社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－HPC－113 (7)
2007／12／7

グリッド上の事前予約スケジューリング手法の性能評価

安藤 誠士郎[†1] 合田 憲人[†2,†1]

グリッドアプリケーションの実効性能を保証するために計算資源とネットワーク資源を連携して事前予約するサービスが注目されているが，予約を行う資源を自動的に決定する手法についてはまだ深く議論されていない．本稿では計算資源とネットワーク資源の事前予約プランを自動生成するアルゴリズムの性能評価について述べる．本性能評価より，予約プラン作成と各資源の予約を並行して行い，予約可能時刻の早い資源に優先的に予約を行う方法が予約成功率で優れていることがわかった．

Evaluation of Scheduling Algorithms for Advance Reservations

Seishiro Ando [†1] and Kento Aida[†2,†1]

An advance reservation is effective way to run an application with a guarantee of QoS on heterogeneous and dynamic computing environment like the Grid. The service to make advance reservations for both computing and network resources on behalf of users reduces user's burdens to run their applications. However, such scheduling algorithms have not been well discussed. This paper presents preliminary evaluation of scheduling algorithms for advance reservations on the Grid. The simulation results exhibit the advantage of the incremented reservation algorithm with the time oriented selection policy.

1. はじめに

グリッドは，広域ネットワーク上に分散する様々な資源を用いた大規模な高性能計算を可能にする．しかしそれらの資源は多くの利用者に共有されるため，コアロケーションまたはワークフローアプリケーションの実効性能を保証するには事前予約によって複数の計算資源やネットワーク資源を同時に確保する必要がある．既にPBSPro[1]をはじめとする事前予約機能を備えたジョブスケジューラやネットワーク帯域を予め確保するシステム[2]が実装されているが，ユーザがグリッド上の膨大な資源情報を把握して予約プランを練り，複数の資源に事前予約を行うことは大変な作業である．G-lambdaプロジェクトは計算資源とネットワーク資源を同時に確保するシステムの実証実験を行っているが[3]，予約を行う資源を自動的に決定する手法についてはまだ深く議論されておらず，このような事前予約スケジューリング手法は非常に重要なテーマである．

これまでに複数の資源間で連携した事前予約スケジューリング手法が提案されてきた．しかしながら，そのほとんどは予約対象が計算資源かネットワーク資源のどちらかに限られており，グリッドに適用するにはいくつかの問題を残している[4)–9)]．さらにこれらの研究ではスケジューラが中央集権的であり，グリッド上で起こり得る複数のスケジューラが資源を取り合うような状況を想定していない．

本稿では，複数の計算資源とネットワーク資源間で連携して事前予約プランを自動生成する予約アルゴリズムを考案し，シミュレーションによる性能評価結果を示す．提案アルゴリズムは，コアロケーションまたはワークフローアプリケーション（各アプリケーションは複数ジョブから構成されている）を対象とし，予め設定された締切時刻（デッドライン）までに実行が終了するように資源の事前予約プランを作成する．また性能評価では，複数のスケジューラ間で予約の競合が発生する状況を再現してシミュレーションを行う．

2. シミュレーションモデル

本節では本稿で用いるシミュレーションモデルについて述べる．

2.1 資源モデル

本シミュレーションモデルにおいて，グリッドは事前予約機能を備えた計算資源とネットワーク資源から成る．計算資源はクラスタのような性能が均一なノー

†1 東京工業大学
Tokyo Institute of Technology
†2 国立情報学研究所
National Institute of Informatics

ドで構成されおり，ユーザまたはスケジューラからの要求（予約開始/終了時刻，CPU 数）に対して，要求を満足する資源量が確保できる場合に CPU の事前予約が行われる．ネットワークについてもバンド幅を予約する機能があり，複数のリンクからネットワークが構成される場合の予約可能な最大バンド幅は経路を構成するリンクの最小値となる．

