An Advance Reservation-based Co-Allocation Algorithm for Distributed Computers and Network Bandwidth on QoS-guaranteed Grids

Atsuko Takefusa[1], Hidemoto Nakada[1], Tomohiro Kudoh[1], and Yoshio Tanaka[1]

National Institute of Advanced Industrial Science and Technology (AIST)

Abstract. Co-allocation of performance-guaranteed computing and network resources provided by several administrative domains is one of the key issues for constructing a QoS-guaranteed Grid. We propose an advance reservation-based co-allocation algorithm for both computing and network resources on a QoS-guaranteed Grid, modeled as an integer programming (IP) problem. The goal of our algorithm is to create reservation plans satisfying user resource requirements as an on-line service. Also the algorithm takes co-allocation options for user and resource administrator issues into consideration. We evaluate the proposed algorithm with extensive simulation, in terms of both functionality and practicality. The results show: The algorithm enables efficient co-allocation of both computing and network resources provided by multiple domains, and can reflect reservation options for resource administrators issues as a first step. The calculation times needed for selecting resources using an IP solver are acceptable for an on-line service.

1 Introduction

Grid and network resource management technologies have enabled the construction of large-scale QoS-guaranteed Grid environments, which consist not only of performance-guaranteed multiple-computer clusters and storage resources, but also bandwidth-guaranteed networks linking the distributed resources. Several research projects have achieved coordination of resource managers for computers and network bandwidth and have constructed QoS-guaranteed Grid environments[1–3]. In contrast to canonical Grid environments, whose network resources are shared by abundant users, network links in these QoS-guaranteed Grids are dedicated to requesting users in order to guarantee the specified bandwidth.

In QoS-guaranteed Grid environments, each resource is managed by a local resource manager (RM) provided by several administrative domains or organizations, including commercial sectors. Therefore, each RM had better have an advance reservation capability, in order to provide a performance-guaranteed resource for a QoS-guaranteed Grid user, who also co-reserves other resources, including commercial resources. The KOALA[4] Grid scheduler and the QBETS[5] batch queue prediction service provide co-allocation of multiple cluster resources in coordination with RMs, without advance reservation, by acquiring and predicting the status of RMs. However, these strategies cannot guarantee to allocate the resources at the same time, so that the co-allocation user may have to pay for some resources, even if one resource may not be allocated at the expected time.

Therefore, “advance reservation” is one of the key technologies for a QoS-guaranteed Grid, and we have been working on development of the GridARS resource management framework[6] and the PluS plugin scheduler[7]. GridARS co-works with multiple RMs for computers, networks, and other resources, which manage the actual resources, and a reservation table of the resources, and co-allocates requested resources in advance for each QoS-guaranteed Grid user. PluS can be used in an RM and allows advance reservation on existing batch queuing systems, such as Sun GridEngine[8] and TORQUE[9], as well as Maui[10].

An important issue is then the question of Grid schedulers’ advance reservation-based co-allocation of many kinds of distributed resources provided by various organizations. For building a QoS-guaranteed Grid, co-allocation algorithms have to select not only computers and storage resources, but also network links between the selected resources. Also, all of detailed resource allocation information in each RM will not be disclosed via commercial services. Therefore, Grid schedulers for QoS-guaranteed Grids cannot apply either canonical Grid co-allocation algorithms[11, 4, 12] based on list-scheduling heuristic approaches, or network routing algorithms[13] based on Dijkstra’s algorithm, straightforwardly.

In addition, the co-allocation algorithms should reflect the following user and administrators scheduling options: In a user view, there should be options for resource co-allocation: (a) reservation time, (b) price, and (c) quality (availability). On the other hand, there should be options: (A) load balancing among RMs, (B) preference allocation to specific RMs because of energy savings or alliance issues, and (C) allocation suited for each user service level in an administrator view. Some studies[14–17] have already proposed advance reservation-based co-allocation algorithms for the both computer and network resources, but they have not adequately taken these options into account.

Furthermore, such a scheduling problem is known as NP-hard. It is important to determine co-allocation plans with short calculation time, especially for an on-line service.

We propose an on-line advance reservation-based co-allocation algorithm for both computing and network resources on QoS-guaranteed Grids. The goal of our algorithm is to create reservation plans satisfying user resource requirements and to take the above co-allocation options in the user and administrator issues into consideration.

In the proposed algorithm, our Grid scheduler (1) receives limited dynamic resource information from related RMs, (2) selects multiple combinations of suitable resources using the information, and (3) co-allocates the resources based on the selections. In phase (2), we modeled the co-allocation problem as an integer programming (IP) problem and applied IP solvers. We also describe how to apply the user and administrator options to these phases. This proposed algorithm could also be applied to co-allocation without advance reservation.

We evaluate the proposed algorithm in our advance reservation-based co-allocation model with extensive simulation, and show the validity of the algorithm in terms of functionality and practicality. Experiments on functional issues show that our algorithm

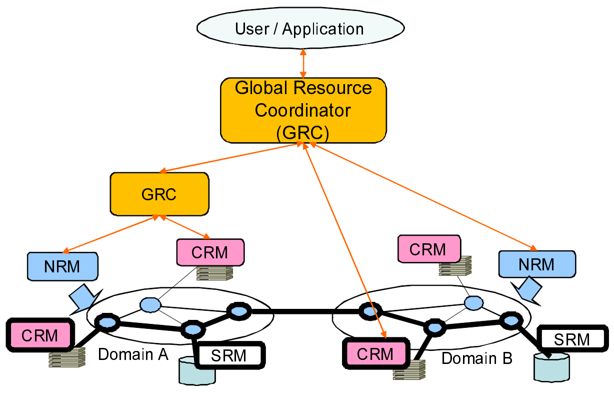


Fig. 1. Overview of the Resource Management Framework.

enables efficient co-allocation of both computing and network resources provided by multiple domains, and can take administrator co-allocation options as a first step. In the experiments on practical issues, the calculation times of the proposed co-allocation method are acceptable for an on-line service.

