Future Generation Computer Systems 25 (2009) 912–925

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Multi-cost job routing and scheduling in Grid networks

T. Stevens [a,*], M. De Leenheer [a], C. Develder [a], B. Dhoedt [a], K. Christodoulopoulos [b,c], P. Kokkinos [b,c], E. Varvarigos [b,c]

[a] Department of Information Technology, Ghent University – IBBT, Gaston Crommenlaan 8 bus 201, 9050 Gent, Belgium
[b] Department of Computer Engineering and Informatics, University of Patras, Greece
[c] Research Academic Computer Technology Institute, Patras, Greece

ARTICLE INFO

Article history:
Received 9 November 2007
Received in revised form
12 June 2008
Accepted 18 August 2008
Available online 9 September 2008

Keywords:
Algorithms
Network problems
Scheduling
Optimization

ABSTRACT

A key problem in Grid networks is how to efficiently manage the available infrastructure, in order to satisfy user requirements and maximize resource utilization. This is in large part influenced by the algorithms responsible for the routing of data and the scheduling of tasks. In this paper, we present several multi-cost algorithms for the joint scheduling of the communication and computation resources that will be used by a Grid task. We propose a multi-cost scheme of polynomial complexity that performs immediate reservations and selects the computation resource to execute the task and determines the path to route the input data. Furthermore, we introduce multi-cost algorithms that perform advance reservations and thus also find the starting times for the data transmission and the task execution. 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. Our performance results indicate that in a Grid network in which tasks are either CPU- or data-intensive (or both), it is beneficial for the scheduling algorithm to jointly consider the computational and communication problems. A comparison between immediate and advance reservation schemes shows the trade-offs with respect to task blocking probability, end-to-end delay and the complexity of the algorithms.

© 2008 Elsevier B.V. All rights reserved.

1. Introduction

Grid computing aims to offer a unified interface to various resources such as computational clusters, data storage sites and scientific instruments. In general, these resources are heterogeneous in nature, have different access and control policies, and are distributed on a large scale (possibly global) network. The driving force for the realization of such Grid technology is the highly challenging applications emerging from large-scale collaborations and eScience experiments. Recently, several proposals have been made to extend the concept for supporting enterprise- and consumer-oriented Grid applications [1]. Management and control of an efficient Grid system will therefore require intelligent *scheduling* at various levels [2]. Indeed, the complexity of the Grid applications, the user requirements and the system heterogeneity would result in suboptimal system performance in case manual procedures are used. Scheduling tasks on a set of heterogeneous, dynamically changing resources is a complex problem that requires sophisticated algorithms that take into account multiple optimization criteria. Such algorithms should try to balance the users' individual demands (e.g., cost, response-time), the objectives represented by the resource providers (profit, utilization) while at the same time also maintain a good overall performance for the Grid network.

Grid applications usually pose challenging demands to the networking fabric, as data transfers demand high-bandwidth and low latency connections. As such, optical networks have been identified as the most suitable technology to interconnect the distributed resources. Irrespective of the transport technology, efficient *routing* algorithms for data transfer between Grid sites is of great importance for the performance of any Grid deployment, and especially for Data-Grids.

Another issue for efficient management of Grid resources is how to take into account temporal information in the scheduling and routing decisions. Traffic demands are known to fluctuate over time, and both network and computational resources are subject to failures and disconnections. Thus, the option of immediately reserving the resources for a task is not always the best choice. In a Grid scenario, a user typically submits a task to have it processed within a predetermined deadline. In this context, network transfers and/or task executions can be delayed, allowing communication and computational loads to be more spread out

* Corresponding author.
E-mail addresses: tim.stevens@intec.ugent.be (T. Stevens), marc.deleenheer@intec.ugent.be (M. De Leenheer), kchristodou@ceid.upatras.gr (K. Christodoulopoulos), kokkinop@ceid.upatras.gr (P. Kokkinos), manos@ceid.upatras.gr (E. Varvarigos).

0167-739X/$ – see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2008.08.004

over time in order to efficiently serve/manage all the requests. Moreover, a number of Grid tasks require the utilization of certain resources for specific time periods. In both the above cases, the starting time of the resource reservation has to be (or is relaxed to be) in the future. This is generally referred to as *advance reservations*, as opposed to *immediate reservations* which are made in a just-in-time manner. However, inclusion of this temporal information in routing and scheduling algorithms will naturally increase the complexity of these algorithms. This paper will provide insight into this matter by comparing the delay, the acceptance performance, and the computational complexity of algorithms for both immediate and advance reservations.

In this work we address several of the previously introduced issues, by proposing two separate techniques in order to solve a communication and computation co-allocation problem of significant importance. The general assumption is that a task consists of two phases: (i) the transfer of data from the scheduler or a data repository resource (which we will call source) to a Computing Element (CE) or a cluster (which we will call destination) and (ii) the task's execution. Without loss of generality, we assume there is no output data associated with the tasks: this can be modelled as an additional task with only data transfer, to a specified destination. (Note that the algorithms can be extended to include output data transfer explicitly, but to limit the complexity we chose not to present those extensions in this paper.) The algorithms return the destination (CE) to execute the task and the path over which to route the input data. The first proposed technique, which is an extension of SAMCRA, is a multi-cost algorithm that uses the path delay and the computation load as selection metrics and is proven to be of polynomial complexity. This algorithm employs immediate reservations and can be applied in cases that the data size and the computation load of a task are either known or not known in advance. The second technique, called MC-T, is a multi-cost algorithm that is mainly used in the cases where the data sizes and the task execution times are known in advance. MC-T uses utilization vectors of the communication and computation resources in the multi-cost formulation in order to cope with advance reservations. This technique provides advance reservations by orchestrating the corresponding network and computational resources, and specifically returns not only the path and the destination (CE), but also the time that the data transmission should start and the time that the task execution should begin at the CE. Due to the large number of cost parameters that MC-T utilizes, the number of paths that it calculates can be exponential. By appropriately pruning the set of candidate paths we also present a heuristic algorithm of polynomial complexity. Table 1 provides an overview of the proposed algorithms.

We evaluate the performance of the various algorithms for multi-cost task routing and scheduling using full network simulation experiments. Our results indicate that in a Grid network in which tasks are either CPU- or data-intensive (or both), it is beneficial for the scheduling algorithm to jointly consider the computational and communication problems. A comparison between immediate and advance reservations shows the trade-offs with respect to task blocking probability and end-to-end delay. Finally, an analysis of the computational complexity of the proposed algorithms is also presented. Ultimately, we will show that the proposed algorithms combine the strength of multi-cost optimization (both for immediate and advance reservations) with a low computational complexity and running time.

The remainder of this paper is organized as follows. In Section 2 we report on previous work. Section 3 presents our model for Grid networks, with details on the assumptions for both computation and communication resources. In the same section we also propose an algorithm for immediate joint reservation of network and computation Grid resources. In Section 4, we initially present network and computation resource state models in the form of utilization profiles. Based on these profiles we present our advance reservation multi-cost algorithms (optimal and heuristic). In Section 5 we compare the proposed algorithms for a wide range of input parameters and scenarios, and analyze the performance results. Finally, our conclusions are summarized in Section 6.

2. Related work

The Grid Scheduling Architecture Research Group (GSA-RG) of the Open Grid Forum (OGF) in [3] provides different Grid scheduling use case scenarios and describes common usage patterns. Among them the most complicated scenario is scheduling tasks requesting more than one and possibly different service guarantees. In this content, a "workflow" is defined as a task that consists of a number of other "subtasks" with various interdependencies. Thus a workflow requests the co-allocation of resources in different time frames.

In general, Grid applications can be categorized as CPU- or data-intensive [4]. Different Grid environments have been created to cope with these two types of applications. Computational Grids normally deal with CPU-intensive problems on small data sets. In contrast, Data Grids mostly deal with problems that require the transfer of large amounts of data. However, in general all tasks have a computation and a communication part, even if one part is negligible. For example it is usual for a task to require the transfer of a large chunk of data, as a connection with a constant rate or a data burst, from the location of the user or a storage repository to the computation resource where it will be executed. This problem of communication and computation resources co-allocation will be addressed in this paper.

Resource co-allocation is one of the most challenging problems in Grids. The co-allocation problem for Computational Grids has been defined in [5]. In the Condor project, the gang matchmaking scheme [6] extends the matchmaking model in order to support the co-allocation of resources. In order to co-allocate resources as defined in workflows, the scheduler has to orchestrate resources belonging to different sites and different administrative domains. To do so, advance reservation of these resources has to be supported by the local resource management systems. The Globus Architecture for Reservation and Allocation (GARA) [7] is a framework for advance reservations that treats communication, computation, and storage resources in a uniform way. Although GARA has gained popularity in the Grid community, its limitations in coping with current application requirements and technologies led to the proposal of the Grid Quality of Service Management (G-QoSm) framework [8]. The WS-Agreement protocol [9] has been proposed by the GRAAP working group in the Open Grid Forum (OGF) for the negotiation of advance reservations.

Some specific instances of communication and computation co-allocation problems have previously been examined in the literature. In [10,11] the authors study the relation between job scheduling decisions and data replication strategies within a data Grid environment, in order to minimize job turnaround time. Similar algorithms have been examined in multimedia networks where the co-allocation of computation, communication (bandwidth) and other resources are examined [12]. A framework for specifying and handling co-reservations in Grid environment is presented in [13]. This framework can be applied to the reservation of applications running concurrently on multiple resources and to the planning of job flows, where the components may be linked by some temporal or spatial relationship. More general Grid workflow management systems have been developed by several projects: Condor DAGman [14], GridFlow [15], Gridbus [16], and

Table 1
Main characteristics of the proposed algorithms

Algorithm	Topic	Options
SAMCRA (Section 3)	Input	Known or not known input data size and computation complexity of tasks. Utilization of links and clusters at current time
	Output	Assignment of computation resources, routing of data over the communication resources
	QoS parameters	*Communication*: path availability, delay *Computation*: processing availability
	Objectives	Delay minimization, load balancing
	Reservation type	Immediate
	Complexity	Polynomial for the specific problem (in general NP-hard)
MC-T (Section 4)	Input	Known input data size and computation complexity of tasks Utilization of links and clusters as a function of time
	Output	Assignment and time scheduling of computation resources, routing and time scheduling of communication resources
	QoS parameters	*Communication*: path availability (also in future), delay *Computation*: processing availability (also in future), delay *Total*: availability (also in future), end-to-end delay
	Objectives	Delay minimization, load balancing
	Reservation type	In-advance (can perform immediate as well)
	Complexity	NP-hard, (heuristic: polynomial)

UNICORE [17]. A taxonomy of workflow management systems is presented in [18].

