2, \ldots, x_n\}$, $Y = \{y_1, y_2, \ldots, y_n\}$, and $Z = \{z_1, z_2, \ldots, z_n\}$ three disjoint sets of same cardinal n, and given a set of triples $T \subseteq X \times Y \times T$, does T contain a matching T', i.e. a set of n triples such that no two elements of T' agree in any coordinate? We build the following instance B_2 of MAX-REQUESTS-DEC:

- There are $M = n + 1$ ingress points and $N = n + 1$ egress points. For ingress points we let $B_{in}(i) = 1$ if $1 \leqslant i \leqslant n$ and $B_{in}(n + 1) = n - 1$. For egress points, we let $B_{out}(e) = 1$ if $1 \leqslant e \leqslant n$ and $B_{in}(n + 1) = n - 1$. Both points $B_{in}(n + 1)$ and $B_{out}(n + 1)$ are called *special*, while the $2n$ other points are called *regular*.

- There are $|T| + 2n(n - 1)$ requests in $\mathcal{R}$, and $bw(r) = 1$ for all $r \in R$. Each of the first $|T|$ requests is called *regular* and is associated to a triple in T, while the remaining $2n(n - 1)$ requests are called *special*.

- For each triple $(x_i, y_j, z_k) \in T$, we define a regular request r as follows: we let $B_{in}(r) = i$, $B_{out}(r) = j$ and $[t_s(r), ft(r)] = [k, k+1]$. In other words, there is no flexibility for regular request; if accepted, r must be scheduled at time $\sigma(r) = k$.

- Special requests involve one special point and are flexible. More precisely, for each ingress point i we define $n-1$ identical requests r such that $B_{in}(r) = i$, $B_{out}(r) = n+1$ and $[t_s(r), ft(r)] = [1, n+1]$. Similarly, for each ingress point e we define $n - 1$ identical requests r such that $B_{in}(r) = n + 1$, $B_{out}(r) = e$ and $[t_s(r), ft(r)] = [1, n + 1]$. Therefore, a special request can be scheduled at any time-step between 1 and n.

- Finally, we let $K = n + 2n(n-1)$.

The size of B_2 is polynomial (and even linear) in the size B_1. We have to show that B_1 has a solution if and only if B_2 has a solution.

Assume first that B_1 has a solution. Let T' be the matching. For each time-step k, $1 \leqslant k \leqslant n$, there is a single triple in T' whose last coordinate is z_k. Let (x_i, y_j, z_k) be this triple, and r its associated regular request. Together with r, we route $2(n-1)$ special requests at step k, one from each ingress point except i, and one to each egress point except j. Because T' realizes a permutation in the first and second coordinates of its triples, each regular point is active exactly $n-1$ times during the n scheduling steps. All regular points will thus accept all their $n-1$ special requests. Together with the n regular request (one per step), we have accepted K requests, hence a solution to B_2.

Conversely, assume now that B_2 has a solution. $K = n + 2n(n-1)$ request are accepted during the n scheduling steps. But at a given step, no more than $2n-1$ requests can be accepted, and this is only feasible if $2n-2$ of them are special requests. Therefore, at each step exactly $2(n-1)$ special requests and one regular request are accepted. Let T' be the set of the triples associated to these n regular requests, we claim that T' is the desired matching. By construction, no two triples of T' agree in the third coordinate. Assume that two triples would share the same first coordinate, say x_i. This means that ingress point i is activated at two different time-steps for two regular requests. At most $n-2$ special requests with ingress i will be accepted. But this is a contradiction, because all special requests are accepted (there are $2n(n-1)$ of them, and $2(n-1)$ are accepted at each step). For the second coordinate the reasoning is identical. We have found a solution to B_1. □

6 Heuristics

As proved in section 5, the optimization problem formulated in section 4 is NP-complete. Heuristics are then pursued to solve the problem.

Given that the starting time and finishing time are not violated, that is, $t_f(r) > t_s(r) + \frac{vol(r)}{bw(r)}$, the constraint on the assigned bandwidth is stated as $bw(r) \geqslant MinRate(r)$. Algorithms that adopt different bandwidth assignment policies, either granting $MinRate(r)$ to each accepted request, or ensuring $\max(MinRate(r), f \times MaxRate(r))$ for a prescribed tuning factor f, are introduced in the following sections. One major characteristic of all our proposed heuristics is that they are *on-line*. We take decisions either on the fly (on a pure greedy basis) or after a short delay (scheduling within each time interval). There is no need on the knowledge of the whole set of requests.