2 Advance reservation-based co-allocation model

2.1 Overview of the resource management framework

To enable a QoS-guaranteed Grid, we have been developing a Resource Management Framework called 'GridARS', as shown in Figure 1. Each of the domains, A and B, in Figure 1 denotes a network domain managed by a single administrative organization. This GridARS framework provides users with a QoS-guaranteed Grid, which spans several management domains, and is based on advance reservation.

The GridARS framework consists of a Global Resource Coordinator (GRC), which behaves as a Grid Scheduler, and Resource Managers (RMs), which manage each local resource. GRC and RM work together to provide users a QoS-guaranteed Grid. NRM, CRM, and SRM in Figure 1 denote Resource Managers for Networks, Computers, and Storages, respectively. More than one GRC is allowed in a single system. GRCs could be configured in a coordinated hierarchical manner, or in parallel, where several GRCs compete for resources with each other on behalf of their users. Some GRCs have a co-allocation planning capability, called Planner. Based on the reservation plans produced by a Planner, GRCs will perform resource reservation on subordinate GRCs or RMs.

2.2 User requests

We have performed several experiments on our QoS-guaranteed Grid, where we co-allocated several computing clusters, and light-path networks between those clusters,

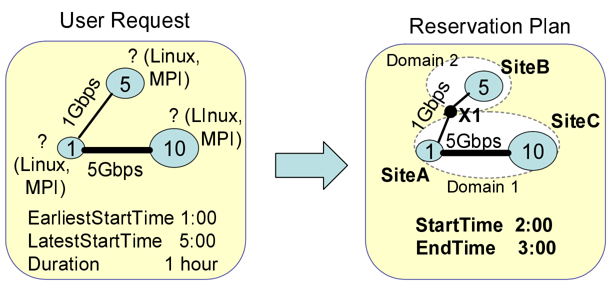


Fig. 2. User's resource request and a reservation plan from the Planner.

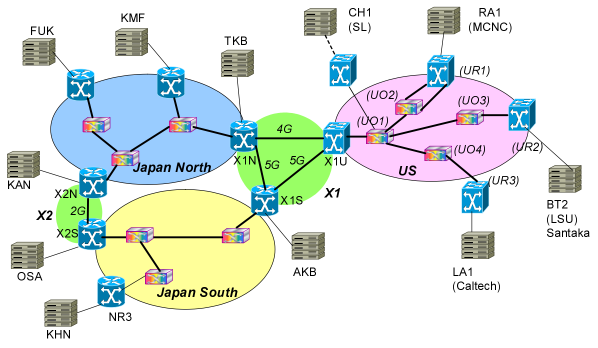


Fig. 3. Resources used for an experiment with G-lambda and EnLIGHTened.

and ran real applications [1, 2, 18]. The applications include a molecular dynamics simulation program with GridRPC and MPI, and HD video live-streaming. The goal of the co-allocation algorithm proposed in this paper is to create resource reservation plans for simultaneous co-allocation.

On the left of Figure 2, we show a resource request from a user. On the right hand side, we show a plan generated by GRC Planner. For computing resources, users can specify the number of sites, the number of CPUs or cores for each site, and other attributes such as OSs. For network resources, users can specify bandwidth between the computing resource sites, latency, and other attributes, such as media types and availability. Users can also specify a time frame for each resource. In Figure 2, we specify EarliestStartTime (EST), LatestStartTime (LST), and Duration (D), where the user wants to reserve a time slot D units long, start after EST and before LST, i.e., finish before $LST + D$.

The GRC Planner gets a request from a user, selects resource groups from a resource pool, as shown in Figure 3, and creates reservation plans. Figure 3 shows a real resource pool used for an experiment performed by Japan's G-lambda project [19] and the United States' EnLIGHTened Computing project[20]. The two projects achieved the world's first inter-domain coordination of resource managers for in-advance reservation of network bandwidth and compute resources between and among both the US and Japan in the fall of 2006[2].

The reservation plan shown on the right hand side of Figure 2 demonstrates how computing resources (SiteA, SiteB, SiteC) and network resources between them are allocated and the start time and end time of the time slot are determined. Note that network topology produced by a Planner is not real network topology with real routers and switches, but an abstracted higher-level notation. This is because NRMs will be provided by the commercial sector, and abstract away the underlying real network configuration. In the planned topology, a network in a single domain is denoted as just a path. When the network spans several domains, it will be denoted as a set of paths connected together. In Figure 2, the network between SiteA and SiteC is denoted as a single path in Domain1, while the network between SiteA and SiteB is denoted as two paths in Domain1 and Domain2, connected at the domain exchange point X1.

2.3 Retrieving available resource information from RMs

In order to have reservation planning, GRC has to retrieve available resource information for the future. In our co-allocation model, we assume that RMs will be provided by providers in the commercial sector, who will not disclose all the available resource information, including reservation time tables. The G-lambda project, which is a collaboration between industrial and governmental laboratories, AIST, KDDI R&D Laboratories, NTT, and NICT, has defined a web services-based resource reservation interface called GNS-WSI, which takes account of commercial services. GNS-WSI provides operations retrieving available resource information as well as reservation operations. We use the GNS-WSI retrieving operations, in which a requester has to specify a time frame to get available resource information.

3 An advance reservation-based co-allocation algorithm

3.1 The stages of resource co-allocation

We propose an on-line advance reservation-based co-allocation algorithm with the goal of creating reservation plans satisfying user resource requirements. The algorithm is invoked at every reservation request arrival.