In the related literature, multi-cost algorithms have mainly been used for QoS routing problems [22–27]. In [22], the authors show that QoS routing with the specific parameters bandwidth and delay is not NP-complete, while the general QoS routing problem is discussed in [23]. To the best of our knowledge, the present work is the first time to use a multi-cost algorithm for the joint communication and computation problem in a Grid environment. Moreover, the algorithm presented in Section 4 is significantly different from other multi-cost approaches, since it is designed to handle temporal information, using timeslots as cost parameters in the multi-cost formulation, in order to cope with the time scheduling of the resources.

3. Exact multiple constraints routing: Immediate reservations

Upon submitting a job to a computing grid, *scheduling* and *routing* actions determine to which computing resource(s) the job is assigned and how it can be transmitted over the network. If there is no possibility to delay the job transmission time or the job execution time, *immediate reservation* of network and computational resources is mandatory. In this case, scheduling and routing can be tackled simultaneously by transforming the problem to a Multi-Constrained Path (MCP) problem with multi-dimensional weight vectors. When *optimal* use of the available network- and computational resources is desired, the Multi-Constrained Optimal Path (MCOP) should be computed for each generated job. For this purpose, we embrace the SAMCRA algorithm [21].

In the following sections we present a model for Grid networks, and show how cluster sites can be virtualized into a single anycast group, which then allows multi-constraint routing to be used to balance network and cluster availability.

3.1. Grid network and resources model

As illustrated in Fig. 1, we consider a Grid network consisting of links l of known propagation delays d_l, and a set M of clusters or CEs (these terms will be used interchangeably in this paper). Cluster $m \in M$ has W_m CPUs of a given computation capacity C_m per CPU (e.g. in MIPS). A task or job (also used interchangeably) is created by a user with specific needs: input data size I (bits), computational workload W (in MIs) and requires r CPUs for its execution. The parameters I and W may or may not be known in advance. We assume that the task consist of two sub-tasks: (i) the transfer of data from the scheduler or a data repository resource (which we will call source) to a computation resource (CE) in the form of a connection with a constant rate or a data burst, and (ii) the task execution at that CE. Observe that we do not consider the possible transfer of output data, as this can be considered an additional job without computational demand. Another implicit assumption is that each task can be processed at a single resource site, but multiple CPU's can be allocated to that task. Although this model does not allow complex workflows between processing sites, we adopt it here for simplicity and tractability purposes. Furthermore, there is an upper bound D on the maximum delay the task can tolerate; if this deadline cannot be met, the task is rejected. Each user communicates this information to its corresponding meta-scheduler S. We assume a centralized scheduling architecture, where all users forward their task scheduling requests to a single central scheduler S, which maintains information about the utilization of the communication and computation resources throughout the network.

We will examine two cases that differ in the type and the amount of information the scheduler maintains. In case (i) we assume that the data size and the computation workload of a task are not known in advance. In this case the scheduler cannot use advance reservations and thus maintains resources' utilization information that is not time dependent. In case (ii) we assume the data size and the computation workload of a task are known in advance and thus the scheduler maintains profiles that record the utilization of the resources as a function of time. Such communication and computation utilization profiles are presented in the next subsection. We are interested in scheduling algorithms that decide the destination (CE) at which to execute the task and also the path over which to route the input data. These scheduling algorithms take into account the utilization information available at the scheduler in order to efficiently utilize the available Grid resources.

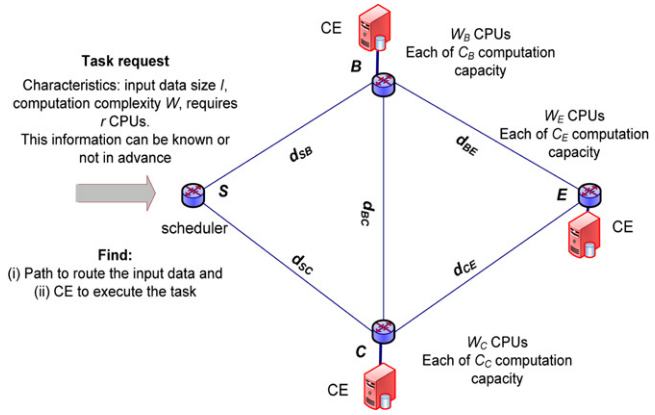


Fig. 1. A request for task execution is generated by a user and is forwarded to the scheduler S. The task has input data with size I, computation complexity W and requires r CPUs to execute.

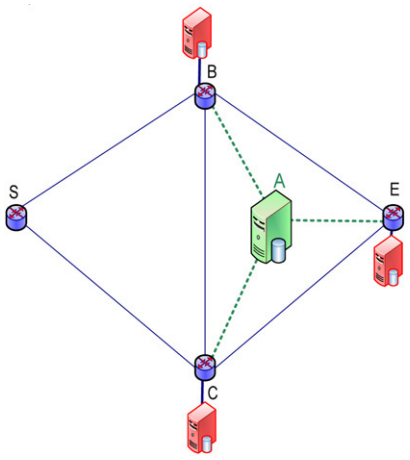


Fig. 2. Virtual topology.

3.2. Routing towards anycast destinations

From a routing perspective, cluster sites can be grouped in a single virtual anycast node A, as depicted in Fig. 2. Such an abstraction is only applicable if the virtual anycast group is the *final routing destination*, because this virtual anycast node cannot forward packets over virtual links. For the application in mind, anycast nodes are computational resources ($m \in M$) and not routers, making this a realistic assumption. Furthermore, this approach requires a distinct logical network topology – and hence a distinct routing component instance – for each anycast group present in the routing domain. In general, however, the number of anycast groups in an autonomous system will probably be small because of their focused applicability.

When traditional single constraint shortest path routing is applied to the logical topology, *shortest shortest path (SSP) routing* as discussed in [21] is achieved. Using SSP, the nearest anycast target node is contacted and data is sent over the shortest path. In this paper, additional metrics are also taken into account (e.g., resource availability), so Dijkstra shortest path routing or other single constraint routing algorithms are not sufficient. In a general multi-constrained anycast routing problem, each network link l is characterized by k link weights $w_i(l) \geq 0$ for all $1 \leq i \leq k$. Besides additive network-related constraints such as delay or hop count, also server-related constraints can be taken into account (e.g., server load). In this case, edges not directly attached to a member of the anycast server group have a weight equal to zero for all server-related constraints. Therefore, only the last edge of a path p from a source node to an anycast member can have a non-zero value for the server-related components of the weight vector. Based on [3], the corresponding anycast MCOP problem can be defined as follows: Given k constraints L_i ($1 \leq i \leq k$), find a path P from a source node (which in Fig. 2 is S) to the virtual anycast node A which satisfies the following condition:

$$w_i(p) \overset{\text{def}}{=} \sum_{l \in P} w_i(l) \leq L_i.$$

Additionally, for some length function $d(.)$, the condition $d(p) \leq d(p')$ should hold for all paths p' between S and A.

This anycast multiple constraints routing problem can now be solved by applying the SAMCRA algorithm, which is discussed in the next section.

3.3. Self-adaptive multiple constraints routing algorithm: SAMCRA

In this section, the SAMCRA algorithm with look-ahead extension [23] is briefly summarized; in the next section we present an adaptation of the original SAMCRA sub-path evaluation ordering. For an in-depth investigation of the algorithm, the reader is referred to [23].

SAMCRA is based on four key concepts: non-linear path length, k-shortest paths, non-dominated paths and look-ahead. Before presenting the complete algorithm, these key concepts are clarified.

The q-vector norm of path P from source S to destination E can be computed as follows:

$$d_q(p) = \left(\sum_{i=1}^{k} \left(\frac{\sum_{l \in S \to E} w_i(l)}{L_i} \right)^q \right)^{\frac{1}{q}}.$$

For $q \to \infty$, this length function can be rewritten as follows:

$$d_\infty(p) = \max_{1 \leq i \leq k} \left(\frac{\sum_{l \in S \to E} w_i(l)}{L_i} \right).$$

According to Van Mieghem et al. [23], finding the shortest path between S and E using the *non-linear length function* d_∞ in the equation above, solves the MCOP routing problem.

The *k-shortest paths* algorithm is similar to Dijkstra's algorithm, but stores the k shortest paths instead of the single previous *hop* in each node. This is necessary because in a multi-constrained environment, sub-paths of shortest paths are not necessarily shortest paths themselves. If a fixed value for the number of paths k is specified, the multi-constrained shortest path from a source to a destination may not be found. For SAMCRA, the number of paths stored in each node is unrestricted, meaning that *all* possible paths may need to be stored before the shortest one can be selected. This property leads to the worst-case NP-complete behavior of SAMCRA [24].

The *non-dominance* concept allows for a drastic reduction of the number of sub-paths that need to be stored in the router nodes. This optimization dismisses newly computed sub-paths from the source to the current routing node that a priori lead to a non-optimal path from the source to the final destination, based on previously computed sub-paths stored in the node. More concrete, new sub-paths between the source and the current routing node are dismissed if a previous sub-path to the same routing node has a weight vector for which each vector component is smaller than the corresponding component of the new sub-path weight vector.

The *look-ahead* extension further reduces the search space of possible paths by predicting the total length from source to destination for each sub-path stored in intermediate routers. This path length prediction is based on the sub-path history and Dijkstra

```
1.  SAMCRA(S,D)
2.  MAX_LENGTH ← 1, Q ← {∅}
3.  for all nodes v do
4.      K[v] ← 0
5.      Store Dijkstra look-ahead info for all constraints in b[v]
6.  for all constraints do
7.      if d_∞(Dijkstra path S → D for current constraint) < MAX_LENGTH then
8.      MAX_LENGTH ← d_∞(Dijkstra path S → D for current constraint)
9.  priority(S[1]) ← 0, path(S[1]) ← NULL
10. insert (Q,S[1])
11. while Q ≠ {∅} do
12.     u[i] ← extract_min(Q)
13.     u[i] marked GREY
14.     if u = D then
15.         stop and return path(u[i])
16.     else
17.         for all v in adjacency_list(u) do
18.             if v not in path(u[i]) then
19.                 PATH ← path(u[i]) + (u → v)
20.                 DOMINATED ← dominated(PATH)
21.                 mark all obsoleted paths v[j] BLACK
22.                 PRED_LENGTH ← d_∞( d[PATH] + d[b[v]] )
23.                 if PRED_LENGTH ≤ MAX_LENGTH and DOMINATED = false then
24.                     if all v[j] ≠ BLACK then
25.                         K[v] ← K[v] + 1
26.                         path(v[K[v]]) ← PATH
27.                         priority(v[K[v]]) ← PRED_LENGTH
28.                         insert(Q,v[K[v]])
29.                     else
30.                         path(v[BLACK_k]) ← PATH
31.                         decrease_key(Q, v[BLACK_k], PRED_LENGTH)
32.                     if v = D and PRED_LENGTH < MAX_LENGTH then
33.                         MAX_LENGTH ← PRED_LENGTH
```

Fig. 3. Meta-code for the SAMCRA algorithm.

shortest path information for all constraints from the intermediate router to the final destination. By applying the look-ahead concept, sub-paths with the lowest end-to-end predicted path length are evaluated *first*.