The heuristics can be classified according to the decision procedure:

- **Greedy–** Requests are accepted or rejected on a first-come first-serve basis. If two requests have the same arrival time $t_s(r)$, we schedule the one with the smallest $MinRate(r)$.
- **Interval-based–** Requests are accepted or rejected within consecutive time intervals. These intervals have the same length. The scheduler considers all the requests whose arrival time lies within the current interval. More requests are expected to be processed in longer intervals; this leaves more space for optimization, at the price of a longer response time for grid users.

6.1 Greedy heuristics

Both proposed greedy heuristics schedule requests as soon as they arrive. However, they assign different bandwidth in to each accepted request r: one is the minimum rate $MinRate(r)$ as originally requested by the user, the other one is a prescribed fraction $f \times MaxRate(r)$. The parameter f varies in experiments. To simplify the notation, we use

$$bw(r) \leftarrow \text{BANDWIDTHASSIGNMENTALGORITHM}(r),$$

where BANDWIDTHASSIGNMENTALGORITHM denotes any of the previous bandwidth assignment strategies.

GREEDY($\mathcal{R}, \mathcal{I}, \mathcal{E}$)
  $\mathcal{A} \leftarrow \emptyset$
  for each ingress i in $\mathcal{I}$ **do**
    $alloc\_ingress(i) \leftarrow 0$
  for each ingress e in $\mathcal{E}$ **do**
    $alloc\_egress(e) \leftarrow 0$
  for $t = t_{begin}$ to $t = t_{end}$ **do**
    if $t = t_f(r)$ for some request $r \in \mathcal{A}$ **then**
      {reclaim bandwidth allocated to r}
      $alloc\_ingress(ingress(r)) \leftarrow alloc\_ingress(ingress(r)) - bw(r)$
      $alloc\_egress(egress(r)) \leftarrow alloc\_egress(egress(r)) - bw(r)$
    if $t = t_s(r)$ for some request $r \in \mathcal{A}$ **then**
      {try to schedule r}
      $bw(r) \leftarrow$ BANDWIDTHASSIGNMENTALGORITHM(r)
      $i \leftarrow ingress(r)$
      $e \leftarrow egress(r)$
      if $(alloc\_ingress(i) + bw(r) \leqslant B_{in}(i))$ **and** $(alloc\_egress(e) + bw(r) \leqslant B_{out}(e)$
      then
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{r\}$
        $alloc\_ingress(i) \leftarrow alloc\_ingress(i) + bw(r)$
        $alloc\_egress(e) \leftarrow alloc\_egress(e) + bw(r)$
  return $\mathcal{A}$

Algorithm 1: The greedy heuristics.

The pseudo code of greedy heuristics is shown in Algorithm 1, where $alloc\_ingress(i)$ denotes the amount of bandwidth which is currently allocated for ingress $i \in \mathcal{I}$, and which should never exceed $B_{in}(i)$ (and similarly $alloc\_egress(e)$ for $e \in \mathcal{E}$). t_{begin} and t_{end} denote the times at which execution begins and ends. $\mathcal{A}$ is the set of accepted requests.

6.2 Interval-based heuristics

Interval-based heuristics do not schedule requests as soon as they arrive. Instead, they take decisions every time step t_{step}. The execution is thus divided into time intervals of length t_{step}. At the end of each interval $[t, t + t_{step}[$, scheduling decisions are taken for all *candidate* requests, i.e. request r whose arrival time lie in the interval: $t \leqslant t_s(r) < t + t_{step}$. As for the greedy heuristics, we keep track of the bandwidth $alloc\_ingress(i)$ and $alloc\_egress(e)$ already allocated on each ingress and egress ports. The first thing to do is to reclaim the bandwidth assigned to accepted requests r whose execution is finished in the previous interval, i.e. requests satisfying $t - t_{step} \leqslant t_f(r) < t + t_{step}$. Then we compute a *cost* associated with each candidate request. The intuition is to balance the resource assignments among access points. A request of high cost is likely to saturate its ingress or egress point, thereby hindering the possibility of scheduling more requests later.