The stages of reservation planning and resource co-allocation in GRC are as follows:

1. GRC receives a co-allocation request from a user.
2. GRC Planner creates multiple reservation plans for the request.
   - 2i Planner selects N laddered time frames from $[EST, LST + D]$.
   - 2ii The Planner retrieves available resource information results at N time frames from RMs.
   - 2iii Using this available resource information, the Planner determines N' ($N' \leq N$) reservation plans, based on a co-allocation method described in the next section.

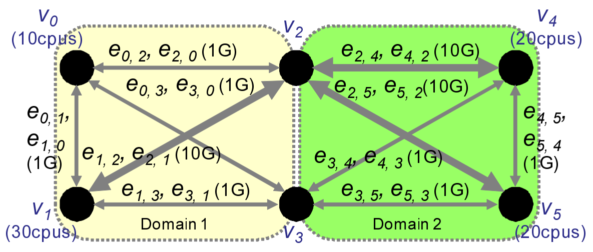


Fig. 4. Resources denoted as a graph.

- 2iv The Planner sorts N' plans by suitable order, which depends on co-allocation options in user and administrator issues.

3. In accordance with the reservation plans created by Planner, GRC tries to co-allocate the selected resources in cooperation with the subordinate RMs.
4. GRC returns co-allocation results, whether the resource co-allocation has succeeded or not, to the user. If it has failed, the user will resubmit a request with updated resource requirements.

As described in Section 2.2, a user can specify an exact reservation time or a range using the EST, LST, and D parameters. In the former case, GRC Planner creates reservation plans at the specified time frame. In the latter case, the Planner seeks available resources of available time frames in $[EST$,$LST + D]$. Therefore, the Planner creates multiple plans in stage 2.

In stage 2i, it is possible to allow GRC administrators to make a trade-off between creating more suitable reservation plans with a large N and small planning cost with a small N. In stage 2ii, multiple query results are retrieved by a single query operation, using the GNS-WSI interface described in Section 2.3. In addition, GRC Planner can send queries to subordinate RMs concurrently. In stage 2iii, N reservation planning can running concurrently. Co-allocation options shown in stage 2iv are described in Section 3.4.

3.2 The co-allocation method based on a general optimization problem

We propose a co-allocation method for both computing and network resources, modeled as an integer programming (IP) problem. This method is applied in stage 2iii.

Resource Notation We denote resources as a directed graph $G = (V, E)$, as shown in Figure 4, where V is a set of vertices in G and E is a set of edges in G. Vertex v_q denotes a computing resource site or a network exchange point between network domains. Edges $e_{o,p}$ and $e_{p,o}$ denote network paths managed by NRMs. In Figure 4, there are two network domains (Domain1 and Domain2), which provide network paths. Here, $e_{o,p}$ denotes an edge from v_o to v_p, while $e_{p,o}$ denotes an edge from v_p to v_o. Parentheses attached to a vertex denote the number of available CPUs (or cores) at the sites, which will be referred to as $wc_i(i \in V)$. Parentheses attached to an edge denote the bandwidth of the path, which will be referred to as $wb_k(k \in E)$. Note that v_2 and v_3 in Figure 4 are network exchange points, which do not have any CPUs. We could add more attributes on vertices and edges, such as network latency or availability. The values per unit of each CPU and bandwidth are denoted as $vc_i(i \in V)$ and $vb_k(k \in E)$, respectively. These values will be prices or cumulative points to reflect co-allocation options. Note that wb_k and vb_k are shared by $e_{o,p}$ and $e_{p,o}$.

Resource Request Notation Next, we denote a resource request from a user as a complete graph $G_r = (V_r, E_r)$, where V_r denotes required compute sites, and E_r denotes edges between V_r. The number of CPUs, provided by each compute site, and the network bandwidth are denoted as $rc_j(j \in V_r)$ and $rb_l(l \in E_r)$.

Modeling as a mixed integer programming problem Now, we can plan resource reservation as the 0-1 integer programming (0-1 IP) problem to determine the following variables, with the parameters shown above. "0-1 IP" aims to find a combination of binary (0 or 1) variables to minimize or maximize an objective function subject to linear constraints.

$$x_{i,j} \in \{0,1\} \qquad (i \in V, j \in V_r) \tag{1}$$

$$y_{k,l} \in \{0,1\} \; (k = (m,n) \in E, m,n \in V, \; l = (o,p) \in E_r, o,p \in V_r) \tag{2}$$

$x_{i,j}$ describes computing resource allocation, 1 means the requested resource denoted by the column is allocated to the actual resource denoted by the row. $y_{k,l}$ describes network resource allocation, 1 means the network path is taken, while 0 means it is not.

The objective function and constraints are described as follows:

$Minimize$

$$\sum_{i \in V, j \in V_r} vc_i \cdot rc_j \cdot x_{i,j} + \sum_{k \in E, l \in E_r} vb_k \cdot rb_l \cdot y_{k,l} \tag{3}$$

$Subject\ to$

$$\forall j \in V_r, \sum_{i \in V} x_{i,j} = 1 \tag{4}$$

$$\forall i \in V, \sum_{j \in V_r} x_{i,j} \le 1 \tag{5}$$

$$\forall i \in V, \sum_{j \in V_r} rc_j \cdot x_{i,j} \le wc_i \tag{6}$$

$$\forall l \in E_r, \sum_{k \in E} y_{k,l} \begin{cases} \ge 1 \; (rb_l \ne 0) \\ = 0 \; (rb_l = 0) \end{cases} \tag{7}$$

$$\forall k \in E, \sum_{l \in E_r} rb_l \cdot y_{k,l} \leq wb_k \tag{8}$$

$$\forall l = (o,p) \in E_r, \forall m \in V,$$
$$\sum_{n \in V, m \neq n} y_{(n,m),(o,p)} - \sum_{n \in V, m \neq n} y_{(m,n),(o,p)}$$
$$= \begin{cases} x_{m,o} - x_{m,p} & (rb_l > 0) \\ 0 & (rb_l = 0) \end{cases} \tag{9}$$

The objective function Equation (3) is meant to minimize the sum of the selected compute and network resources values.

