Routing and Scheduling in Grids

Konstantinos Christodoulopoulos, Emmanouel (Manos) Varvarigos
Computer Engineering and Informatics Department, University of Patras, Patras, Greece, and
Research Academic Computer Technology Institute, Patras, Greece
e-mail: manos@ceid.upatras.gr

ABSTRACT
We propose QoS-aware scheduling algorithms for Grid Networks that are capable of optimally or near-optimally assigning computation and communication tasks to grid resources. The routing and scheduling algorithms to be presented take as input the resource utilization profiles and the task characteristics and QoS requirements, and co-allocate resources while accounting for the dependencies between communication and computation tasks.
Keywords: communication and computation utilization profiles, multicost routing and scheduling, grid computing.

1. INTRODUCTION

Grids introduce new ways to share computing power, storage resources and specific instruments between geographically distributed sites, the management of which requires scheduling at various levels [1]. The complexity of grid applications, the user requirements and the system heterogeneity would result in inefficient scheduling in the case of a manual procedure. Thus, sophisticated algorithms that take into account multiple optimization criteria have to be used. In order for Grid systems to be employed in real world commercial applications and demanding scientific experiments, end-to-end Quality of Service (QoS) is desirable. The main approach to providing end-to-end QoS is reservations and especially reservations that are performed “in-advance”. However, the complexity of the Grid management systems and the related algorithms increases considerably when temporal information is used.

The Globus Architecture for Reservation and Allocation (GARA) [2] is a framework for advance reservations that treats in a uniform way various types of Grid resources. Up until now, a number of algorithms that use advance reservations have been proposed for task scheduling on computation resources [3] [4]. The authors in [7] introduced the concept of routing and scheduling in communication resources. Some types of joint communication and computation problems have also been examined [5] [6].

In this paper we propose a multicost algorithm that jointly addresses the communication and computation scheduling problem in grid networks. Handling a grid task involves two successive steps: (i) the transfer of data from the scheduler or a data repository site, which we will call source, to the computation resource or cluster in the form of a connection with given rate or a data burst and (ii) the execution of the task at the cluster. In contrast to the solutions proposed in [5]-[6] our algorithm uses advance reservations for the time scheduling of the communication resources, in a way similar to [7]. Multicost algorithms have mainly been used for QoS routing problems [7]. A key difference to other multicost approaches is that the proposed algorithm uses the timeslot utilizations of the computation resources as cost parameters in the formulation.

We initially present an optimal scheme of non-polynomial complexity and by appropriately pruning the set of candidate paths we also give a heuristic algorithm of polynomial complexity. We evaluate the performance of the optimal algorithm and of its proposed polynomial-time heuristic variation using network simulation experiments, and compare it to that of algorithms that handle the computation or communication part of the problem separately. Our results indicate that when the tasks are cpu- and data-intensive important performance benefits can be obtained by jointly optimizing the use of the communication and computation resources, as our proposed algorithms do.

2. COMMUNICATION AND COMPUTATION UTILIZATION PROFILES

In a network that employs advance reservations, each node needs to keep a record of the capacity reserved on its outgoing links, as a function of time [9]. Assuming each connection reserves bandwidth equal to r for a given time duration, the utilization profile $U_l(t)$ of a link l with capacity C_l is a stepwise function with discontinuities at the points where reservations begin or end, and is updated dynamically with the admission of each new connection. To obtain a data structure that is easier to handle in an algorithm, we discretize the time axis in steps (timeslots) of duration τ_l and define the *binary r-capacity availability vector* $\hat{C}_l(r)$, abbreviated CAV, as the vector whose k-th entry is:

$$\left\{\hat{C}_l(r)\right\}_k = \begin{cases} 1, \text{ if } C_l - U_l(t) > r \\ 0, \quad \text{otehwise} \end{cases}, \text{for all } (k-1)\cdot\tau_l \le t \le k\cdot\tau_l, k = 1,2,...,u_l$$

where u_l is the dimension of the CAV. The data structures defined above can be useful in a number of network settings such as WDM networks with or without wavelength conversion, and Optical Burst Switched (OBS) networks. To keep a consistent formulation, we define the binary r-cluster availability vector $\hat{W}_m(r)$, which gives the number of CPUs that are free as a function of quantized time.