The cost of a request r is computed as the following. Let $i = ingress(r)$, $e = egress(r)$, and $bw(r)$ the bandwidth assigned to r if accepted. As before, $bw(r)$ will be either the minimum rate $MinRate(r)$ or a prescribed fraction $f \times MaxRate(r)$. If r is accepted, the utilization rate of its ingress point i becomes $\frac{alloc\_ingress(i)+bw(r)}{B_{in}(i)}$, and that of its egress point e becomes $\frac{alloc\_egress(e)+bw(r)}{B_{out}(e)}$. We define the cost of r as the maximum value of these two quantities, that is,

$$cost(r) = \max\left(\frac{alloc\_ingress(i) + bw(r)}{B_{in}(i)}, \frac{alloc\_egress(e) + bw(r)}{B_{out}(e)}\right).$$

The candidate request with minimum cost will be accepted. See Algorithm 2 for a detailed description of the interval-based heuristics, where $\mathcal{C}$ denotes the set of candidate requests, and $\mathcal{A}$ the set of accepted requests.

INTERVAL($\mathcal{R}, \mathcal{I}, \mathcal{E}$)
  $\mathcal{A} \leftarrow \emptyset$
  for each ingress i in $\mathcal{I}$ **do**
    $alloc\_ingress(i) \leftarrow 0$
  for each ingress e in $\mathcal{E}$ **do**
    $alloc\_egress(e) \leftarrow 0$
  for $t = t_{begin}$ to $t = t_{end}$ by step t_{step} **do**
    {reclaim bandwidth of finished requests}
    for each request $r \in \mathcal{A}$ s.t. $t - t_{step} \leqslant t_f(r) < t$ **do**
      $alloc\_ingress(ingress(r)) \leftarrow alloc\_ingress(ingress(r)) - bw(r)$
      $alloc\_egress(egress(r)) \leftarrow alloc\_egress(egress(r)) - bw(r)$
    {determine set of candidate requests}
    $\mathcal{C} \leftarrow \emptyset$
    for each request $r \in \mathcal{R}$ s.t. $t \leqslant t_s(r) < t + t_{step}$ **do**
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{r\}$
      $bw(r) \leftarrow$ BANDWIDTHASSIGNMENTALGORITHM(r)
    {schedule candidate requests}
    $continue \leftarrow true$
    while ($\mathcal{C} \neq \emptyset$) and $continue$ **do**
      select r_{min} such that $cost(r_{min}) \leqslant cost(r)$ for all $r \in \mathcal{C}$
      where $cost(r) \leftarrow \max(\frac{alloc\_ingress(ingress(r))+bw(r)}{B_{in}(ingress(r))}, \frac{alloc\_egress(egress(r))+bw(r)}{B_{out}(egress(r))})$
      if ($cost(r_{min}) > 1$) **then**
        $continue \leftarrow false$
      else
        $\mathcal{C} \leftarrow \mathcal{C} \setminus \{r\}$
        $\mathcal{A} \leftarrow \mathcal{A} \cup \{r\}$
        $alloc\_ingress(ingress(r)) \leftarrow alloc\_ingress(ingress(r)) + bw(r)$
        $alloc\_egress(egress(r)) \leftarrow alloc\_egress(egress(r)) + bw(r)$
  return $\mathcal{A}$

Algorithm 2: The interval-based heuristics.

7 Simulations and discussions

In this section, simulations are carried out to illustrate and compare the performance of the heuristics given in the previous section.

The simulated grid network includes 10 ingress and 10 egress points with the capacity of 1GB/s. The volume of each request is randomly chosen between 100GB and 1TB. The transmission time varies from a couple of minutes to about one day, by randomly generating bandwidth requests between 10MB/s and 1GB/s. Request starting time is Poisson distributed, the parameter of this Poisson distribution is the average arrival time of the requests. The value of the parameter varies to obtain heavy loaded scenarios and less loaded cases. As described in Section 4.2, the optimization objective is the accept rate, that is, the number of accepted requests over the number of total requests.

A heavy loaded scenario is illustrated as in Figure 3. The average arrival time of requests vary from 0.1 to 5 seconds. The bandwidth assignment policy assigns $f \times MaxRate(r)$ with $f = 1$. The simulation results show that in a very loaded network, the interval-based heuristics achieves better accept rate than the greedy one. And for the interval-based algorithms, the larger the time interval, the better the request accept rate.