Equations from (4) to (6) are constraints on computing resources, while Equations (7), (8) are constraints on network resources. Equation (9) is a constraint on both computing and network resources.

Equation (4) ensures each compute site request j will be allocated on just one site. Equation (5) ensures each real site i will not be allocated to more than two sites. Equation (6) ensures each allocated site i has more CPUs than the required number.

Equation (7) denotes that for a path l, the sum of $y_{k,l}$ is more than 1 when a user requests bandwidth on path l, and 0 when a user does not. The sum will become 1 if the path is included in a single domain, and become n if the path spans n domains. Equation (8) denotes real path k can provide more bandwidth than required.

Equation (9) is derived from the mass balance constraints[21], which claim that at any vertex on a graph, total inflows plus generation on the vertex are equal to total outflows. Assume a path of flow f with one intermediate node between start and end. Here, generations are f from the start point, $-f$ from the end point, and 0 from the intermediate node of the path. Assume we have a bandwidth reservation request for path l. From application of the mass balance constraint with flow $f = 1$, for each path $l = (o,p)$ (o denotes a start point and p denotes an end point of l) and each m (a computing resource site or a network exchange point), we obtain Equation (9). The value of Equation (9) will be 1 if m is the start point, and -1 if m is the end point, and 0 if m is the others. Here, $x_{m,o} = 1$ when m is the start point, $x_{m,p} = 1$ when m is the end point, and $x_{m,o} = x_{m,p} = 0$ when m is neither the start nor end point. Therefore, the right of Equation (9) could be represented as $x_{m,o} - x_{m,p}$. Thus, this equation ties $x_{i,j}$ and $y_{k,l}$ together.

The proposed co-allocation method, based on a general optimization problem, could be applied to co-allocation without advance reservation. It is also effective when some of the resources are specified by the users in advance.

3.3 Additional constraints for optimization

Generally, calculation times of general optimization problems, including 0-1 IP, become exponentially long when the number of variables becomes large, due to NP-hard. We propose additional constraints, which are expected to make calculation times of our co-allocation method shorter.

$Subject\ to$

$$\forall l \in E_r, \forall m,n \in V(m \neq n),$$

$$y_{(m,n),l} + y_{(n,m),l} \leq 1 \tag{10}$$

$$\forall l \in E_r,\ \sum_{k \in E} y_{k,l} \leq N_{max} \tag{11}$$

Equation (10) indicates that both of the directed edges between the same two points, (m, n) and (n, m), are not selected in each requested network. Equation (10) enables solvers to avoid redundant search for an optimal solution. Equation (11) specifies N_{max}, the maximum number of paths, which make up each requested network. Here, N_{max}, given heuristically, makes the search area smaller and calculation time prospects shorter, although we might not be able to find an optimal solution, whose network consists of more than N_{max} paths.

3.4 Reflecting co-allocation options in the algorithm

As mentioned in Section 1, there are co-allocation options in user and GRC administrator issues: A user uses her co-allocation option to prioritize (a) reservation time, (b) price, and (c) quality (availability), in addition to general resource requirements. A GRC administrator has options to prioritize (A) load balancing among RMs, (B) preference allocation to specific RMs, and (C) allocation suited for each user service level.

These co-allocation options can be reflected in the proposed algorithm as follows: For option (a), we sort reservation plans by late reservation time in stage 2iv. For (b), we set the values vc_i and vb_k to CPU and bandwidth unit prices and sort plans by the total price in stage 2iv. For (c), we set vc_i and vb_k to their points, such as levels of fault tolerance, and sort plans by the total points in stage 2iv.

To fulfill the administrator's options (A) and (B), we have to weight each resource and add other objective functions. Option (C) could be handled by modification of the available resource retrieval information, which reflects service level requirements from the users.

4 Experiments

4.1 Simulation model

We conduct simulations to investigate the validity of our co-allocation algorithms, in terms of functionality and practicality. In the experiments on the functional issues, we investigate if the algorithm can schedule both computing and network resources from multiple domains efficiently, and if our algorithm can take co-allocation options in user and administrator issues into consideration. In the experiments on the practical issues, we compare the calculation times of our algorithm with/without additional constraints, Equation (10) and Equation (11), applying different IP solvers.

In the both simulations, we assume the experimental environment shown in Figure 3, used in the EnLIGHTened and G-lambda (ELGL) experiments. The environment consists of three network domains and two domain exchange points, as shown in Figure 5. In-domain computing resource sites, denoted by black circles, are inter-connected by

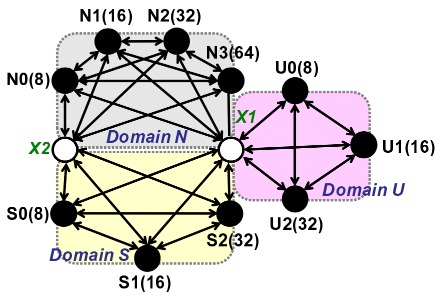


Fig. 5. Simulation environment.

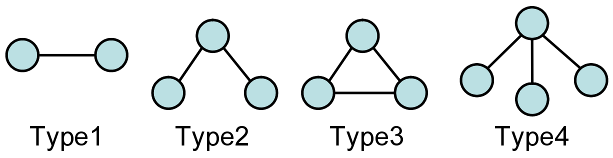


Fig. 6. Resource requirement types.

a complete graph and each domain exchange point, denoted by a white circle, and each in-domain site is connected to every other, respectively.