2.2 アプリケーションモデル

本シミュレーションモデルでは，1 つのアプリケーションは複数のジョブで構成される．アプリケーションは MPI に代表されるコアロケーション型のアプリケーション，およびワークフローアプリケーションに分けられる．コアロケーション型アプリケーションは全てのジョブや通信が同時刻に開始されて同時刻に終了するアプリケーションである．ワークフローはジョブの出力が次のジョブの入力となるように計算が進むアプリケーションであるが，通信中は送受信先のジョブ両方が実行されていなければならない．ここでジョブはユーザが指定した数の CPU 上で並列実行されるものとする．

これらのアプリケーションの事前予約を行うために，ユーザは予約プラン要求を提出する．1 つの予約プラン要求は以下の複数の資源要求とデッドラインから構成される．

計算資源要求（ジョブ毎） 要求実行環境（OS），ジョブに割り当てる CPU 数，アプリケーションからみた相対的な予約開始時刻（アプリケーション中で最初に実行するジョブの予約開始時刻を 0 とした場合のオフセット），予約時間幅．

ネットワーク資源要求（ジョブ間の通信毎） 通信に割り当てるバンド幅，予約開始時刻（オフセット），予約時間幅，通信を行うジョブ．

アプリケーション終了時刻（デッドライン）

図 1 はユーザ A が 2 つの計算資源要求と 1 つのネットワーク資源要求から成る 2 ステップのワークフローアプリケーションの予約プラン要求を提出し，実際に計算資源 A と C，および AC 間のネットワークが予約されている様子を示している．尚，コアロケーション型アプリケーションの場合は図中の予約開始時刻（オフセット）st と予約時間幅 d は全ての資源要求それぞれについて同じ値が指定される．

2.3 サービスモデル

図 1 はシミュレーションにおける事前予約サービスの構成を示している．事前予約サービスは以下の 4 つのサービスにより構成される．

Local Information Service (LIS) サイト毎の計算資源の予約を管理し，予約空き時間照会等のサービスを提供する．例えば “10:00〜20:00 の間で計算資源 A の CPU20 個を 1 時間予約可能な時間帯はいつか” という問い合わせに答える機能や，事前予約要求を受けて実際に予約登録を行うサイト内予約スケジューラとしての機能を持つ．複数資源の同時予約はその手続きをトランザクションとして扱う必要があるため，2 相コミット[10] をサポートする．

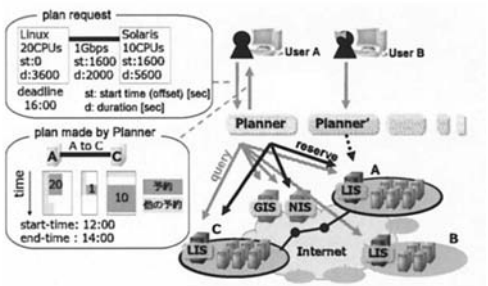

図 1 事前予約サービスの例

Global Information Service (GIS) グリッド上の計算資源情報を提供するサービスである．例えば “CPU を 10 個以上もつ計算資源はどのサイトにあるか” といった問い合わせに答える機能を持つ．これによって問い合わせ者は，資源の予約空き情報を得るためにはどの LIS に問い合わせればよいかを知ることができる．GIS で管理される資源情報はグリッドような動的な環境において絶えず変動するため，定期的もしくは資源構成変更時に LIS によって更新される必要がある．これは本質的に情報のタイムラグが生じることを示唆するが，本稿では GIS が持つ情報と実際の資源構成に差異はないものとする．

Network Information Service (NIS) ネットワーク資源の予約管理および予約情報を提供するサービスであり，例えば “10:00〜12:00 の間，計算資源 A，B 間のネットワークを 1Gbps だけ利用可能か” という問い合わせに答える機能を持つ．LIS と同じく予約処理は 2 相コミットをサポートする．

Planner ユーザから予約プラン要求を受け取って予約プランを作成し，各資源に対して予約を行うスケジューラに相当するサービスである．予約プランの作成は GIS・LIS・NIS の各サービスに問い合わせることで資源や予約情報を収集しながら行う．Planner はグリッド上に複数存在しており，互いに競合する状況が発生し得る．