Meta-code for the complete algorithm is presented in Fig. 3. Lines 1–10 take care of the initialization. For each node v, Dijkstra look-ahead information $b[v]$ is computed and $K[v]$, the number of stored sub-paths between S and v, is initialized to 0. If the length of a Dijkstra path for one of the constraints is smaller than the maximum length MAX_LENGTH, the value of MAX_LENGTH is updated. Furthermore, the source node S is inserted in the priority queue Q with an empty path history. The iterative search for the shortest path starts at line 11. First, the sub-path with the lowest predicted end-to-end path length is dequeued and marked "GREY", which means this path is not considered anymore during the next iterations. If the destination has been reached, the algorithm stops. Starting at line 17, the current sub-path is extended by examining all nodes that are adjacent to the current intermediate router and are not listed in the sub-path history. For each of these path extensions, path dominance is evaluated and previously computed sub-paths that have become obsolete, are marked "BLACK". If the extended path is non-dominated and the predicted length is less than or equal to the maximum length, the priority queue is updated. When an arbitrary sub-path $v[BLACK_k]$ has become obsolete, it is replaced by the new sub-path in the priority queue (by decreasing the key value and updating the path). Otherwise, a new item is created and inserted into the priority queue. On lines 32–33, the maximum length MAX_LENGTH is updated if a shorter path reaching D is found.

In Fig. 4, a two-dimensional example illustrates the SAMCRA routing algorithm steps to compute a path from C (bottom) to E (right). Two-dimensional link weights and Dijkstra look-ahead vectors are depicted next to links and nodes, respectively. Step 1 depicts the algorithm state after the initialization phase (lines 1–10): look-ahead information is computed and MAX_LENGTH is reduced according to lines 6–8 in the SAMCRA meta-code. Subsequently, the priority queue is initialized by inserting a single element with empty path history. Steps 2–3 represent successive iterations of the main algorithm loop (lines 11–33). Each step starts by dequeueing the sub-path with the smallest predicted end-to-end cost (drawn in bold), whereupon possible path extensions (indicated by dotted arrows) are investigated. Path extensions that do not create a routing cycle and are not dominated by other sub-paths to the same intermediate router are inserted into the priority queue if their predicted end-to-end (i.e., $C \rightarrow E$) length (the priority queue key value) does not exceed MAX_LENGTH. During step 4, an end-to-end path is dequeued from the priority queue and the SAMCRA algorithm returns the solution $C \rightarrow B \rightarrow E$.

3.4. Avoiding sub-optimal SAMCRA results

Unfortunately, the path computed in the example above ($C \rightarrow B \rightarrow E$) is not the optimal solution: when taking into account the path length in the non-dominating dimension(s), the SAMCRA algorithm should have returned the path $C \rightarrow E$. Actually, it is not the SAMCRA algorithm itself but the $d_\infty(.)$ metric which leads to sub-optimal results. Because the path evaluation order (priority queue) is determined only by the dominating component of the predicted path length vector, *optimality in the non-dominating vector components cannot be guaranteed* in the evaluation order between equal length (sub-)paths (according to the $d_\infty(.)$ metric).

We adapted the SAMCRA algorithm to cope with this issue. Instead of using the $d_\infty(.)$ metric to determine the order in which sub-paths are evaluated, the information contained in the *entire path weight vector* p^w is used when ordering the sub-paths. Let $\rho(p^w)$ be a rescaled vector with components $\rho(p^w)_i = p_i^w/L_i$. Let $o(\rho(p^w))$ be the vector $\rho(p^w)$ with its indices reordered so that the components are in non-increasing order. The vector ordering $p^w < q^w$ holds if the first nonzero component of $o(\rho(p^w)) - o(\rho(q^w)) < 0$. In contrast, the original SAMCRA algorithm only considers the outcome of the first component of $o(\rho(p^w)) - o(\rho(q^w))$. It is clear that the proposed vector ordering is a refinement of the $d_\infty(.)$ metric that allows to differentiate between (sub-)paths with equal length according to the $d_\infty(.)$ metric. For unequal path lengths according to the $d_\infty(.)$ metric, the same ordering is preserved. With the new vector ordering, path length can only be equal if $o(\rho(p^w)) = o(\rho(q^w))$.

When revisiting the example depicted in Fig. 4 and applying the modification presented above, step 4 will dequeue the optimal path $C \rightarrow E$ because the non-dominating vector component is smaller. Indeed, we have

$$o(\rho(C \rightarrow E)) = (0.4, 0.1) \quad \text{and} \quad o(\rho(C \rightarrow B \rightarrow E)) = (0.4, 0.2)$$

and consequently

$$C \rightarrow E < C \rightarrow B \rightarrow E.$$

This adaptation is particularly important when the SAMCRA algorithm is applied in a hop-by-hop scenario [25]. If zero link weight components are allowed, hop-by-hop SAMCRA requires an exact solution in each intermediate node in order to prevent *routing loops*.

3.5. Complexity analysis

In the general case, the SAMCRA algorithm cannot guarantee to find the optimal path in polynomial time. According to Kuipers [26], the worst-case complexity of SAMCRA is:

$$C_{SAMCRA} = O(p_{\max} N \log(p_{\max} N) + p_{\max}^2 kE),$$

where N and E depict the number of nodes and edges in the network, respectively. $p_{\max}$ corresponds to the maximum number of sub-paths in an intermediate node that need to be evaluated before the optimal solution is found and is bounded by

$$p_{\max} = O(N!) = O(\exp(N \ln N)),$$

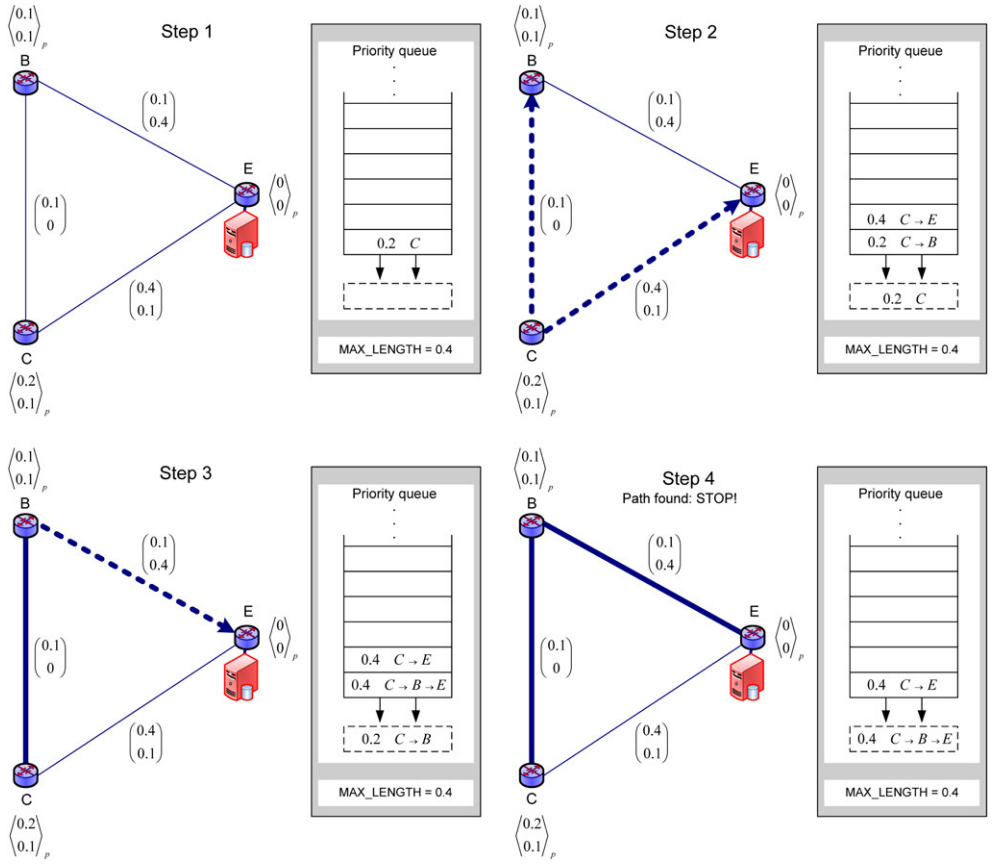


Fig. 4. Illustration of the SAMCRA algorithm.

thereby potentially leading to intractability. In practice, link weights will have a finite granularity and can be represented by integer values. In this case

$$p_{\max} = \frac{\prod_{i=1}^{k} L_i}{\max_{1 \le i \le k} L_i},$$

where L_i are the constraints (higher possible values), and SAMCRA has a pseudo-polynomial-time complexity.

For the two-constraint scenario considered, a link has two weights, being the delay, which is a float cost, and the availability of CPUs at the Computing Element of the ending node, which is a bounded integer cost (and thus, by definition, has finite granularity). Link delays will be zero for the virtual links leading to the computation resources, and resource's load will be zero for ordinary network links. By definition, each regular node will store at most one sub-path and the virtual target node will store at most M paths, where M stands for the number of Computing Elements. To this end the maximum number of sub-paths at an intermediate node is $p_{\max} = O(M)$. The resulting worst-case complexity is given by:

$$C_{SAMCRA-2costs} = O(MN \log(MN) + M^2 kE),$$

which is polynomial.

Note that this only applies to the case of these two specific cost weights and not the general case of SAMCRA algorithm. For an in-depth complexity analysis of the SAMCRA algorithm, we refer to [26].

4. Multi-cost task routing and scheduling: Employing advance reservations

The second algorithm that is presented in this section requires that the task's input data size and the computation load are known in advance (or an accurate estimation is available) and employs advance reservations of communication and computation resources.

4.1. Time dependent utilization profiles

Link utilization profile

The immediate reservation algorithm proposed in Section 3 poses no constraints on the underlying network, while the algorithm proposed in this section requires a network that supports advance reservations, as discussed in [20,28].