The discretization of the time axis results in some loss of information, and provides a tradeoff between the accuracy and the size of the maintained information. The discretization steps τ_l and τ_m used in the link and cluster utilization profiles, respectively, can be different in order to account for the different time scales in the reservations performed in these different types of resources. Note that the timeslot-based management of Grid resources is a well established and efficient way to manage utilization information (e.g. denoted as *timeslot table* in GARA [2]).

In a distributed architecture, each distributed scheduler maintains a "picture" of the utilization of all communication and computation resources, which can be different among the distributed schedulers. In our approach, this is done by maintaining a utilization database with link and cluster availability vectors for all resources. Update information is communicated to synchronize the locally maintained profiles with the actual utilization.

3. THE TASK ROUTING AND SCHEDULING PROBLEM UNDER STUDY

We are given a Grid infrastructure consisting of a network with links l of known propagation delays d_l and capacity C_l (bps), and a set M of clusters. Cluster $m\in M$ has W_m CPUs of a given processor speed C_m (e.g. MIPS). A task is created by a user with specific needs: input data size I (bits) and computational workload W (e.g. MI). The user communicates this information to distributed scheduler S. We assume that the input data are forwarded by the user to the scheduler S or are located at a data repository site R. Thus, S or R comprises the source of the input data. Also, S has (possibly outdated) information about the capacity availability vectors $\hat{C}_l$ of all links l, and the cluster-availability vectors $\hat{W}_m$ of all clusters M. We assume that there is an upper bound D on the maximum delay tasks can tolerate. Even when no limit D is given, we still assume that the dimension u_l and u_m of the link and cluster utilization vectors are finite, corresponding to the latest time for which reservations have been made. Given the previous information, we want to find a suitable cluster to execute the task, a feasible path over which to route the data, and the time at which the task should start transmission from the source and execution at the cluster, so as to optimize some performance criterion, such as the completion time of the task. Figure 1a presents an instance of the problem.

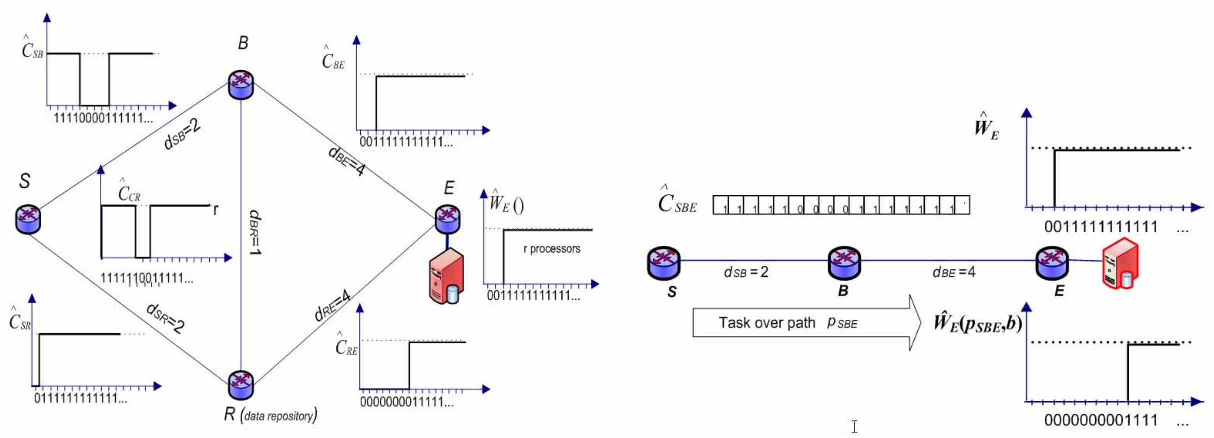


Figure 1. (a) An instance of the routing and scheduling problem under study, (b) calculation of the binary cluster availability vector $\hat{W}_m(p,b)$ over p.