ユーザは 2.2 節で述べた予約プラン要求を Planner に提出し，Planner から受け取った予約プランに従っ

```
00: st := 予約プラン要求到着時刻 +w_interval
01: rm_root := 探索開始位置となる計算資源要求
02: loop do
03:   /* 予約開始時刻 st における資源組み合わせの決定 */
04:   ret := search(rm_root, st)
05:   if ret = Δt then
06:     st := st + Δt
07:     continue
08:   else if ret = NOTFOUND then
09:     終了（予約プラン要求却下）
10:   else if ret = SUCCESS then
11:     予約プラン作成終了
12:   end if
13: end loop
```

図 2 アルゴリズムの概略

```
00: procedure search (rm, st)
01:   st_req, d_req : 資源要求 req のオフセットと予約時間幅
02:   t_bgn := st + st_rm /* rm の予約開始予定時刻 */
03:   C_rm := [t_bgn, t_deadline] にある rm の予約候補リスト.
              予約幅 d_rm. selection policy に従ってソート
04:   for each cm in C_rm do /* LABEL-1 */
05:     Δt := cm の予約開始時刻 −t_bgn /* Δt ≥ 0 */
06:     if Δt > 0 then return Δt
07:     FC_rm := cm
08:     RP_rm := rm 周りのネット資源要求リスト （注 1）
09:     for each rp in RP_rm do
10:       rm_next := rp の（rm でない）もう一つの端点
11:       if FC_rm_next は決定済み then
12:         t'_bgn := st + st_rp /* rp の予約開始予定時刻 */
13:         cp := 時間帯 [t'_bgn, t'_bgn + d_rp] の rp の予約候補
14:         if cp != null then FC_rp = cp, continue
15:         ret := NOTFOUND
16:       else
17:         ret :=search(rm_next, rm, st)
18:       end if
19:       if ret > 0 then return ret /* Δt */
20:       if ret = NOTFOUND then
21:         FC_rm と FC_rp, ∀rp ∈ RP_rm を取消
22:         continue LABEL-1
23:       end if
24:     end for
25:     return SUCCESS
26:   end for
27:   return NOTFOUND
28: end
```

FC_{req}： 仮決定した資源要求 req の予約候補

注 1： rm に隣接する複数の計算資源要求 rm_next について，まず FC_{rm_next} が定義済のものがランダムに，次に未定義のものが資源候補数について昇順に詰められる．

図 3 資源組み合わせを決定するバックトラック関数

て実際にグリッドアプリケーションの実行を行う．

3. スケジューリング手法

本節では，本稿で性能評価を行う事前予約スケジューリングアルゴリズムについて述べる．ユーザからの予約プラン要求は，計算資源要求を“頂点”，ジョブ間（2 頂点間）の通信に対応するネットワーク資源要求をそれらを結ぶ“辺”とする無向グラフにより表すことができる．本アルゴリズムでは，この無向グラフをバックトラック法を用いて探索することにより，計算資源およびネットワーク資源要求を満足し，かつアプリケーションの実行をデッドラインまでに終了させることが可能な解（各資源の予約の組み合わせ）を発見する．

3.1 予約プラン探索

図 2 は予約プラン探索の概略を示している．本アルゴリズムでは，まず GIS への問い合わせにより，頂点の要求実行環境を満たす計算資源の候補（資源候補）のリストを頂点毎に取得する．次にアプリケーション全体の予約開始時刻 st を一旦固定し，**図 3** のバックトラック関数 `search()` を用いてこの時刻における 解の探索を行う．この探索は無向グラフ上の頂点を訪問する形で行われるため，探索空間を早く削減するために資源候補の少ない頂点が最初の訪問先 rm_{root} に選ばれる．バックトラック探索中に構成された部分解で，st を後方へずらすことで予約が可能となるような資源が選ばれた場合は `search()` による探索を中断し，その時間差分 Δt を st に加えて，再びバックトラック法による解の探索を行う．これをデッドライン $t_{deadline}$ を超過しない解が得られるまで繰り返し行う．ここで st の初期化に用いる図 2 中の $w_{interval}$ はプラン作成に必要な最長時間を保証する時間幅であり，スケジューラ（Planner）によって予め設定される．