We require that each node records the capacity reserved on its outgoing links as a function of time, in order to perform channel scheduling and reservations. Assuming each connection or data burst reserves bandwidth equal to the link capacity for a given time duration, the *utilization profile* $U_l(t)$ of a link l is a stepwise binary function with discontinuities at the points where reservations begin or end, and is updated dynamically with the admission of each new connection. We define the *capacity availability profile* of link l of capacity C_l as $C_l(t) = C_l - U_l(t)$. In order to obtain a data structure that is easier to handle in an algorithm, we discretize $C_l(t)$ in time steps of duration τ_l to obtain the *binary capacity availability vector* $\hat{C}_l$, abbreviated CAV, as the vector whose k-th entry is:

$$\left\{\hat{C}_l\right\}_k = \begin{Bmatrix} 1, & \text{if } C_l(t) = 1 \\ 0, & \text{otherwise} \end{Bmatrix},$$

$$\text{for all } (k-1) \cdot \tau_l \le t \le k \cdot \tau_l,\ k = 1, \ldots, u_l$$

where u_l is the dimension of the CAV (see Fig. 5).

Cluster utilization profile

To have a consistent formulation, we define the utilization profile $U_m(t)$ of cluster m as an integer function of time, which records the

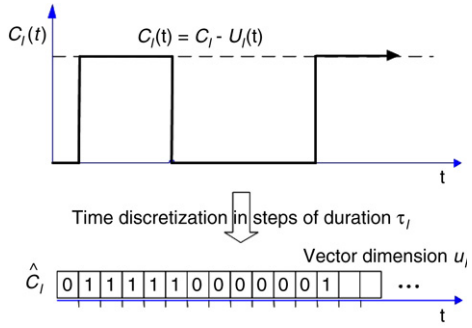


Fig. 5. The capacity availability profile $C_l(t)$, and the binary capacity availability vector $\hat{C}_l$ of a link l of capacity C_l, when the discretization step is τ_l.

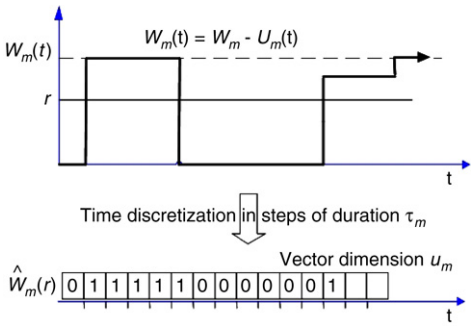


Fig. 6. The cluster availability profile $W_m(t)$, and the binary r-cluster availability vector $\hat{W}_m(r)$ of a cluster m with W_m processors, when the task requests to be executed on r processors and the discretization step is τ_m.

number of processing elements that have been committed to tasks at time t relative to the present time. The maximum value of $U_m(t)$ is the number of CPUs W_m, and it has a stepwise character with discontinuities of height r (always integer number) at the starting and ending times of tasks. In case all tasks request a single CPU, the steps are always unitary. We define the *cluster availability profile*, which gives the number of CPUs that are free as a function of time, as $W_m(t) = W_m - U_m(t)$. In order to obtain a data structure that is easier to communicate and store, we discretize the time axis in steps of duration τ_m and define the *binary r-cluster availability vector* $\hat{W}_m(r)$, as follows:

$$\left\{\hat{W}_m(r)\right\}_k = \begin{Bmatrix} 1, & \text{if } W_m - U_m(t) > r \\ 0, & \text{otherwise} \end{Bmatrix},$$

$$\text{for all } (k-1)\cdot\tau_m \le t \le k\cdot\tau_m,\ k = 1, 2, \ldots, u_m$$

where u_m is the maximum size of the cluster availability vector (see Fig. 6).

To simplify the presentation and the experiments, we assume for this study that each task requests $r = 1$ processors, which is the most usual case. Then, we can denote $\hat{W}_m(r)$ by $\hat{W}_m$ suppressing the dependence on r.

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 and the dimensions u_l and u_m used in the link and cluster utilization profiles, respectively, can be different to account for the different time scales in the reservations performed on the communication and computation resources, and to separately control the efficiency–accuracy we want to obtain in each case.

The timeslot-based management of allocated resources [27–29] has been used in different environments for advance reservations, e.g. denoted as "slot table" in GARA [7]. A recent paper [30] has focused on enhancing the timeslot-based approach by introducing dynamic timeslots and the notion of granularity, and

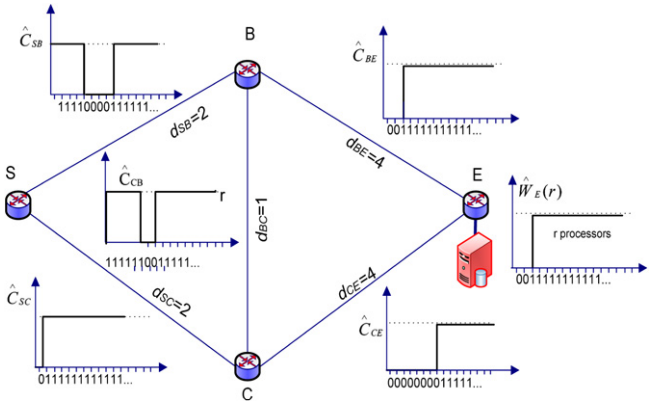


Fig. 7. The co-allocation problem when the task's input data size I and computation workload W are known in advance. Each link is characterized by its propagation delay and its binary capacity availability vector. Node E is a cluster with binary r-cluster availability vector $\hat{W}_E(r)$.

developed analytical models of the interrelation between user and system parameters. A different approach to maintain utilization information as a function of time is instead of bit-vectors to have linked lists that store the changes of states (from 0 to 1 and from 1 to 0) [31]. Manipulating such linked lists is also straightforward.

4.2. Grid network and resources model in the case of advance reservations

The assumptions of the Grid Network and the traffic are the same as Section 3.1, but we also assume that we know in advance or have an accurate estimation of the input data size I and the computational workload W of a task. Moreover, we assume that the scheduler S has 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. Given the previous information, we want to find a suitable cluster to execute the task, a feasible path over which to route the input data from the source (which can be the scheduler or a data repository site), 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. In other words we want to find a (path, cluster) pair and the corresponding Time Offsets, to transmit the data of the task (TO_{path}), and execute the task at the cluster ($TO_{cluster}$). Fig. 7 presents an instance of the problem.

4.3. Binary capacity availability vector of a path and binary cluster availability vector over a path

Calculating the binary capacity availability vector of a path

Assuming the routing and scheduling decision is made at the scheduler S, the capacity availability vectors of all links should be gathered continuously. To calculate the Capacity Availability Vector (CAV) of a path we have to combine the CAVs of the links that comprise it, as described in [20].

For example, for the topology of Fig. 7, the CAV of path SBE (p_{SBE}), consisting of links l_{SB} and l_{BE}, is

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

where $\hat{C}_{SB}$ and $\hat{C}_{BE}$ are the CAVs of links l_{SB} and l_{BE}, respectively, and $\mathrm{LSH}_{2\cdot d_{SB}}$ defines the left shift of $\hat{C}_{BE}$ by $2\cdot d_{SB}$ (twice the propagation delay of link l_{SB} measured in τ_l-time units). Left shifting $\hat{C}_{BE}$ by

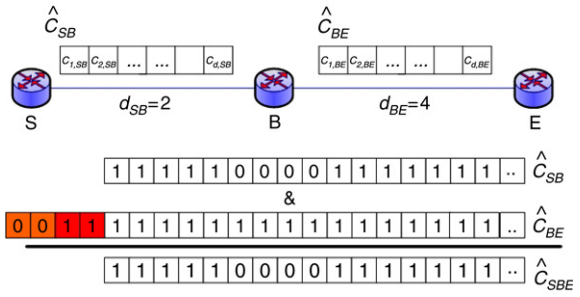


Fig. 8. Calculation of the path capacity availability vector $\hat{C}_{SBE}$. $\hat{C}_{BE}$ is shifted by 2 $d_{SB}\tau_l$-time units ($d_{SB} = 2$ in this example), before the AND operation is applied.

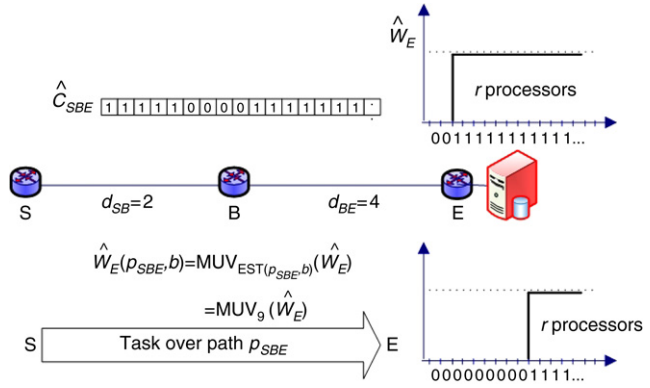


Fig. 9. Calculation of the cluster availability vector of E at the scheduler S that wants to transfer a task of transmission duration $b = 3$ over path p_{SBE}. We denote by $EST(p_{SBE}, b)$ the earliest time the task can reach E over p_{SBE} and by $\hat{W}_E(p_{SBE}, b)$ the cluster availability vector that gives the time periods at which S can schedule the task at E. To calculate $\hat{W}_E(p_{SBE}, b)$, we put 0's in the first $EST(p_{SBE}, b) = 9$ elements of $\hat{W}_E$.

d_{SB} positions purges utilization information corresponding to time periods that have already expired (time to transfer $\hat{C}_{BE}$ information from B to S), while left shifting it by another d_{SB} accounts for the propagation delay any data sent from S suffers to reach node B (assuming the link propagation delay is the same in both directions). We finally execute a bit-wise AND operation, denoted by '**&**', between the CAVs of SB and BE to compute the binary availability vector of the whole path SBE. This process is depicted in Fig. 8.

Calculating the binary cluster availability vector over a path

Let p be the path that starts at the scheduler S and ends at a cluster m, C_p its CAV and $d_p = \sum_{l=1}^{k} d_l$ its delay. We want to transmit a task with data duration b ($b = I/C_l$ where I is the size of the input data) over the path p_1 in order to be executed on cluster m. We define $R_p(b)$ the as the first position after which C_p has b consecutive ones. In other words, $R_p(b)$ is the earliest time after which a connection or a data burst of duration b can start its transmission on path p. The earliest time that the task can reach cluster m is given by $\text{EST}(p, b) = R_p(b) + b + d_p$. The scheduler S maintains information about the cluster availability vector $\hat{W}_m$ of m. We define $\text{MUV}_k(\hat{W}_m)$ as the operation of setting zeros (making unavailable) the first k elements of vector $\hat{W}_m$. Then vector $\hat{W}_m(p, b) = \text{MUV}_{\text{EST}(p,b)}(\hat{W}_m)$ gives the time periods that S can schedule the task over path p at cluster m.