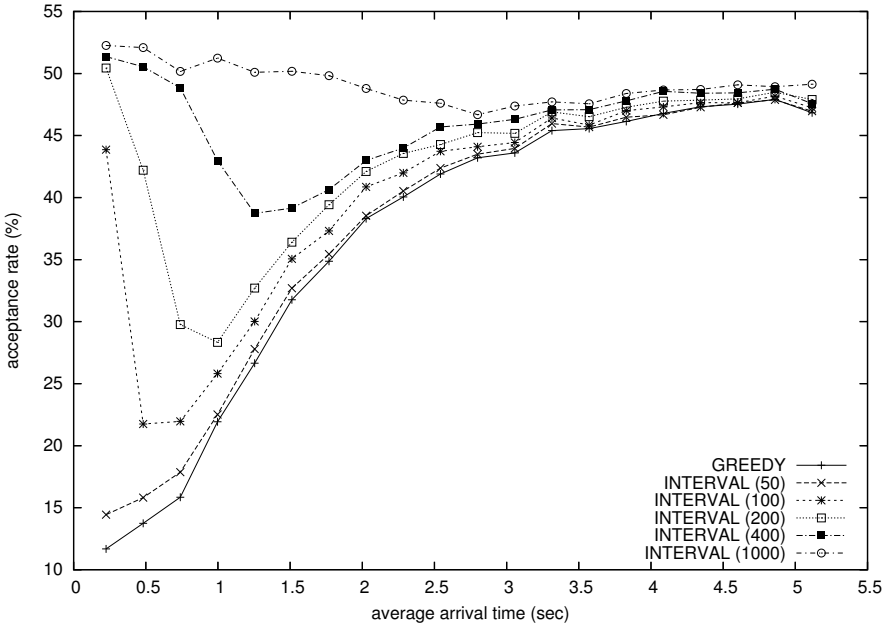


Figure 3: Comparison of Greedy and Interval-based (with different interval lengths) heuristics in heavy loaded context

Figures 4 and 5 present the performance of the Greedy heuristic. The network is less loaded (average arrival time goes from 3 to 20 seconds). Different bandwidth assignment policies apply: either the minimum bandwidth to every accepted request (MIN_BW), or a fixed ratio f of the maximum transmission rate $MaxRate$. As expected, a smaller bandwidth to each request results in more accepted requests, especially when the network is not too much loaded. This is no longer true, however, for heavy loaded networks. For example, assigning a request the maximum rate of the its user leads to a smaller transmission time, thus the corresponding ingress/egress points are freed to other requests more quickly. The same set of simulations is run for the interval-based heuristic, and the results are depicted in Figures 6 and 7. The same conclusions of the greedy heuristic hold, except that the slightly better results for small values of the average arrival time.

8 Conclusions

Network bandwidth sharing in data grids has been investigated in this report. With bottlenecks presented at the network edge, network bandwidth are scheduled based on the concept of what enters the network shall be able to leave the network. Referred to as short-lived requests, the data transfer requests of grids are scheduled with respect to optimizing the request accept rate. Proven NP-complete, the optimization solutions are pursued with heuristics. These algorithms are studied and compared by simulations.

Along with other protocols and interfaces, the bandwidth sharing strategies studied here are going to be integrated in grid network middleware of the Grid 5000 project, a large scale experimental grid testbed that gathers more than 3000 CPU in eight French sites to study Grid software. They may work closely with the scheduling of other resources, such as computational and storage. In future work we plan to investigate realistic scenarios with real grid input data and real applications. We want to examine how our bandwidth allocation strategies associated with enhanced TCP protocols for high speed networks achieve the transfer optimisation goals. We are also interested in exploring how these protocols would symmetrically better perform at very high speed and for long duration in un-congested long fat networks offered by our control algorithms.