3.1 Binary Capacity Availability Vector of a Path

For this and the following section we assume that the input data are located at S. To calculate the CAV of a path we have to combine the CAVs of the links that comprise it, by defining an associative operator '**&**', as described in [10]. For example, for the topology of Figure 2, the CAV of path p_{SBE}, consisting of links l_{SB} and l_{BE}, is

$$\hat{C}_p = \hat{C}_{SBE} = \hat{C}_{SB} \,\&\, \hat{C}_{BE} = \hat{C}_{SB} \oplus \mathrm{LSH}_{2\cdot d_{SB}}(\hat{C}_{BE}), \tag{1}$$

where $\hat{C}_{SB}$ and $\hat{C}_{BE}$ are the CAVs of links l_{SB} and l_{BE}, respectively, and LSH() defines the left shift of $\hat{C}_{BE}$ by $2\cdot d_{SB}$ (propagation delay of l_{SB} measured in τ_l units). Left shifting $\hat{C}_{BE}$ by d_{SB} purges the time periods that have expired to transfer utilization information from B to S, while left shifting it by another d_{SB} accounts for the propagation delay to forward the data from S to B.

3.2 Binary Cluster Availability Vector over a Path

We want to transmit data of duration $b=I/C_l$, where I is the data size, over the path p in order to execute the task at cluster m. We define $R_p(b)$ as the first position after which $\hat{C}_p$ has b consecutive ones. In other words, $R_p(b)$ is the earliest time after which data of duration b can be transmitted on path p. The earliest time that the task can reach cluster m is then given by $\mathrm{EST}(p, b) = R_p(b) + b + d_p$. The distributed scheduler S has a partial (and possibly outdated) knowledge of the cluster availability vector $\hat{W}_m$ of m. We define $\mathrm{MUV}_k(\hat{W}_m)$ as the operation of setting zeros (making unavailable) the first k elements of vector $\hat{W}_m$. Then, vector $\hat{W}_m(p,b) = \mathrm{MUV}_{\mathrm{EST}(p,b)}(\hat{W}_m)$ gives the time periods that S can schedule the task at cluster m over path p.

4. JOINT COMMUNICATION AND COMPUTATION TASK SCHEDULING ALGORITHM IN GRIDS

In what follows we present a multicost algorithm for the joint communication and computation scheduling of tasks.

4.1 Computing the Set of Non-Dominated Paths

In multicost routing, each link l is assigned a vector V_l of cost parameters, as opposed to the scalar cost parameter assigned in single-cost routing. In our initial formulation, the cost parameters of a link l include the propagation delay d_l of the link and its binary capacity availability vector $\hat{C}_l$, that is, $V_l = (d_l, \hat{C}_l)$, but they may also include other parameters of interest (such as number of hops, the number of executed tasks in a cluster, etc). A cost vector [11] can then be defined for a path p consisting of links $l \in p$, based on the cost vectors of its links:

$$V(p) = \underset{l\in p}{\odot} V_l \overset{def}{=} \left(\sum_{l\in p} d_l, \underset{l\in p}{\&} \hat{C}_l \right), \tag{2}$$

where **&** is the associative operator defined in Eq. (1).

We say that path p_1 dominates path p_2 for a given source-destination pair, if the propagation delay of p_1 is smaller than that of p_2, and path p_1 is available (at least) at all time intervals at which path p_2 is available. Formally:

$$\text{p1 dominates p2 (notation: p1} > \text{p2) iff } \sum_{l\in p_1} d_l < \sum_{l\in p_2} d_l \text{ and } \underset{l\in p1}{\&} \hat{C}_l \geq \underset{l\in p_2}{\&} \hat{C}_l, \tag{3}$$

where the vector inequality "≥" should be interpreted component-wise.

An algorithm for obtaining the set P_{n-d} of non-dominated paths from a given source to all destination nodes is given in [10] and [11], and is a generalization of Dijsktra's algorithm that only considers scalar link costs.

4.2 Polynomial Algorithm for Computing the Set of Non-Pseudo-Dominated Paths

A serious drawback of the algorithm described above is that the number of non-dominated paths may be exponential, and the algorithm is not guaranteed to finish in polynomial time. We define a new link metric, called the slot availability weight of the link, as *weight*($\hat{C}_l$), which represents the total number of 1's in the vector $\hat{C}_l$. The polynomial-time heuristic variation of the optimal multicost algorithm computes the set of non-pseudo-dominated paths following the same steps, but the domination relationship that is used to prune the paths is not Eq. (3) but a relationship based on the slot availability weights of the paths. By defining this pseudo-domination relationship an upper limit on the number of non-pseudo-dominated paths between the source and a given node is the dimension u_l of the capacity availability vectors. The reduced problem was proven to be polynomial in [7].