With respect to the Figs. 7 and 8 we assume that we want to transmit a task with transmission duration $b = 3$ from S to the cluster located at node E, over path p_{SBE} with propagation delay $d_{SBE} = 6$. The capacity availability vector of the path $\hat{C}_{SBE}$ was calculated in the previous section, and thus we can compute $R_{p_{SBE}}(b) \Rightarrow R_{p_{SBE}}(3) = 0$. The task can reach E after time

$$\text{EST}(p_{SBE}, b) = R_{p_{SBE}}(b) + b + d_{SBE} = 0 + 3 + 6 = 9.$$

Also, S has an accurate knowledge of the cluster availability profile $\hat{W}_E$. Based on the above, the cluster availability vector that gives the time periods that S can schedule the execution of task at E is $\hat{W}_E(p_{SBE}, b) = \text{MUV}_{\text{EST}(p_{SBE},b)}(\hat{W}_E) = \text{MUV}_9(\hat{W}_E)$, which is the operation of setting the 9 first entries of vector $\hat{W}_E$ to zero. This process is depicted in Fig. 9.

4.4. Optimal advance reservation multi-cost algorithm (MC-T)

In what follows, we present a multi-cost algorithm for the joint communication and computation scheduling of tasks. The algorithm we propose consists of three phases: given a source (which can be the scheduler S or a data repository site R) we first calculate the set P_{n-d} of non-dominated paths between the source and all the network nodes. In the second phase, we obtain the set PM_{n-d} of candidate non-dominated (path-cluster) pairs from the source to all the clusters that can process the task. Finally, in the third phase, we choose from the PM_{n-d} set the pair that minimizes the completion of the task execution, or some other performance criterion.

Phase 1: *Algorithm for computing the set of non-dominated paths*

In multi-cost 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,

$$V_l = (d_l, \hat{C}_l) = (d_l, c_{1,l}, c_{2,l}, \ldots, c_{u_l}, l),$$

but they may also include other parameters of interest (such as number of hops, the capacity availability profile instead of the binary CAV vector, the number of executed tasks in a cluster, etc.). A cost vector can then be defined for a path p consisting of links $1, 2, \ldots, k$, based on the cost vectors of its links, according to

$$V(p) = \bigotimes_{l=1}^{k} V_l \stackrel{\text{def}}{=} \left(\sum_{l=1}^{k} d_l, \bigoplus_{l=1}^{k} \hat{C}_l \right), \tag{2}$$

where $\oplus$ is the associative operator defined in Eq. (1).

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

p_1 dominates p_2 (notation : $p_1 > p_2$) iff

$$\sum_{l \in p_1} d_l < \sum_{l \in p_2} d_l \quad \text{and} \quad \bigoplus_{l \in p_1} \hat{C}_l \geq \bigoplus_{l \in p_2} \hat{C}_l, \tag{3}$$

where the vector inequality "$\geq$" should be interpreted component wise. The set of non-dominated paths P_{n-d} for a given connection and source-destination pair is then defined as the set of paths with the property that no path in P_{n-d} dominates another path in P_{n-d}.

An algorithm for obtaining the set P_{n-d} of non-dominated paths from a given source node to all destination nodes is given in [19] and [20], and it can be viewed as a generalization of Dijkstra's algorithm that only considers scalar link costs.

Phase 2: *Calculating the set of non-dominated (path, cluster) pairs*

In the first phase of our proposed routing and scheduling algorithm we obtained the set of non-dominated paths between the source and all the nodes of the network. We now expand the definition of the path cost vector to include the utilization profiles

of the clusters. More specifically, we define the cost vector of a (path, cluster) pair pm, of a path p ending to a cluster m, as:

$$V(pm) = \left(V(p), \hat{W}_m(p,b)\right) = \left(\sum_{l=1}^{k} d_l, \bigoplus_{l\in p} \hat{C}_l, \hat{W}_m(p,b)\right) \quad (4)$$

where $\hat{W}_m(p,b) = \mathrm{MUV}_{\mathrm{EST}(p,b)}(\hat{W}_m)$ is the binary cluster availability vector of m, located at the end of path p, with 0's at the first EST (p, b) elements (as described in Section 4.3).

We define a domination relationship between (path, cluster) pairs as follows. A (path, cluster) pair p_1m_1 dominates another pair p_2m_2 for a given task, if p_1 dominates p_2 according to Eq. (3), and also the cluster m_1 can schedule the task at least at all time intervals at which the cluster m_2 is available. More formally,

p_1m_1 dominates p_2m_2(notation : $p_1m_1 > p_2m_2$) iff

$$p_1 > p_2 \quad \text{and} \quad \widehat{W}_{m_1}(p_1,b) \geq \widehat{W}_{m_2}(p_2,b) \quad (5)$$

where the vector inequality "$\geq$" should be interpreted component wise. The set of non-dominated (path, cluster) pairs PM_{n-d} for a given task is then defined as the set of (path, cluster) pairs with the property that no pair in PM_{n-d} dominates another pair in PM_{n-d}.

From the previous definitions it is easy to conclude that the set P_{n-d} is a superset of PM_{n-d}($PM_{n-d} \subseteq P_{n-d}$}. Therefore, in order to obtain the PM_{n-d} set we apply Eq. (5) to the elements of the P_{n-d} set.

Phase 3: *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 of each pair $pm \in PM_{n-d}$ to select the optimal one. The function f can be different for different tasks, depending on QoS requirements. For example, if we consider the optimization of data transmission, which corresponds to the routing optimization problem, function f will select the path that minimizes the reception time of the data at the cluster. If we consider the optimization of the computation problem, function f will select the cluster that has the fewer scheduled tasks, or the one that minimizes its completion time. A combination of the above considerations can be also employed.

The optimization function f applied to a (path, cluster) cost vector to compute the final (scalar) cost 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 capacity, decreasing with increased capacity availability, decreasing with increased cluster availability, etc. Each path cost component is characterized by the way it is obtained from the links' cost vector components (e.g., addition for the delays, $\oplus$ for the capacity availability vectors, selection of the final node for the cluster availability vector) and also by the optimization criterion applied to it (e.g., minimization for the delay, maximization for capacity and cluster availability, etc.).

For the given task and the given instance of the Grid network (established and scheduled data connections, executed and scheduled tasks at clusters), we are sure that we can find the (path,cluster) pair that optimizes the given objective function f. The set of non-dominated (path,cluster) pairs PM_{n-d} is by definition bound to include all the pairs that have at least one availability slot more than any other, or delay less than any other pair, etc, thus include all the paths with a distinct cost that can affect the objective function. To this end, the optimum solution is among the pairs of this PM_{n-d} set.

In the third phase of the algorithm we choose from the set PM_{n-d} of non-dominated pm pairs the one that minimizes the optimization function $f(V(pm))$. For the context of this study we assume that we want to minimize the completion of the task and that we are using a one-way connection establishment and reservation scheme. This is done in the following way:

Step 1: *Compute the first available position to schedule the task*

We start from the cost vector $V(p_im_i)$ of pair p_im_i and calculate the first position $R_i(w_i)$ after which $\hat{W}_{mi}(p_i,b)$ has $w_i = W/C_{mi}$ consecutive ones. In other words, $R_i(w_i)$ is the earliest time at which a task of computation workload W can start execution on m_i. Note that the way w_i is calculated accounts for the computation capacity of resource m_i, and that $\hat{W}_{mi}(p_i,b)$, by definition, accounts for the earliest transmission time, the propagation delay of path p_i and the transmission delay (as described in Section 4.1).

Step 2: *Select the cluster with the minimum task completion time*

Select the pair p_im_i that results in the minimum completion time $R_i(w_i) + w_i$ for the task. In case of a tie, select the path with the smallest propagation delay The time offset of task execution ($TO_{cluster}$) is given by $R_i(w_i)$.

Step 3: *Selecting the time to transmit the input data*

Having chosen the pair p_im_i we transmit the data of the task at the earliest time possible. The time offset TO_{path} for the task transmission is $R_{p_i}(b)$, defined as the first position after which C_{p_i} has b consecutive ones.

Step 4: *Updating the CAV of chosen (path, cluster)*

Having chosen the pm pair and the time offsets (TO_{path} and $TO_{cluster}$) to transmit and execute the task, the next step is to update the utilization profiles of the corresponding links and the cluster.

The procedure described above assumes a tell-and-go reservation protocol. If we wish to use a tell-and-wait protocol we simply have to redefine $\hat{W}_m(p_i,b)$, $R_i(w_i)$ and $R_{p_i}(b)$ to take into account the round trip time before data is transmitted.

4.5. Polynomial algorithm for computing the set of non-dominated paths

A serious drawback of the optimal algorithm described in the previous section is that the number of non-dominated paths calculated in the first phase may be exponential, and the algorithm is not guaranteed to finish in polynomial time. More specifically, the optimal multi-cost algorithm uses a path cost vector that has $1 + u_l$ cost parameters, and in particular 1 float (delay) and u_l Boolean costs (path utilization vector), where u_l is the size of the link utilization vectors. The complexity of this algorithm is clearly exponential since for a given source-destination pair there can be $O(2^{u_l})$ non-dominated paths.

The basic idea to obtain polynomial time variations of this algorithm is to define a pseudo-domination relationship $>_{ps}$ between paths, which has weaker requirement than the domination relationship $>$ defined in Eq. (3), in the sense that two paths may not dominate each other with respect to the $>$ relationship, but one of them may dominate the other with respect to the $>_{ps}$ relationship.

In [20] two such pseudo-domination relations were proposed and evaluated. For the scope of this study we present the better performing relation. We define a new link metric, called the *slot availability weight* of the link, as $w_l = \mathrm{weight}(\hat{C}_l)$, where the weight() of a binary vector represents the total number of 1's in the vector.

The polynomial-time heuristic variation of the optimal multi-cost algorithm computes the set of non-pseudo-dominated paths following exactly the same steps as before. The algorithm still maintains the binary vectors of the paths but the domination relationship that is used to prune the paths is not Eq. (3) but the following:

p_1 pseudo-dominates p_2($p_1 >_{ps} p_2$) iff

$$\sum_{l\in p_1} d_l < \sum_{l\in p_2} d_l \text{ and weight}\left(\bigoplus_{l\in p_1} \hat{C}_l\right) > \text{weight}\left(\bigoplus_{l\in p_2} \hat{C}_l\right). \quad (6)$$

This pseudo-domination relation transforms the cost vector of a link into a cost vector with 2 costs. The first cost is the delay of the link which is an additive float cost, while the second cost is the availability weight of the link which is a concave bounded integer. The upper bound of the integer second cost is the size of the link vector u_l. A cost vector with these 2 costs (1 additive float + 1 bounded concave integer) results in a polynomial-time problem as proven in [22] and also used in [27]. More specifically, for a given value of the integer cost, there can be only one non-dominated path between a source-destination pair, the one with the smallest delay. Since the integer cost can take values at the most equal to the length of the vector, an upper limit on the number of non-pseudo-dominated paths per source-destination pair is the dimension u_l of the link availability vector, which is polynomial in u_l and clearly do not depend on the network size. Assuming the *general* case where the size of the problem is proportional to u_l (link utilization profiles are part of the problem and generally require $O(u_l)$ bits to record), this corresponds to a polynomial time algorithm.