An overview of simulation settings is given in Table 1. In our simulations, there are two users, UserA and UserB, and each user requests resource co-allocation, repeatedly, as shown in Figure 6. Each user request arrives in the first 24 hours and it reserves resources for the next 24 hours. Interarrival rate of each user request is set to 407.327 [sec], so that the request loads are set to 10 [%] at 144 [min] to 100 [%] at 1440 [min]. The number of reservation plans N in the GRC Planner is set to 10. For each request, co-allocation plans are sorted by reservation time, and applied the (a) reservation time option.

In the experiments, we assume a smallish numbers of CPUs at each site (8 - 64 CPUs) and the requested site (1-8), because the calculation times of our algorithm does not depend on the number of CPUs, but on the number of sites.

4.2 Experiments on functional issues

First, we compare success ratios of resource co-allocation among two users, UserA and UserB and investigate the functionality needed to reflect the administrator option (C). In this experiment, we used GLPK (GNU Linear Programming Kit)[22] as a solver for 0-1 IP in the proposed algorithm. We assume that the users co-allocation option is (a), and the administrator options are (A) and (C). In the experiments with option (C), the service level (SL) of UserB is set to low: UserB can book half of the available resources, while UserA can book all of them.

Table 1. Simulation settings.

Simulation environment settings	
Configuration	No. of GRC=1, No. of NRM=3 (N, S, U), No. of CRM=10
No. of sites / domain name	4/N, 3/S, 3/U
Domain exchange points	X1 (N, S, U), X2 (N, S)
No. of CPUs	N{8, 16, 32, 64}, S{8, 16, 32}, U{8, 16, 32}
CPU unit value	1
Bandwidth [Gbps]	in-domain paths : 5, inter domain paths : 10
Bandwidth unit values	in-domain paths : 5, inter domain paths : 3
Resource requirement settings	
Users	UserA, UserB
Resource requirement types	Type1,2,3,4 (Uniform distribution)
Requested No. of CPUs	1, 2, 4, 8 for all sites in Type1,2,3,4 (Uniform distribution)
Requested bandwidth [Gbps]	1 for all paths in Type1,2,3,4 (Fixed)
Interval of each user request	Poisson arrivals
Reservation duration [min]	30, 60, 120 (Uniform distribution)
LST - EST	Reservation duration $\times$ 3

Figure 7 shows success ratios of resource co-allocation, requested by UserA and UserB, respectively. The horizontal axis indicates elapsed time in each simulation and the vertical axis indicates the success ratio. Each plot shows the average success ratio of requests that arrived between 0 and 144 [min] to between 1296 and 1440 [min] in 10 simulation runs, respectively. The request load is 0-10 [%] between 0 and 144 [min] and 90-100 [%] between 1296 and 1440 [min]. "UserA" and "UserB" denote UserA and UserB, and "-N" and "-S" denote results with option (A) and (C) applied. UserB is set to a low SL.

The results of a normal case ("-N") show that success ratios of UserA and UserB are 0.918 and 0.897, when the request load = 50 [%] (720 [min]), and still 0.618 and 0.609, when the load = 80 [%] (1152 [min]). The main result here is that the proposed algorithm is effective for co-allocation of multiple computing and network resources spanning over multiple network domains.

In comparison of service levels, success ratios of UserA and UserB are comparable in the results with option (A) ("-N") applied. On the other hand, UserA's results show better success ratios, 0.595 when request load = 100 [%], than UserB's results, 0.374, in the option (C) results. Therefore, Figure 7 shows that the algorithm can take option (C) into consideration.

Next, we compare the co-allocation results with administrator options (A) and (B). In the cases with option (B), specific sites are prioritized by the weights of CPUs. Figure 8 shows the results of applying option (A) (top), option (B) prioritized by the number of CPUs in each site (middle), and option (B) prioritized by network domains (bottom). Each CPU unit value is set to 1 in the top cases, 1, 10, 100, 1000 for 64, 32, 16, 8 CPU

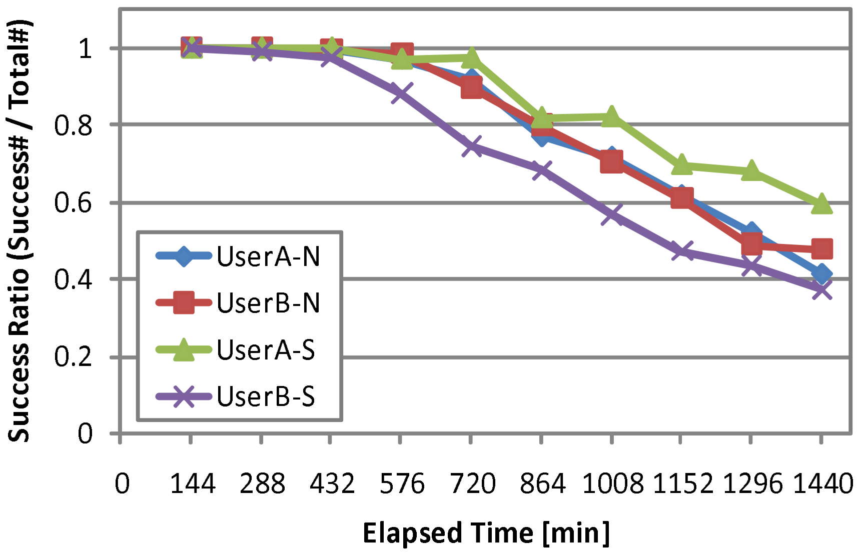


Fig. 7. Comparison of resource co-allocation success ratios. The request load varies from 10 [%] to 100 [%].