`search`(rm, st) は，計算資源要求 rm および rm 周りのネットワーク資源要求群 RP_{rm} を満たす予約候補の決定を行う再帰関数である．`search()` では，rm の資源候補それぞれについて予約可能な時間帯（予約候補）を LIS への問い合わせで取得し，それらを順に rm の仮決定予約候補 FC_{rm} として試していく．予約候補を試す上で，現在の頂点と訪問済みの頂点間のネットワーク空きチェックがまず行われ（但し隣接する頂点に同じ計算資源が割り当てられている場合を除く），未訪問の頂点は `search()` による再帰的な探索にかけられる．再帰呼出しからの返り値が `NOTFOUND` の場合は，指定した頂点以降で部分解が得られなかったことを意味しており，次の予約候補が rm の仮決定予約候補として試される．呼出し先で st の変更を必要とする資源が選ばれた場合はこの関数の返り値が Δt となり，バックトラックは終了する．ここで，rm の予約候補をどの順序で調べていくかは以下の選択方針 (selection policy) に従う．

ランダム 特に何も優先せず，ランダムに選ぶ．

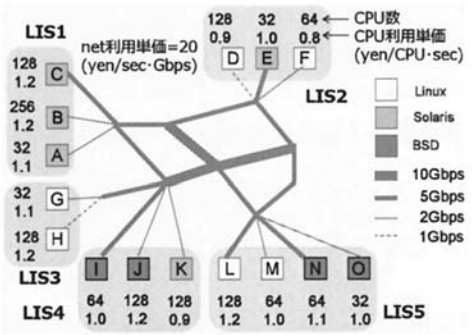

図 4 GridG で生成した狭帯域トポロジ gridg-n

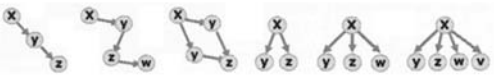

図 5 ワークフロー要求

図 6 コアロケーション要求

負荷優先 負荷が低いものを優先する．この負荷は予約候補取得時に指定した $[t_{bgn}, t_{deadline}]$ の資源利用率であり，LIS の応答に付加される情報である．

費用優先 単位時間あたりの CPU 利用単価が低いものを優先する．

時間優先 予約開始時刻が早いものを優先する．

3.2 予約方法

本アルゴリズムは予約の方法によって BBR と BIR の 2 つに分けられる．**BBR** (Backtracking with Batch Reservation) では，予約プランを作成した後に一括して計算資源およびネットワーク資源への予約を行う．この処理は 2 相コミットで行われる．一方で **BIR** (Backtracking with Incremental Reservation) は 3.1 節で述べたバックトラック探索で計算資源やネットワーク資源の予約候補の決定が行われる度に仮予約を行い，予約プランが出来上がった時点で一斉コミットを行う．ただし BIR の場合，バックトラックや st の更新毎に仮予約をキャンセルする問い合わせが必要である．

4. 実　　験

4.1 実験設定

3 節で述べたアルゴリズムを評価するためにシミュレータを実装した．本シミュレータでは予約プラン要求のシナリオを自動で生成し，離散事象シミュレーションを行うことが出来る．ここでいう事象とはプラン要求の発生や，各サービスの問い合わせ処理等を指す．メモリやハードディスクを予約対象に加えたシミュレーションを行うことも可能であるが，本実験では CPU のみを扱う．

表 1 各資源要求の生成値と生成比率

計算資源要求	OS	Linux:Solaris:BSD = 1:1:1
	CPU 数	SDSC-BLUE のログより
	予約時間幅	600:1200:1800:3600:5400:7200 = 2:2:2:2:1:1 (*1)
ネット要求	バンド幅	0.5:1:2 (Gbps) = 6:3:1
	予約時間幅	(*1) に同じ
締切時刻	-	アプリケーション全体の予約幅 * 95%信頼区間が [3, 7] の正規乱数