The heuristic algorithm obtained by the pseudo-domination relationship of Eq. (6) avoids the tedious comparisons of the CAVs of the optimal multi-cost algorithm, by essentially converting a $u_l + 1$-dimensioned cost vector into a cost vector of dimension 2 that conveys most of the important information contained in the original vector. Although the set of non-pseudo-dominated paths that this relationship generates is not guaranteed to always contain the optimum path, our performance results indicate that by appropriate choosing the pseudo-domination relationship we can obtain performance that is very close to that of the optimal multicost algorithm.

5. Performance evaluation

The aim of this section is to explore the algorithmic performance of immediate versus advance reservations, and also compare the proposed algorithms to algorithms that handle only the computation or only the communication part of the problem. Note that the scheduling algorithms for communication and computation resources that are available in the literature do not consider the joint problem in the way the algorithms proposed in this paper do. Therefore, we were not able to compare the proposed algorithms directly to other approaches. The comparison is based on the algorithm's behavior for different job scenarios, more specifically CPU-intensive, data-intensive and combined CPU- and data-intensive tasks. The studied performance parameters are job blocking probability, average end-to-end delay, and algorithmic complexity in terms of the average number of computed paths and average number of operations per job.

5.1. Simulation setup

The network topology used is the Phosphorus network [32], consisting of 9 nodes and 16 bidirectional links as shown in Fig. 10 (link distances are also presented). We assume the network is composed of 1Gbps optical links, each offering a single wavelength. Three nodes are randomly selected to function as Computing Elements (CE), with each CE containing 60 CPUs, and each CPU offering 25 000 MIPS (Million Instructions Per Second—a typical value for current CPUs). Thus, the total computing power of the Grid network is given by $3 \cdot 60 \cdot 25\,000$ MIPS.

The main characteristics of the grid jobs are given by:

- Poisson arrival of jobs with average rate λ (jobs/s).
- Uniform source selection from the nine (9) network nodes for each job created, which implies that at every node jobs are created according to a Poisson process with average rate $\lambda/9$.

Table 2
Parameter values for different experiments

	W (MI)	I (MByte)	λ (jobs/s)
CPU-intensive	40 0000 (16 s)	10	1 ... 10
Data-intensive	25 000 (1 s)	100	1 ... 10
CPU- and data-intensive	400 000	100	1 ... 10

- Exponentially distributed job computation complexity, with average W (MI/job).
- Exponentially distributed data input sizes, in the form of a data burst, with average I (Mbytes/job).

The parameters λ, W and I have to be chosen carefully, in order to prevent a scenario in which the Grid infrastructure is constantly overloaded. For this purpose, it is necessary to impose restrictions on the generated traffic. More specifically, for the Computing Elements it should hold that:

$$\lambda \cdot W < 3 \cdot 60 \cdot 25\,000,$$

and for the network:

$$\lambda \cdot I < \text{sum\_of\_3\_smallest degrees} \cdot 1\ \text{Gb/s} = 6\ \text{Gb/s}.$$

Since there are 3 computation resource sites, drawn at random from the 9 available network nodes, the above network load limitation guarantees that the last links towards the resources are not overloaded. This helps us make the assumption that these last links do not become a network bottleneck.

The total end-to-end delay (in seconds) for a task can be computed as follows:

$$\begin{aligned}\text{Total delay} = {} & \text{path\_length} \cdot (5 \times 10^{-6}\ \text{s/km}) \\ & + (\text{input\_size}/(1\ \text{Gb/s})).\end{aligned}$$

The first component relates to the propagation delay of the link (5×10^{-6} s/km is a typical value for the propagation of a fiber with refractive index 1.5), whereas the second component accounts for the transmission delay of the input data burst.

In the following, we performed three sets of experiments as detailed in Table 2.

For all simulation results discussed in the following sections, 5 runs of 10 000 jobs were performed for each experiment, all with independent random seed. To account for the transient regime at the beginning, measurements only take into account a steady load and stop when the last job has been submitted.

In order to evaluate the performance of the proposed multi-cost algorithms for the joint communication and computation task scheduling, several simulation experiments were conducted and results are compared with several immediate and advance reservation scheduling algorithms. For advance reservation algorithms, a job execution deadline is defined as follows:

$$\begin{aligned}&\text{Job deadline} \\ &= 2 \cdot (\text{burst transmission delay} + \text{execution delay}) \\ &= 2 \cdot ((\text{burst\_size}/(1\ \text{Gb/s})) \\ &\quad + (\text{execution\_time (MI)}/25\,000\ \text{MIPS})).\end{aligned}$$

The following algorithms were implemented and evaluated:

- SAMCRA and SAMCRA HBH, the centralized and distributed (hop-by-hop) version, as discussed in Section 3.3. These are multi-constraint immediate scheduling algorithms. Note that in cases where we omit results for SAMCRA HBH, the performance is very similar to the centralized version of the algorithm.
- Optimal advance reservation multi-cost algorithm for the joint communication and computation task scheduling (MC-T), as presented in Section 4.4.

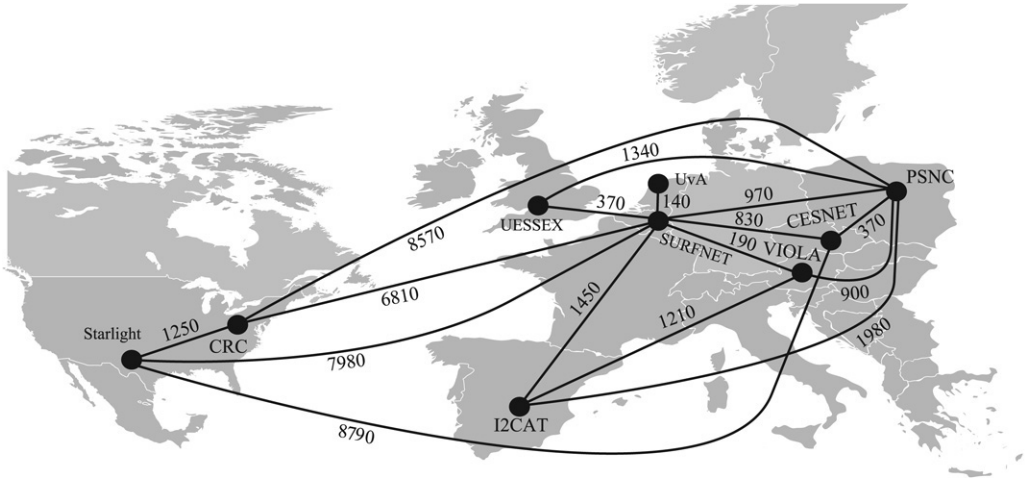


Fig. 10. Simulated topology—Phosphorus network.

- AW heuristic advance reservation multi-cost algorithm for the joint communication and computation task scheduling (AWMC-T), as presented in Section 4.5.
- Optimal advance reservation multi-cost burst routing and scheduling algorithm (MC-B). The MC-B algorithm takes into account only the communication part of the problem, and routes the input data to the cluster at which the data will arrive earlier, without using the corresponding time-dependent utilization profiles of the clusters.
- Earliest Completion time (ECT) advance reservation scheduling. The ECT algorithm considers only the computation part of the problem, and sends the task to the cluster where it will complete execution earlier, using the shortest path and taking into account contentions on the links that comprise that path.

For the MC-T, MC-B and ECT algorithms the used time-discretization steps and utilization vector dimensions (Section 4.1) for the links and clusters are: $\tau_l = 0.001$ s, $d_l = 10\,000$, and $\tau_m = 0.04$ s, $d_m = 10\,000$, respectively.

To assess the performance of the algorithms we used the following metrics:

- Job blocking probability: the probability of a job to be blocked, because of lack of network capacity or resource availability. For immediate scheduling, jobs can be blocked either because of network burst contention or resource contention (i.e., jobs arrive in a busy resource). For the algorithms that employ advance reservations (ECT, MC-B and MC-T), jobs can be blocked due to time-outs that occur when the job is waiting too long to be sent over the network or when it is queued upon arrival in a busy resource.
- Average end-to-end delay: defined as the time between the task creation and the time the task completes its execution. End-to-end job delay measurements include job queuing time (only applicable for advance reservations) and job transmission and execution duration. As such, delay measurements are only measured for successfully executed jobs.
- Average number of operations: the number of operations required to execute the routing and scheduling algorithm. An operation is defined as an addition, a boolean operation (e.g., XOR) or a comparison ($<$, $>$, $\neq$, etc.).

5.2. CPU-intensive tasks

Figs. 11 and 12 depict the job blocking rate and average end-to-end delay for the CPU-intensive experiments as described in Table 2. Note that the SAMCRA algorithm is an immediate scheduling approach, dropping tasks if it cannot immediately serve them, and as such the observed end-to-end delay measurements serve as a lower bound for all advance reservation algorithms (i.e., MC-T, AWMC-T, MC-B, and ECT). Also, note that MC-T is an optimum algorithm, in the context that for a given task and the specific network instance (link and cluster utilizations), it schedules the task (even in the future) in order to obtain the minimum end-to-end delay. A task that is scheduled by MC-T in the future would be dropped by SAMCRA. However, this task will have larger end-to-end delay than its transmission and execution time, due to the advance reservation. To this end, it is expected that SAMCRA will experience the minimum average end-to-end delay with a increased blocking rate, while MC-T will experience the best blocking performance with an increased delay (when compared to SAMCRA).

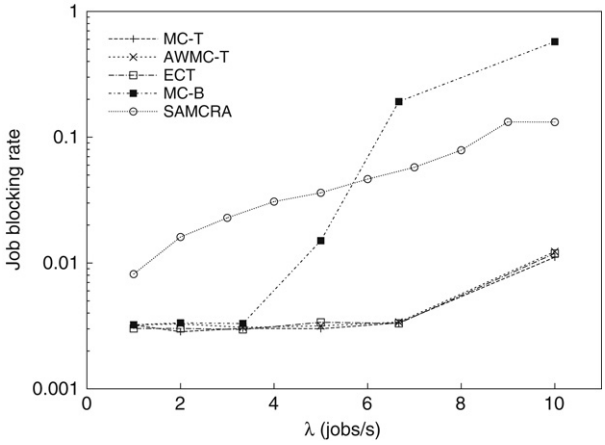


Fig. 11. Job blocking probability for CPU-intensive tasks.