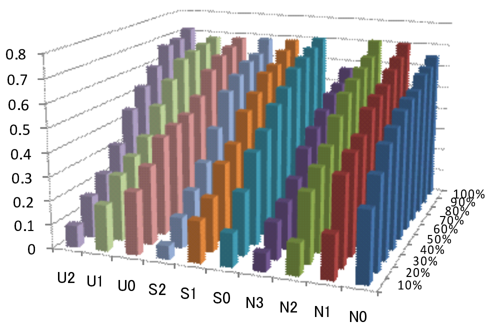
sites in the middle cases, and 1, 10, 100 for domain N, S, U sites in the bottom cases. Our algorithm selects resources to minimize total resource weight.

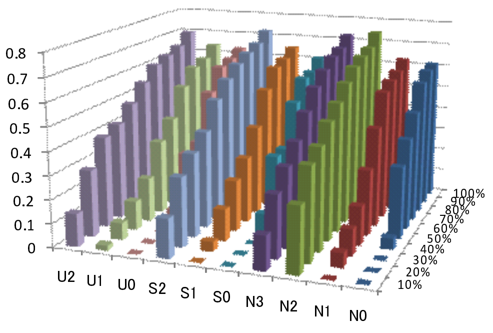
The simulation results show that load averages of the top graph increase almost uniformly. On the other hand, sites with many CPUs and sites belonging to domain N are preferentially selected in the middle and bottom graphs if the total load of resource requests is not high. Therefore, the experimental results prove that our algorithm can takes co-allocation options in administrator issues into consideration.

4.3 Experiments on practical issues

The goal of our algorithm is to be used in an on-line service. However, our algorithm is modeled as 0-1 IP, and so its calculation time becomes drastically long when the number of valuables becomes large, due to NP-hard. Therefore, we confirm our algorithm is practical when used to compare the calculation times of our algorithms with/without additional constraints in Section 3.3, applying different IP solvers.

In the comparison of IP solvers, we apply free open source solvers, GLPK (GNU Linear Programming Kit)[22] and a satisfiability problem (SAT) based solver, Sugar++[23] with a SAT solver, MiniSat[24]. Sugar++ enables a SAT solver to solve an optimization problem, which maximizes or minimizes its objective functions. Sugar++ temporally determines the maximum or minimum value of the objective function and solves a SAT problem using the SAT solver, repeatedly. Then, Sugar++ finds an optimal solution.





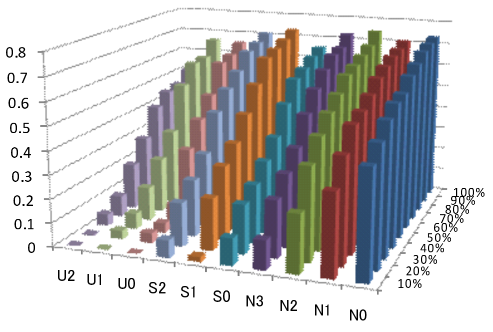


Fig. 8. Comparison of load averages in each site. Three patterns of administrator's options are applied, option (A) (top), option (B) prioritized by the number of CPUs in each site (middle), and option (B) prioritized by network domains (bottom). For each axis, N0 - U2 indicate compute resource site names, 10 % - 100 % indicate request load, and 0 - 0.8 indicate load average of each site, respectively.

Table 2. Comparison of calculation times of 0-1 IP.

	Avg [sec]	Max [sec]	σ
GLPK	0.779	8.492	1.721
GLPK-st	0.333	4.205	0.700
MiniSat-st	12.848	216.434	27.914
MiniSat-st-1	1.918	2.753	0.420

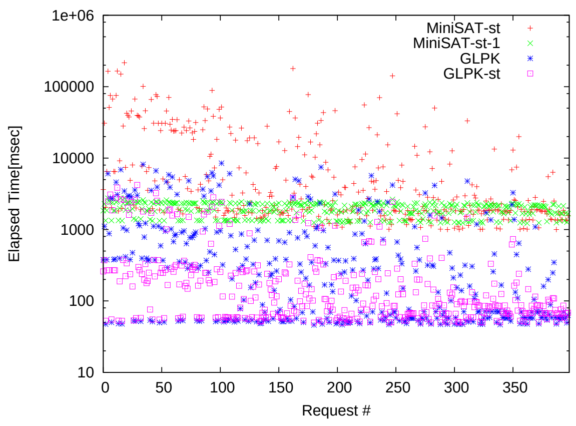


Fig. 9. Comparison of the average calculation times for each resource request. The vertical axis indicates elapsed time in log scale.

Experimental results of calculation times We compare four patterns of solvers and constraints as follows:

GLPK: GLPK is applied.
GLPK-st: GLPK with additional constraints in Section 3.3 is applied.
MiniSat-st: Sugar++ and MiniSat with the additional constraints are applied.
MiniSat-st-1: Sugar++ and MiniSat with the additional constraints are applied, however, this solves a SAT problem once only, and does not obtain an optimal solution.

Table 2 shows the average, maximum, and standard deviation (σ) of calculation values after applying the different combinations shown above. The results of GLPK and GLPK-st show that GLPK-st is twice as fast than GLPK without additional constraints. From the results one can see that much improvement can be gained by applying additional constraints for IP problems. In our comparison of solvers, IP-based GLPK-st and SAT-based MiniSat-st, the GLPK-st results show much shorter times than the MiniSat-st ones. The results here indicate IP-based solvers are quite suitable for our scheduling problems. However, MiniSat-st-1 shows the best performance of all combinations, in terms of the maximum values and standard deviations.

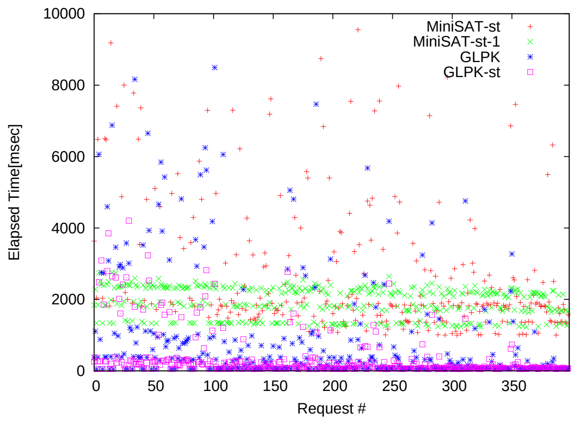


Fig. 10. Comparison of the average calculation times for each resource request. The graph shows the results from 0 to 10 [sec].