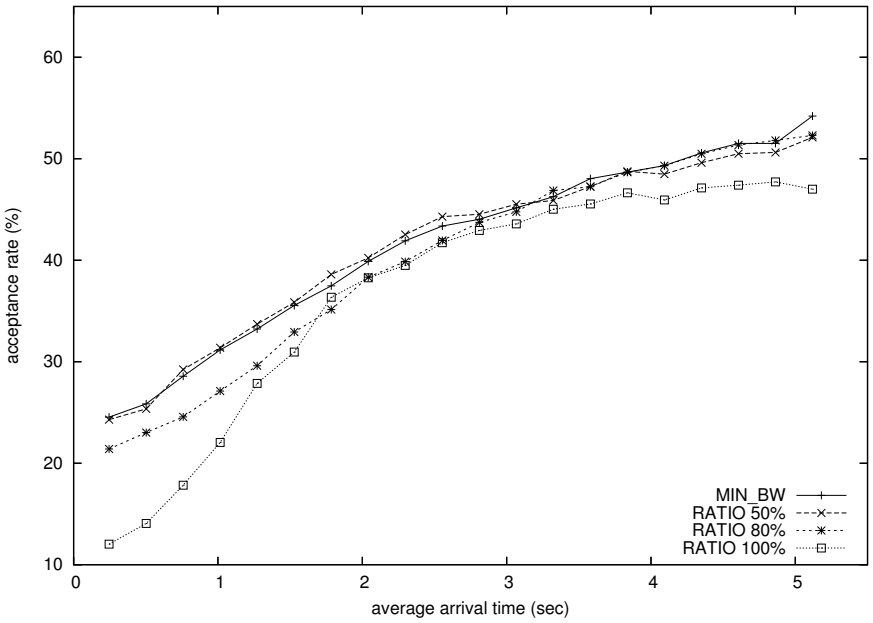


Figure 4: Performance of the Greedy heuristic with different bandwidth allocation policies (overload)

Acknowledgment

This work has been funded by the French ministry of Education and Research, INRIA, and CNRS, via ACI GRID's Grid5000 project and GRIPPS project.

References

[1] I. Foster, *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.

[2] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, "Data management and transfer in high performance computational grid environments," *Parallel Computing*, vol. 28, pp. 749–771, May 2002.

[3] K. Ranganathan and I. T. Foster, "Simulation studies of computation and data scheduling algorithms for data grids." *Journal Of Grid Computing*, vol. 1, no. 1, pp. 53–62, 2003.

[4] J. W. Roberts, "A survey on statistical bandwidth sharing." *Computer Networks: The International Journal of Computer and Telecommunications Networking archive*, vol. 45, pp. 319–332, June 2004.

[5] R. Braden, "Resource reservation protocol rsvp) version 1 functional specification," 1997.

[6] K. Avrachenkov, U. Ayesta, P. Brown, and E. Nyberg, "Differentiation between short and long tcp flows: Predictability of the response time." in *INFOCOM*, 2004.

[7] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control (bic) for fast long-distance networks." in *INFOCOM*, 2004.

[8] I. T. Foster, M. Fidler, A. Roy, V. Sander, and L. Winkler, "End-to-end quality of service for high-end applications," *Computer Communications*, vol. 27, no. 14, pp. 1375–1388, 2004.

[9] L. Burchard, H.-U. Heiss, and C. A. F. D. Rose, "Performance issues of bandwidth reservations for grid computing," in *Proc. IEEE the 15th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'03)*, Nov. 2003, pp. 82–90.

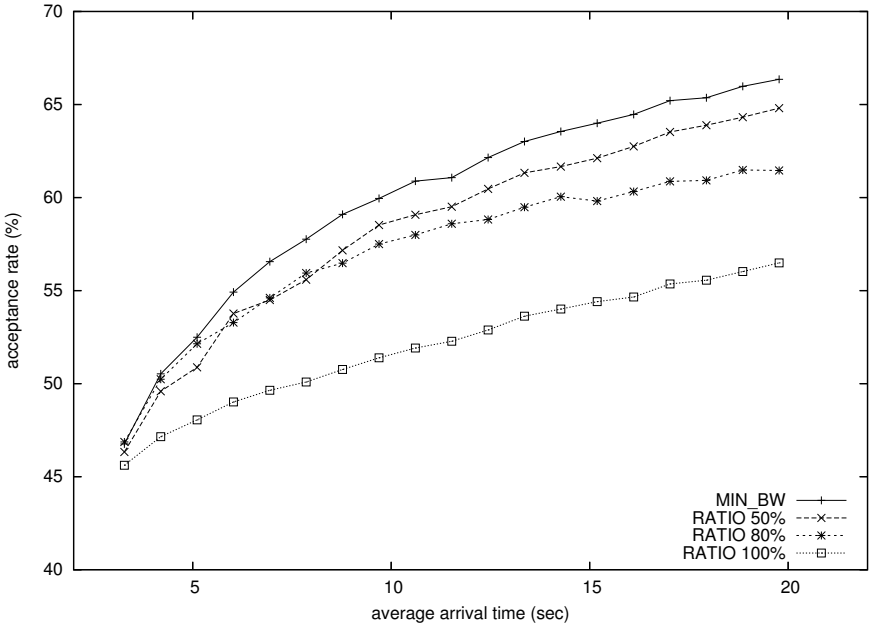


Figure 5: Performance of the Greedy heuristic with different bandwidth allocation policies (underload)

[10] L. L. Smarr, A. A. Chien, T. Defanti, J. Leigh, and P. M. Papadopoulos, "The optiputer," *Communications of the ACM special issue: blueprint for the future of high-performance networking*, vol. 46, pp. 58–67, Nov. 2003.