表 2 各種問い合わせの処理時間

予約の空きチェック	1 ～ 2 sec
計算資源空き領域探索	2 ～ 3 sec
仮予約	1 ～ 2 sec
予約コミット	1 ～ 2 sec
予約キャンセル	1 ～ 2 sec

本実験では GridG[11] を用いて自動生成したトポロジをもつグリッド環境 gridg-n (**図 4**) 及び gridg-n 上の計算資源に接続するリンクのバンド幅を全て 5Gbps にした広帯域グリッド環境 gridg-w それぞれをシミュレータ上に構築し，**図 5**，**図 6** のワークフロー (wf) またはコアロケーション型アプリケーション (co) の予約プラン要求それぞれを発生させて実験を行う．各計算資源要求では，Parallel Workloads Archive[12] にて公開されている SDSC-BLUE のログを用い，CPU 数が [8, 128] の範囲になるようモデル化を行った．また OS や予約時間幅，バンド幅については**表 1** に従う．ただし，コアロケーション型アプリケーションについては予約プラン要求中の全ての資源要求について予約幅が同一となるようにする．各サービスの問い合わせ処理時間は**表 2** のように設定した．Planner 数は 3 とし，3 節で述べた $w_{interval}$ を 30 分に設定した．ユーザの要求発生はポアソン到着に従って各 Planner にランダムに投入されるが，ユーザは一度受け取った予約プランをキャンセルできないものとする．

予約プラン要求総数に対する作成された予約プラン数の比を予約成功率，予約プラン中の計算資源とネットワーク資源の費用の総和を単位時間当たりの CPU 利用単価に換算した値を平均費用として，乱数の種を変えて 10 回シミューレションを行った結果の相加平均を測定値とした．尚，本稿で示すシミュレーション結果の信頼度 95%の信頼区間は，±4%以内である．最初に gridg-n について，最後に grid-w での結果について述べる．

4.2 BBR と BIR の比較

図 7，**図 9** は gridg-n におけるワークフローとコアロケーション型アプリケーションそれぞれの成功率を示している．グラフ中のアルゴリズム名の語尾 r，c，l，t はそれぞれ選択方針がランダム，費用優先，負荷