Indeed, from Fig. 11 we can observe that ECT, MC-T and AWMC-T experience better blocking performance than the other algorithms, because of their effective job scheduling on the computational resources. MC-B optimizes network resource utilization, but this is not the bottleneck for the CPU-intensive workload of this scenario. As a consequence, when λ increases, MC-B schedules the jobs in the buffers of already busy clusters, their delay is increased and they eventually time-out. Note that even SAMCRA (without buffering capability in resources) outperforms MC-B in terms of job blocking rate when the load increases. ECT, MC-T and AWMC-T achieve efficient spreading of the jobs over the available resources, combining an acceptable end-to-end delay (which is dominated by the job execution time) with low job blocking rate (Fig. 12).

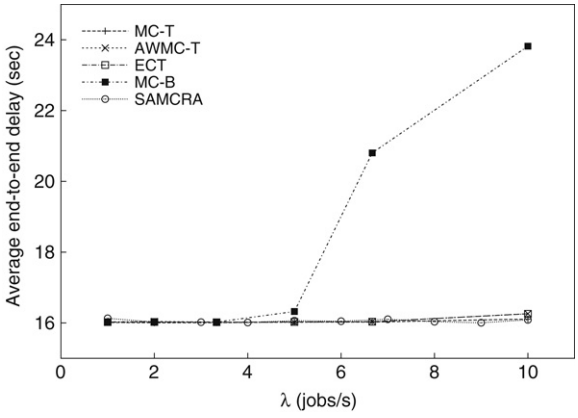


Fig. 12. Average end-to-end delay for CPU-intensive tasks.

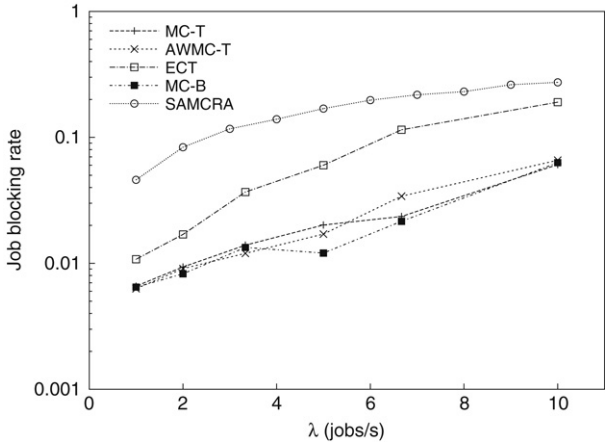


Fig. 13. Job blocking probability for data-intensive tasks.

5.3. Data-intensive tasks

In this section we focus on data-intensive tasks (see Table 2). Generating jobs at a higher rate results in an increasing network load, while computation resource utilization remains low. For this scenario, the job blocking rate and average end-to-end delay are depicted in Figs. 13 and 14, respectively. In terms of job blocking rate, SAMCRA and ECT cannot compete with MC-B, MC-T and AWMC-T. This is because both SAMCRA and ECT cannot schedule the input data in the future, which results in higher data blocking rates in the network core. In the case of SAMCRA, when the input data cannot be immediately routed, the job is rejected, whereas ECT waits for the shortest path to become available. This increases the end-to-end delay and the job may be rejected if its deadline is reached. The blocking rate for MC-T, AWMC-T and MC-B, are similar since all three algorithms consider the utilization of the communication resources, which is dominant in this Data-intensive scenario, in a similar way.

5.4. CPU- and data-intensive tasks

In this scenario, the generated jobs increasingly stress both the network and the computational resources as the job generation frequency λ increases. As expected, Figs. 15 and 16 show that the job blocking rate and average end-to-end delay are higher for this scenario than for the others examined in the previous paragraphs. SAMCRA still offers a lower bound for the end-to-end delay, at the expense of the highest job blocking rate. Both MC-B and ECT results expose a high job blocking rate combined with relatively high end-to-end delay values, because these approaches neglect one aspect

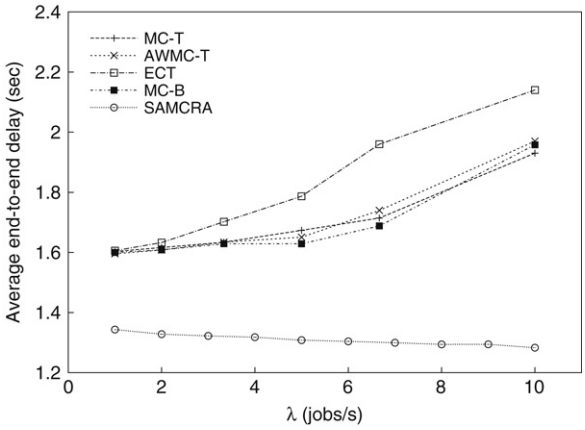


Fig. 14. Average end-to-end delay for data-intensive tasks.

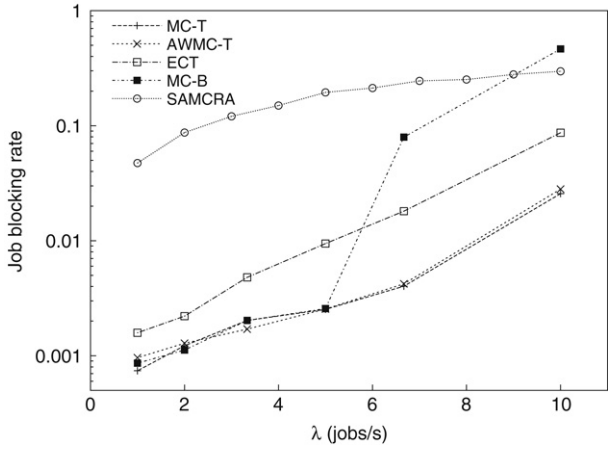


Fig. 15. Job blocking probability for CPU- and data-intensive tasks.

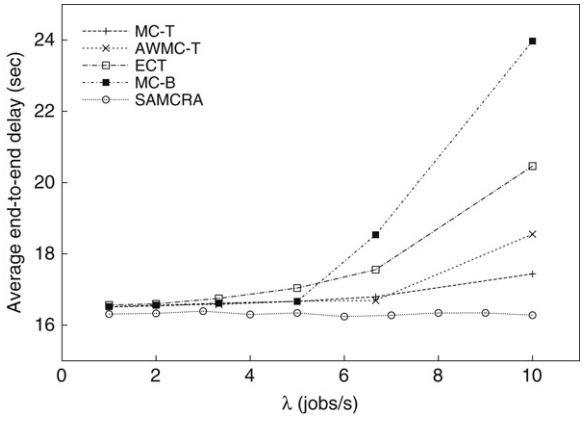


Fig. 16. Average end-to-end delay for CPU- and data-intensive tasks.

of the job characteristics (as already discussed in the CPU- and data-intensive scenario cases). To this end, when tasks are CPU- and data intensive, MC-T and AWMC-T algorithms excel for both performance criteria due to their combined optimization approach and their ability to perform advance reservations.

Note that the performance difference with respect to blocking and end-to-end delay between MC-T and AWMC-T is marginal in favor of MC-T in this and all the previously examined scenarios. AWMC-T is a heuristic version of the optimal MC-T and these performance results indicate that it can fairly approximate the optimal algorithm. However, it is expected that MC-T will have

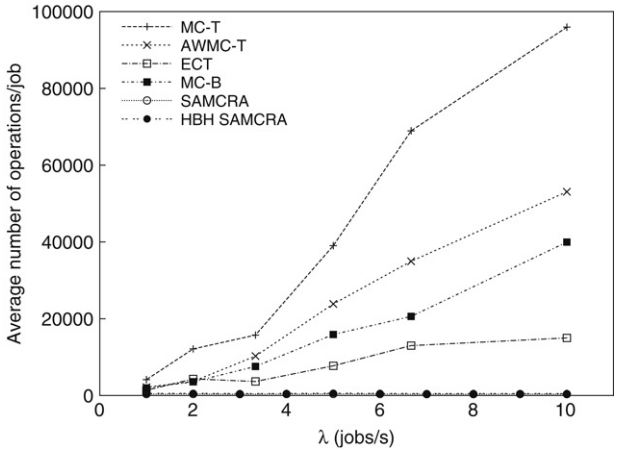


Fig. 17. Average number of operations required per task (CPU- and data-intensive tasks).

worse complexity performance, since it is an NP-hard algorithm, while AWMC-T is polynomial.

5.5. Complexity

Thus far, algorithm complexity has not been taken into account in the performance evaluation of the proposed algorithms. In theory we have proven that the exact multi-constraint routing algorithm for immediate reservations (SAMCRA) with the two specific costs as presented in Section 3 is polynomial. For advance reservations, the optimal multi-cost algorithm presented in Section 4.4 is NP-hard, while its heuristic variation presented in Section 4.5 is polynomial.

Fig. 17 shows algorithm complexity in terms of the average number of operations required to schedule a single job. When the job generation intensity λ increases, it becomes increasingly important to come up with a scheduling solution in a timely fashion. The figure shows that the advance reservation algorithms require a number of operations that are at least two orders of magnitude larger than those for the SAMCRA (HBH) algorithm. This is more severe as tasks are more data-intensive, in which case larger link utilization vectors need to be maintained and combined for all advance reservation algorithms. As explained in Section 4, MC-T optimal algorithm use as cost parameters the delay and the link (or path) utilization profiles and thus can be viewed as a multi-cost algorithm with $1 + u_l$ costs. Although reducing the value of u_l would result in decreased complexity, the algorithm's blocking performance would deteriorate since this would constrain its capability of advance scheduling. The value $u_l = 10\,000$ was chosen so as to optimize the blocking probability and in particular, in all the experiments that we conducted all task requests were able to find available (temporal) placements for their input data. On the other hand, AWMC-T heuristic algorithm was designed so as to avoid the tedious comparisons between the $1 + u_l$ costs, but it still has to perform additions between cost vectors and thus the high number of average operations. However, since it is proven that the number of non-dominated paths that it computes is polynomial on the size of the input, its complexity performance is expected to be acceptable irrespective of the network topology. MC-B algorithm handles link utilization information similar to MC-T and thus it also experiences high average number of operations. The ECT algorithm has to combine the utilization vectors of the links that comprise the shortest paths to the computation resources, in order to avoid contention over them. To this end ECT exhibits a higher number of performed operations than the SAMCRA algorithm. Finally, it is worth noting that for the SAMCRA algorithm the number of operations is independent of the generated load, potentially leading to greater scheduling scalability when the load is high.