[11] L. Marchal, P. Primet, Y. Robert, and J. Zeng, "On network resource scheduling in grid networking," INRIA - ENS/LIP, Lyon, France, Tech. Rep., 2005.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.

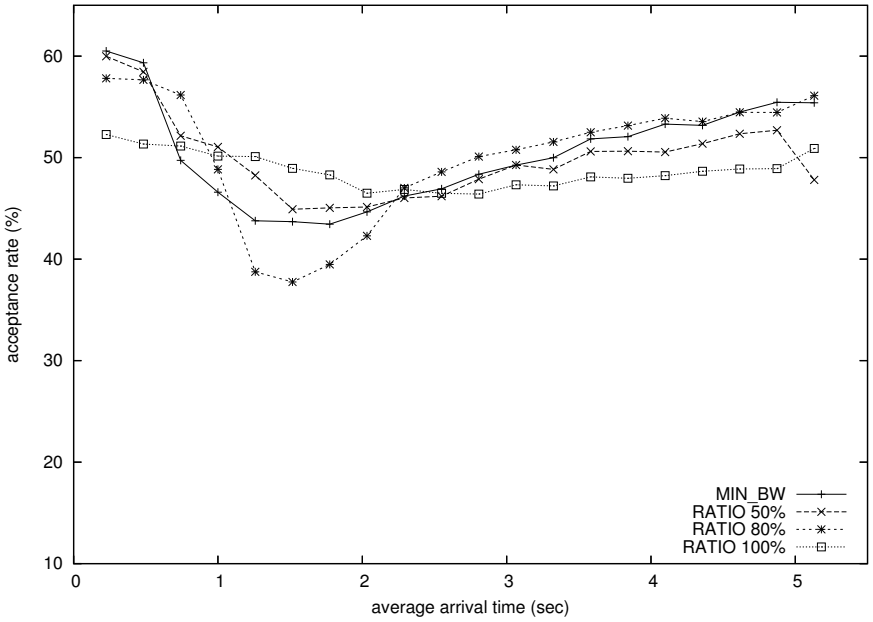


Figure 6: Performance of the Interval-based heuristic (with length 400) with different bandwidth allocation policies (overload)

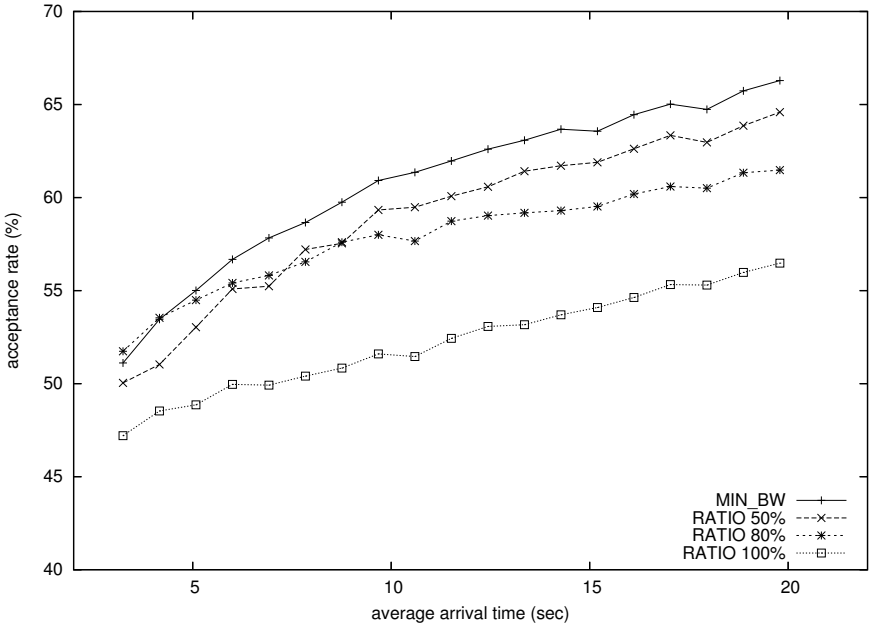


Figure 7: Performance of the Interval-based heuristic (with length 400) with different bandwidth allocation policies (underload)