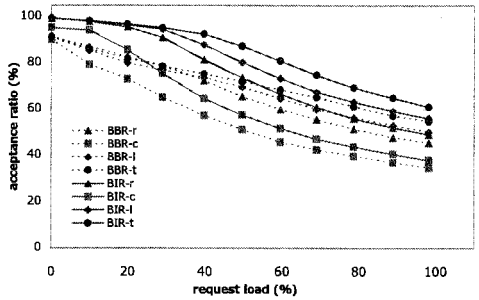

図 7 予約プラン作成成功率（gridg-n + wf）

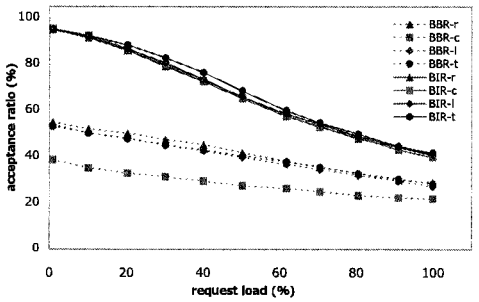

図 9 予約プラン作成成功率（gridg-n + co）

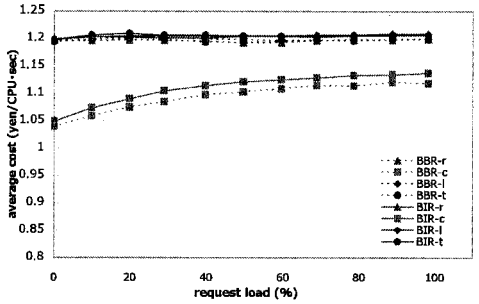

図 8 単位時間あたりの平均 CPU 単価（gridg-n + wf）

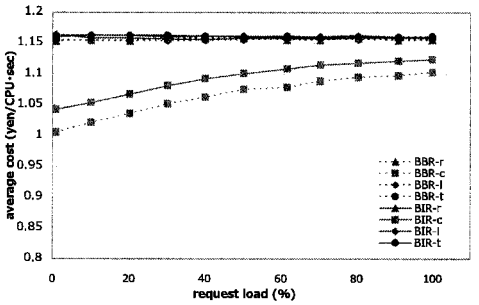

図 10 単位時間あたりの平均 CPU 単価（gridg-n + co）

優先，時間優先であることを示す．まず両アプリケーションにおいて，予約成功率の面で BBR が BIR に劣ることがわかる．BBR の場合は予約プラン作成中に他の Planner に空き領域を予約される可能性が高いことも原因の 1 つであるが，最大の原因は Planner 自身が生み出す予約の競合である．Planner が予約プラン作成中に予約候補を仮決定しても仮予約処理を行わない限りは LIS や NIS にその情報が反映されない．そのため同一プラン内の予約時間が重複する異なる計算資源要求で，同じ資源を割り当てようとした場合に予約の競合が生じる得る．全ての資源に同時刻の予約を必要とするコアロケーション型アプリケーションにおいてこの傾向は顕著に表れている（図 9）．一方で BIR はバックトラック法による探索中に仮予約処理を行うため，そのような競合が起こりにくい．

図 8 は**図 10** は平均費用の結果を示しているが，BBR と BIR 間で特に目立つ差は得られなかった．

4.3 資源選択方針の比較

ワークフローについて，各資源選択方針を BBR と BIR で分けて比較すると予約成功率の面では時間優先のアルゴリズム BBR-t，BIR-t がそれぞれ優れている（図 7）．これは複数の予約候補のうち予約開始時刻が早いものを優先することが，探索時の時間移動を最小限にとどめ，より多くの予約の空き領域を探索することにつながるためである．費用優先のアルゴリズム BBR-c，BIR-c の性能が悪いのは，たとえデッドラインに限りなく近い予約候補であっても費用が低いものであればそれが優先されてしまうためである．探索開始時にそのような予約候補を選ぶことは，作成中のプランがすぐにデッドラインを超過する原因となる．これはランダムや負荷優先にも起こり得る問題であるが，費用優先では資源毎の費用が固定されているために 1 つの資源に予約が集中してしまい，他の選択方針より悪循環を繰り返す原因となっている．一方でコアロケーション型アプリケーションについても時間優先が若干優れているものの，他の方針については殆ど差がないという結果が得られた．平均費用に関しては費用優先（BBR-c, BIR-c）が優れている．平均費用の傾きが右上がりなのは，発生負荷の増加に伴い，低費用の資源の利用率が飽和状態に近づくことで他の資源への予約が増えるためであると考えられる．他の選択方針については殆ど差はない．

4.4 gridg-n，gridg-w における性能比較

gridg-w における各アルゴリズムの予約成功率を**図 11** および**図 12** に示す．BBR・BIR 共に狭帯域の gridg-n に比べて，全体的に予約成功率が上昇しており，ネットワークのボトルネックが解消されてより多くの計算資源の予約が行われている様子が分かる．こ

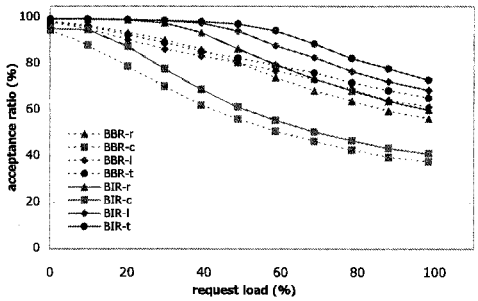

図 11 予約プラン作成成功率 (gridg-w + wf)