Next, we wish to compare the average calculation times for each request in Figure 9 and Figure 10. The horizontal axis denotes the request number and each plot is the average calculation time of $N = 10$ reservation plans for each request, because these N plans can be solved independently in the embarrassingly parallel (EP) manner. The vertical axis of Figure 9 denotes elapsed time in log scale and the results from 0 to 10 [sec] are shown in Figure 10.

Figure 9 indicates that the calculation times of solvers needed to obtain an optimal solution are rather dispersed, while those of MiniSat-st-1 are not. However, the dispersion decreases when the request number becomes large. The main results here indicate that the search areas of IP problems become small and the calculation times decrease, when the available resources decrease.

In Figure 10, one can see the three lines of the results of applying MiniSat-st-1. Therefore, the calculation time of MiniSat-st-1, which satisfies all of the constraints, but does not obtain an optimal solution, is proportional to the number of vertices in Figure 6.

Discussion There are lots of scheduling studies applying a sort of heuristic method, because scheduling problems are known as NP-hard. On the other hand, the coverage of IP problems is expanding, because of the recent rapid increase in computer performance and the improvement of IP algorithms and solvers. Also, the IP calculation times can be reduced by applying additional constraints and approximate solutions, which does not obtain an optimal solution. Commercial IP solvers, such as ILOG CPLEX[25], are also known to reduce calculation times by applying additional constraints in pre-processing. In addition, approximate solutions can provide a solution, which is not optimal, but close to an optimal solution, with a short calculation time. Therefore, approximate solutions seem efficient for scheduling problems, which do not need an optimal solution.

While the number of variables in the proposed algorithm becomes N^3 depending on the number of computer sites N, our co-allocation problem has the following characteristics:

– The search area of a single GRC can be localized, because GRCs are located hierarchically as shown in Figure 1.
– The number of variables scales by the number of computer sites, not computers.
– In practical use, additional constraints will be defined, such as those for communication latencies, resource hardware requirements, and execution environments.

Consequently, modeling as an IP problem is an effective approach for our problem.

5 Related work

There are several Grid scheduling algorithms for both computing and network resources. The differences between our algorithms are described as follows:

The VIOLA project has work on development of the MetaScheduling Service (MSS)[14], which co-allocates both computing and network resources, based on advance reservation. Roblitz proposed a Grid scheduling algorithm of co-reservation for multiple resources, based on general optimization problems[15]. The differences between the above two algorithms and ours are: their GRC can obtain all of the reservation time tables managed by local resource managers and their algorithms assume a simple network resource model, such as a single domain and single switch configuration.

Ando and Aida proposed a Grid scheduling algorithm for both computing and network resources, and modeled a single domain network and multiple switch configuration[16]. Their algorithm, based on a general backtrack approach, reserves computers and the related network paths incrementally, releases the reserved resources when the next required resource could not be discovered, and then finds the next candidate. This results in a complicated co-allocation process and blocking of many resources during the process.

Elmroth and Tordsson also propose a co-allocation algorithm[17] for NorduGrid[26]. Their algorithm fixes a reservation time and searches a combination of required computers first, and the related network paths next. If it cannot find the requested resources, it slides the reservation time frame and searches for resources in the same manner, repeatedly. This approach causes a long planning time, when constraints of the latter resources are strict, such as less network bandwidth available, and when resource co-allocation is failed, while our approach does not.

Both backtrack and NorduGrid approaches were not able to find suitable resources, e.g., due to expensive price, long communication latency, and redundant path networks, because they select the first found resources. On the other hand, our algorithm can take co-allocation options in user and administrator issues into consideration and find suitable resources.

Netto and Buyya proposed automatic rescheduling of multiple co-allocation requests of computing resources based on advance reservation[27], in order to achieve high utilization. However, it is difficult to reschedule various allocated resources automatically, when the number of allocated resources, including networks, becomes larger and the resources are provided by commercial entities, which do not disclose the detailed reservation time tables and also charge for the resources.

Rescheduling is an important issue for a QoS-guaranteed Grid not only to increase system utilization but to recover failures. Our approach is that our monitoring system[28] provides a user monitoring information on the reserved resources and the user can send a modification request to our co-allocation system, if required. The proposed algorithm can be applied to such a modification request.

6 Conclusions and future work

We propose an on-line advance reservation-based co-allocation algorithm for both computing and network resources on QoS-guaranteed Grids, constructed over multiple network domains. The proposed IP-based algorithm can create reservation plans satisfying user resource requirements and takes co-allocation options in user and administrator issues into consideration. The proposed algorithm could also be applied to co-allocation without advance reservation.

Our experimental results showed the validity of the proposed algorithm, in terms of both functionality and practicality: Our algorithm enables efficient co-allocation of both computing and network resources provided by multiple domains, and can reflect reservation options in administrator issues. The calculation time needed for selecting resources is acceptable for an on-line service.

For future work, we will improve our algorithm and conduct further experiments on the scalability with more actual constraints, such as communication latencies, resource hardware requirements, and execution environments. We also plan to apply sophisticated economy models for resource pricing and SLA models on resource provider sides, and will confirm that our algorithm can also take user co-allocation options efficiently under these situations.

Acknowledgments

We would like to thank Prof. Naoyuki Tamura and Mr. Tomoya Tanjo from Kobe University, and Prof. Katsuki Fujisawa and Mr. Yuichiro Yasui from Chuo University for their generous support with respect to optimization problems.