4.3 Obtaining the Set of Non-Dominated (path, cluster) Pairs

We define the cost vector of a (path, cluster) pair pm (path p ending to cluster m) as:

$$V(pm) = \left(V(p),\ \hat{W}_m(p,b) \right) = \left(\sum_{l\in p} d_l,\ \underset{l\in p}{\&} \hat{C}_l,\ \hat{W}_m(p,b) \right), \tag{4}$$

where $\hat{W}_m(p,b)$=$\mathrm{MUV}_{\mathrm{EST}(p,b)}(\hat{W}_m)$ is the binary cluster availability vector of m with 0's at the first $\mathrm{EST}(p,b)$ elements.

We define a domination relationship between (path, cluster) pairs: p_1m_1 dominates another pair p_2m_2 for a given task, if p_1 dominates p_2 (Eq. 3), and also the cluster m_1 can execute the task (over p_1) at least at all time intervals at which cluster m_2 is available (over p_2). Formally:

$$\text{p1m1 dominates p2m2 (notation: p1m1} > \text{p2m2) iff } p_1 > p_2 \text{ and } \hat{W}_{m_1}(p_1,b) \geq \hat{W}_{m_2}(p_2,b)\ . \tag{5}$$

Clearly, we have $PM_{n\text{-}d} \subseteq P_{n\text{-}d}$. Therefore, to obtain the set $PM_{n\text{-}d}$ we apply Eq. (5) to the elements of $P_{n\text{-}d}$.

4.4 Finding the Optimal (path, cluster) Pair and the Transmission and Execution Time Offsets

In the third phase we apply an optimization function $f(V(pm))$ to the cost vector, V, of each pair $pm \in PM_{n\text{-}d}$ and choose the path that minimizes f. The function f yields a scalar cost per pm and has to be monotonic in each of the cost components. For example, it is natural to assume that it is increasing with respect to delay, decreasing with increased capacity availability, decreasing with increased cluster availability, etc.

In the context of this study we assume that we want to minimize the completion time of the task. To select the optimal (path, cluster) pair: (i) we compute the first available position to schedule the task for all pairs, (ii) we select the cluster with the minimum completion time and (iii) we select the time to sent the data as the earliest time possible. These calculations are well defined given the path and cluster utilization profiles of the *pm* pairs.

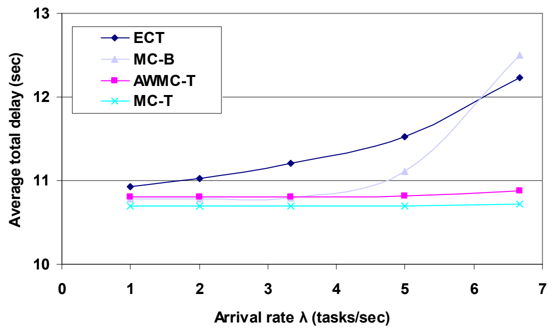
5. PERFORMANCE RESULTS

To evaluate the performance of the proposed multicost algorithm we conducted simulation experiments, assuming an OBS underlined network. We have extended the ns-2 platform [12] and tested the optimal multicost algorithm (MC-T) and the availability weight heuristic multicost algorithm (AWMC-T), as presented in Section 4, an algorithm that considers only the communication part of the problem (MC-B) [10], and a algorithm that considers only the computation problem (Earliest Completion time-ECT). In order to reserve the appropriate communication and computation resources we have implemented a one-way reservation protocol similar to JET [9], with extensions to cope with the reservation of computation resources.

The simulations were performed assuming a 5x5 mesh network with wraparounds, with neighboring nodes placed at a distance of 400 km. In this topology we randomly placed 4 clusters, each having 25 cpus. Each link had a single wavelength of capacity C_l = 1 Gb/s. Users were placed at all the 25 nodes of the network and the tasks were generated according to a Poisson process with rate $\lambda/25$ tasks per second per node. The computation workload of each task and the size of the input data were exponentially distributed. The generated tasks were cpu- and data-intensive since their load was considerable with respect to the total computation and communication capacity of the Grid network.

To assess the performance of the algorithms we measured the average total delay, defined as the time between the task creation and its completion time, the data burst blocking probability, defined as the probability of an input data burst to content with another burst, and the conflict probability defined as the probability of a task finding a cluster unavailable at the time predicted by the algorithm due to another task having already reserved that cluster.