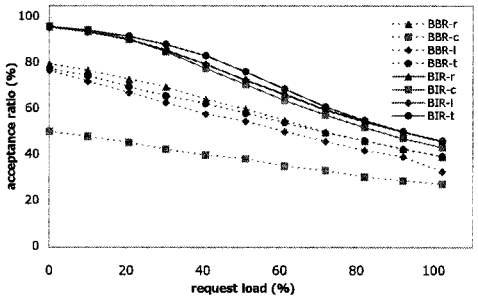

図 12 予約プラン作成成功率 (gridg-w + co)

の点を除けば gridg-n 上で行った実験結果とほぼ同じ傾向の結果が得られた．費用についてもほぼ同様の結果が得られたためグラフは省略した．

5. まとめ

本稿では複数の計算資源とネットワーク資源の事前予約作業を自動化するスケジューリングアルゴリズムの性能評価を行った．本実験結果により，予約プラン作成と予約作業を並行して行い，かつ予約可能時間が早い資源を優先的に割り当てるアルゴリズムが予約成功率について優れていることがわかった．予約プラン作成と予約作業を分けた手法ではスケジューラ自身による予約の競合が著しい性能低下を招いたため，改善の余地が残る．

今後は，ユーザがスケジューリングポリシーを選べる状況におけるユーザの満足度評価や，より大規模なトポロジでのシミュレーションを重ねてアルゴリズムのスケーラビリティの改善を行う予定である．

参考文献

1) PBSPro: `http://www.altair.com/software/pbspro.htm`.
2) Takefusa, A. and et.al.: G-lambda: Coordination of a Grid Scheduler and Lambda Path Service over GMPLS, *Future Generation Computing Systems*, Vol.22, pp.868–875 (2006).
3) 竹房あつ子，中田秀基，　武宮博，松田元彦，工藤知宏，田中良夫，関口智嗣：異種の複数スケジューラで管理される資源を事前同時予約するグリッド高性能計算の実行環境，*HPCS 2007*, pp. 135–142 (2007).
4) Snell, Q., Clement, M., Jackson, D. and Gregory, C.: The Performance Impact of Advance Reservation Meta-Scheduling, *Proc. of 6th Workshop of Job Scheduling Strategies for Parallel Processing* (2000).
5) Min, R. and Maheswaran, M.: Scheduling Co-Reservations with Priorities in Grid Computing Systems, *Proc. of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'02)* (2002).
6) Decker, J. and Schneider, J.: Heuristic Scheduling of Grid Workflows Supporting Co-Allocation and Advance Reservation, *Proc. of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGrid '07)*, pp.335–342 (2007). Rio de Janeiro, Brazil.
7) Zhao, H. and Sakellariou, R.: Advance Reservation Policies for Workflows, *Job Scheduling Strategies for Parallel Processing (Proc. of the 12th Workshop on Job Scheduling Strategies for Parallel Processing, Saint-Malo, France, June 2006)*, Vol.4376/2007, pp.47–67 (2006).
8) Figueira, S., Kaushik, N., Naiksatam, S., Chiappari, S. A. and Bhatnagar, N.: Advanced Reservation of Lightpaths in Optical-Network Based Grids, *Proc. of the ICST/IEEE First Workshop on Networks for Grid Applications (GridNets)* (2004).
9) Naiksatam, S., Figueira, S., Chiappari, S.A. and Bhatnagar, N.: Analyzing the Advance Reservation of Lightpaths in Lambda-Grids, *Proc. of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid2005)* (2005).
10) Kuo, D. and Mckeown, M.: Advance Reservation and Co-Allocation Protocol for Grid Computing, *Proc. of the First International Conference on e-Science and Grid Computing (e-Science'05)*, pp.164–171 (2005).
11) D. Lu, P. D.: GridG: Generating Realistic Computational Grids, *ACM SIGMETRICS Performance Evaluation Review*, Vol.30, No.4 (2003).
12) Logs of Real Parallel Workloads from Production Systems: `http://www.cs.huji.ac.il/labs/parallel/workload/logs.html`.