This work was partly funded by KAKENHI 21700047 and the National Institute of Information and Communications Technology.

References

1. Takefusa, A., Hayashi, M., Nagatsu, N., Nakada, H., Kudoh, T., Miyamoto, T., Otani, T., Tanaka, H., Suzuki, M., Sameshima, Y., Imajuku, W., Jinno, M., Takigawa, Y., Okamoto, S., Tanaka, Y., Sekiguchi, S.: G-lambda: Coordination of a Grid Scheduler and Lambda Path Service over GMPLS. Future Generation Computing Systems **22(2006)** (2006) 868–875
2. Thorpe, S.R., Battestilli, L., Karmous-Edwards, G., Hutanu, A., MacLaren, J., Mambretti, J., Moore, J.H., Sundar, K.S., Xin, Y., Takefusa, A., Hayashi, M., Hirano, A., Okamoto, S.,

Kudoh, T., Miyamoto, T., Tsukishima, Y., Otani, T., Nakada, H., Tanaka, H., Taniguchi, A., Sameshima, Y., Jinno, M.: G-lambda and EnLIGHTened: Wrapped In Middleware Co-allocating Compute and Network Resources Across Japan and the US. In: Proc. Grid-Nets2007. (2007)

3. : AAA scenarios and test-bed experiences. Deliverable reference number:d4.2, The PHOSPHORUS project (2008) http://www.ist-phosphorus.eu/ files/deliverables/Phosphorus-deliverable-D4.2.pdf.
4. Mohamed, H., Epema, D.: Experiences with the KOALA Co-Allocating Scheduler in Multiclusters. In: Proc. 5th IEEE/ACM Int'l Symp. on Cluster Computing and the GRID (CC-Grid2005). (5 2005)
5. Nurmi, D., Brevik, J., Wolski, R.: QBETS: Queue Bounds Estimation from Time Series. In: Proc. 13th Workshop on Job Scheduling Strategies for Parallel Processing. (2007)
6. Takefusa, A., Nakada, H., Kudoh, T., Tanaka, Y., Sekiguchi, S.: GridARS: An Advance Reservation-based Grid Co-allocation Framework for Distributed Computing and Network Resources. In: Job Scheduling Strategies for Parallel Processing. Volume 4942/2008., Springer Berlin / Heidelberg (4 2008) 152–168
7. Nakada, H., Takefusa, A., Ookubo, K., Kishimoto, M., Kudoh, T., Tanaka, Y., Sekiguchi, S.: Design and Implementation of a Local Scheduling System with Advance Reservation for Co-allocation on the Grid. In: Proceedings of CIT2006. (2006)
8. Sun Grid Engine: `http://gridengine.sunsource.net/`
9. TORQUE Resource Manager: `http://www.clusterresources.com/resource-manager.php`
10. Maui Cluster Scheduler: `http://www.clusterresources.com/pages/products/maui-cluster- scheduler.php.`
11. Czajkowski, K., Foster, I., Kesselman, C.: Resource co-allocation in computational grids. In: In Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing (HPDC-8), IEEE Computer Society (1999) 219–228
12. Castillo, C., Rouskas, G.N., Harfoush, K.: Resource Co-Allocation for Large-Scale Distributed Environments. In: Proc. HPDC2009. (2009) 137–150
13. Taesombut, N., Chien, A.A.: Evaluating Network Information Models on Resource Efficiency and Application Performance in Lambda-Grids. In: Proc. SC07. (Nov 2007)
14. Barz, C., Pilz, M., Eickermann, T., Kirtchakova, L., Waldrich, O., Ziegler, W.: Co-Allocation of Compute and Network Resources in the VIOLA Testbed. TR 0051, CoreGrid (9 2006)
15. Roblitz, T.: Global Optimization For Scheduling Multiple Co-Reservations In The Grid. In: Proc. CoreGRID Symposium. (8 2008) 93–109
16. Ando, S., Aida, K.: Evaluation of Scheduling Algorithms for Advance Reservations. In: IPSJ SIG Notes 2007-HPC-113. (2007) 37–42
17. Elmroth, E., Tordsson, J.: A standards-based Grid resource brokering service supporting advance reservations, coallocation and cross-Grid interoperability. Concurrency and Computation: Practice and Experience **25**(18) (2009) 2298–2335
18. Kudoh, T.: GRID computing and a role of photonic networks. Proc. SPIE Asia Pacific Optical Communications (APOC2008) **7137, 713713** (2008)
19. The G-lambda project: `http://www.g-lambda.net/`
20. The EnLIGHTened Computing project: `http://enlightenedcomputing.org/`
21. Ahuja, R.K., Magnanti, T.L., Orlin, J.B. In: Network Flows: Theory, Algorithms, and Applications. Prentice Hall (1993)
22. GLPK (GNU Linear Programming Kit): http://www.gnu.org/software/glpk/glpk.html
23. Tanjo, T., Tamura, N., Banbara, M.: Sugar++: A SAT-based Max-CSP/COP Solver. In: Proc. the Third International CSP Solver Competition. (2008) 144–151
24. MiniSat: http://minisat.se/

25. ILOG CPLEX: http://www.ilog.co.jp/product/ opti/cplex/cplex.html
26. NorduGrid: `http://www.nordugrid.org/`
27. Netto, M.A.S., Buyya, R.: Rescheduling Co-Allocation Requests based on Flexible Advance Reservations and Processor Remapping. In: Proc. Grid2008. (2008) 144–151
28. Takefusa, A., Nakada, H., Yanagita, S., Okazaki, F., Kudoh, T., Tanaka, Y.: Design of a Domain Authorization-based Hierarchical Distributed Resource Monitoring System in co-operation with Resource Reservation. In: Proc. HPC Asia 2009. (3 2009) 77–84