In Figure 2a we observe that the multicost algorithms that jointly consider the communication and computation resources (MC-T, AWMC-T) perform better than the other two algorithms (MC-B, ECT) with respect to the average total delay metric. The average total delay of the MC-T and AWMC-T algorithms increases slightly with the tasks' generation rate λ. The tasks have high demands for both communication and computation resources and these algorithms solve this joint problem efficiently, and exhibit a low data blocking probability and a low conflict probability. On the other hand, the performance of ECT deteriorates as λ increases. ECT does not take into account the communication part of the problem, and thus exhibits a high data blocking probability as λ increases. Similarly, the performance of the MC-B algorithm deteriorates as λ increases. MC-B does not take into account cluster availability, and the clusters chosen are usually not the optimum ones, yielding a high conflict probability.



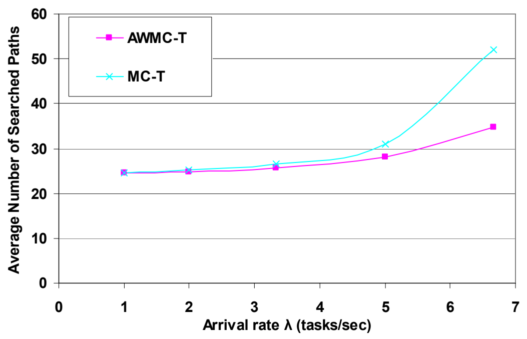


Figure 2. (a) Average total delay for tasks that are CPU- and data-intensive and (b) average number of searched paths for AWMC-T and MC-T.

In Figure 2b we graph the average number of searched paths per task request (that is, the average size of the set $P_{n\text{-}d}$, presented in Section 4.1). We observe that the proposed polynomial time AWMC-T algorithm yields delay performance that is very close to that of the optimal multicost algorithm, while maintaining the number of searched paths and associated required operations at low levels.

6. CONCLUSIONS

We presented a multicost algorithm for the joint selection of the communication and computation resources to be used by a task. We initially presented an optimal scheme of non-polynomial complexity and by appropriately pruning the set of candidate paths we also obtained a heuristic algorithm of polynomial complexity. We showed that in a Grid network where the tasks are CPU- and data-intensive important performance benefits, in terms of average total execution delay, can be obtained by jointly optimizing the use of the communication and computation resources as our proposed algorithms do. The proposed heuristic algorithm was shown to combine the strength of the optimal multicost algorithm with a low computation complexity.

REFERENCES

[1] I. Foster, C. Kesselman, "*The Grid 2: Blueprint for a New Computing Infrastructure*", Morgan Kaufmann, 2003.

[2] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy, "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation", *IWQoS*, 1999.

[3] R. Ali, K. Amin, G. Laszewski, O. Rana, D. Walker, M. Hategan, N. Zaluzec, "Analysis and provision of QoS for distributed grid applications", *Journal of Grid Computing*, vol 2(2), 2004.

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

[5] G. Hoo, W. Johnston, I. Foster, A. Roy, "QoS as Middleware: Bandwidth Reservation System Design", *HPDC*, 1999.

[6] K. Ranganathan, I. Foster. "Decoupling computation and data scheduling in distributed data-intensive applications", *HPDC*, 2002.

[7] R. Guérin, A. Orda, "Networks with Advance Reservations: The Routing Perspective", *Infocom*, 2000.

[8] Z. Wang, J. Crowcroft, "Quality-of-service routing for supporting multi-media applications", *J. Selected Areas in Communications*, vol. 14, no. 7, Sept. 1996.

[9] E. Varvarigos, V. Sharma, "An efficient reservation connection control protocol for gigabit networks", *Computer Networks and ISDN Systems*, vol. 30, no 12, pp. 1135–1156, 1998.

[10] E. Varvarigos, V. Sourlas, K. Christodoulopoulos, "Routing and Scheduling Connections in Networks that Support Advance Reservations", pending 2nd review at *Computer Networks*.

[11] F. Gutierrez, E. Varvarigos, S. Vassiliadis, "Multicost Routing in Max-Min Fair Networks", *Allerton Conference*, 2000.

[12] The Network Simulator (ns2): www.isi.udu/nsnam/ns .