6. Conclusions

In this paper, we presented several multi-cost algorithms for the joint scheduling of the communication and computation resources that will be used by a task. The proposed immediate reservation algorithm selects the computation resource to execute the task and determine the path to route the input data. Furthermore, we introduced multi-cost algorithms that perform advance reservations and thus also find the starting times for the data transmission and task execution. We demonstrated the inherent trade-off for immediate vs. advance reservations; the lower blocking probability achieved by advance reservations, comes at a small or moderate increase in the end-to-end delay. Finally, an analysis on the complexity of the various algorithms, in terms of number of paths computed and computational steps executed, was presented. The quantitative results showed the increased complexity of the advance reservation algorithms due to the inclusion of temporal information in the multi-cost formulation.

Acknowledgements

This work has been supported by the European Commission through the IP Phosphorus project. M. De Leenheer would like to thank the IWT for their financial support through his Ph.D. scholarship. C. Develder acknowledges the support of the FWO for his post-doctoral grant. K. Christodoulopoulos was supported by GSRT through PENED project 03EΔ207, funded 75% by the EC and 25% by the Greek State and the private sector.

References

[1] M. De Leenheer, P. Thysebaert, B. Volckaert, F. De Turck, B. Dhoedt, P. Demeester, D. Simeonidou, R. Nejabati, G. Zervas, D. Klonidis, M.J. O'Mahony, A view on enabling consumer-oriented grids through optical burst switching, IEEE Communications Magazine 44 (3) (2006) 124–131.
[2] I. Foster, C. Kesselman (Eds.), The Grid 2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 2003.
[3] R. Yahyapour, Ph. Wieder, Grid Scheduling Use Cases, Grid Scheduling Architecture Research Group, GSA-RG, Open Grid Forum, OGF, Mar. 2006.
[4] W.E. Johnston, Computational and data grids in large-scale science and engineering, Future Generation Computer Systems 18 (8) (2002) 1085–1100.
[5] K. Czajkowski, I.T. Foster, C. Kesselman, Resource co-allocation in computational grids, in: Proc. of the 8th IEEE International Symposium on High Performance Distributed Computing, HPDC-8, Aug. 1999, pp. 219–228.
[6] R. Raman, M. Livny, M. Solomon, Policy driven heterogeneous resource co-allocation with gangmatching, in: Proc. of the 12th IEEE International Symposium on High Performance Distributed Computing, HPDC-12, June 2000, pp. 80–89.
[7] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, A. Roy, A distributed resource management architecture that supports advance reservation and co-allocation, in: Proc. of International Workshop on Quality of Service, IWQOS, 1999, pp. 27–36.
[8] R. Ali, O. Rana, D. Walker, S. Jha, S. Sohail, G- QoSM: Grid service discovery using QoS properties, in: Grid Computing, Journal of Computing and Informatics 21 (4) (2002) 363–382 (special issue).
[9] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Nakata, J. Pruyne, J. Rofrano, S. Tuecke, M. Xu, Web Services Agreement Specification (WS-Agreement), Open Grid Forum, Mar. 2007.
[10] R.S. Chang, J.S. Chang, S.Y. Lin, Job scheduling and data replication on data grids, Future Generation Computer Systems 23 (7) (2007) 846–860.
[11] M. Tang, B.S. Lee, X. Tang, C.K. Yeo, The impact of data replication on job scheduling performance in the data grid, Future Generation Computer Systems 22 (3) (2006) 254–268.
[12] S. Venugopal, R. Buyya, L. Winton, A grid service broker for scheduling distributed data-oriented applications on global grids, in: Proc. 2nd Workshop on Middleware for Grid Computing, MGC, Feb. 2004, pp. 75–80.
[13] T. Roeblitz, A. Reinefeld, Co-reservation with the concept of virtual resources, in: Proc. 5th IEEE International Symposium on Cluster Computing and the Grid, CCGrid, May 2005, pp. 398-406.
[14] T. Tannenbaum, D. Wright, K. Miller, M. Livny, Condor—A distributed job scheduler, in: Beowulf Cluster Computing with Linux, The MIT Press, MA, USA, 2002.

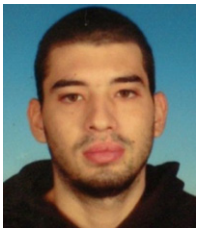
[15] J. Cao, S.A. Jarvis, S. Saini, G.R. Nudd, GridFlow: Workflow management for grid computing, in: Proc. 3rd International Symposium on Cluster Computing and the Grid, CCGrid, May 2003, pp. 198–205.
[16] R. Buyya, S. Venugopal, The gridbus toolkit for service oriented grid and utility computing: An overview and status report, in: Proc. 1st IEEE International Workshop on Grid Economics and Business Models, GECON, April 2004, pp. 19–36.
[17] A. Streit, D. Erwin, Th. Lippert, D. Mallmann, R. Menday, M. Rambadt, M. Riedel, M. Romberg, B. Schuller, Ph. Wieder, UNICORE—from project results to production grids, in: L. Grandinetti (Ed.), Grid Computing: The New Frontiers of High Performance Processing, in: Advances in Parallel Computing, vol. 14, Elsevier, 2005.
[18] J. Yu, R. Buyya, A taxonomy of scientific workflow systems for grid computing, in: Scientific Workflows, SIGMOD Record 34 (3) (2005) 44–49 (special issue).
[19] F. Gutierrez, E.A. Varvarigos, S. Vassiliadis, Multicost routing in max–min fair networks, in: Proc. 38th Allerton Conf. on Communicating, Control and Computing, 2000.
[20] E.A. Varvarigos, V. Sourlas, K. Christodoulopoulos, Routing and scheduling connections in networks that support advance reservatins, Computer Networks 58 (2008) 2988–3006.
[21] D. Xuan, W. Jia, W. Zhao, H. Zhu, A routing protocol for anycast messages, IEEE Transactions on Parallel and Distributed Systems 11 (6) (2000) 571–588.
[22] Z. Wang, J. Crowcroft, Quality of service routing for supporting multimedia applications, Journal on Selected Areas in Communications 14 (7) (1996) 1228–1234.
[23] P. Van Mieghem, F. Kuipers, Concepts of exact QoS routing algorithms, IEEE/ACM Transactions on Networking 12 (5) (2004) 851–864.
[24] F. Kuipers, P. Van Mieghem, Conditions that impact the complexity of QoS routing, IEEE/ACM Transaction on Networking 13 (4) (2005) 717–730.
[25] P. Van Mieghem, H. De Neve, F. Kuipers, Hop-by-hop quality of service routing, Computer Networks 37 (3–4) (2001) 407–423.
[26] F.A. Kuipers, Quality of service routing in the internet: Theory, complexity and algorithms, Ph.D. thesis, Delft University Press, The Netherlands, Sep. 2004.
[27] Shigang Chen, Klara Nahrstedt, On finding multi-constrained paths, in: Proc. IEEE International Conference on Communications, ICC, June 1998, pp. 874–879.
[28] R. Guerin, A. Orda, Networks with advance reservations: The routing perspective, in: Proc. 19th Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM, Mar. 2000, pp. 118–127.
[29] L.O. Burchard, Analysis of data structures for admission control of advance reservation requests, IEEE Transactions on Knowledge and Data Engineering 17 (3) (2005) 413–424.
[30] C. Barz, U. Bornhauser, P. Martini, M. Pilz, Timeslot-based resource management in grid environments, in: IASTED Conference on Parallel and Distributed Computing and Networks, PDCN, 2008.
[31] E.A. Varvarigos, V. Sharma, An efficient reservation connection control protocol for gigabit networks, Computer Networks and ISDN Systems 30 (12) (1998) 1135–1156.
[32] The Phosphorus project. http://www.ist-phosphorus.eu/.

T. Stevens received his M.Sc. degree in Computer Science from Ghent University, Belgium, in June 2001. Until July 2003, he was a database and system administrator for the Flemish public broadcasting company (VRT). At present, Tim Stevens is a Ph.D. student affiliated with the Department of Information Technology of Ghent University. He has been involved in multiple national and European research projects (IWT Move, IST MUSE, IST Phosphorus). His main research interests include future access network architectures, IPv6, Quality of Service (QoS) and traffic engineering in IP networks, and anycast-based services.

M. 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.

C. Develder received the M.Sc. degree in Computer Science Engineering and a Ph.D. in Electrical Engineering from Ghent University (Ghent, Belgium), in July 1999 and December 2003 respectively. From October 1999 on, he has been working in the Department of Information Technology (INTEC), at the same university, as a Researcher for the Research Foundation—Flanders (FWO), in the field of network design and planning, mainly focusing on optical packet switched networks. In January 2004, he left University to join OPNET Technologies, working on transport network design and planning. In September 2005, he re-joined INTEC at Ghent University as a post-doctoral researcher, and as a post-doctoral fellow of the FWO since October 2006. Since October 2007 he holds a part-time professor position at the same institute. He was and is involved in multiple national and European research projects (IST Lion, IST David, IST Stolas, IST Phosphorus, IST E-Photon One). His current research focuses on dimensioning, modeling and optimising optical Grid networks and their control and management. He is an author or co-author of over 45 international publications.

B. Dhoedt received a degree in Engineering from the Ghent University in 1990. In September 1990, he joined the Department of Information Technology of the Faculty of Applied Sciences, University of Ghent. His research, addressing the use of micro-optics to realize parallel free space optical interconnects, resulted in a Ph.D. degree in 1995. After a 2 year post-doc in opto-electronics, he became professor at the Faculty of Applied Sciences, 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 & wireless communications. Bart Dhoedt is author or co-author of approximately 150 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.



K. Christodoulopoulos received the Diploma of Electrical and Computer Engineering from the National Technical University of Athens, Greece, in 2002 and the M.Sc. degree in Advanced Computing from Imperial College London, UK, in 2004. He is currently working toward the Ph.D. degree at the Computer Engineering and Informatics Department of the University of Patras, Greece. His research interests are in the areas of protocols and algorithms for optical networks and grid computing.

P. Kokkinos received a Diploma in Computer Engineering and Informatics and a M.Sc. in Integrated Software and Hardware Systems from the Department of Computer Engineering and Informatics, University of Patras, Greece. Currently, he is pursuing a Ph.D. degree in the same University. His research activities are in the areas of ad-hoc networks and grid computing.

E. Varvarigos received a Diploma in Electrical and Computer Engineering from the National Technical University of Athens in 1988, and the M.S. and Ph.D. degrees in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology in 1990 and 1992, respectively. He has held faculty positions at the University of California, Santa Barbara (1992–1998, as an Assistant and later an Associate Professor) and Delft University of Technology, the Netherlands (1998–2000, as an Associate Professor). In 2000 he became a Professor at the department of Computer Engineering and Informatics at the University of Patras, Greece, where he heads the Communication Networks Lab. He is also the Director of the Network Technologies Sector (NTS) at the Research Academic Computer Technology Institute (RA-CTI).His research activities are in the areas of high-speed networks, protocols, network architectures, distributed computation and grid computing.