

# 性能を保証する分散実行環境のための オンラインコアロケーション手法

竹房 あつ子<sup>†</sup> 中 田 秀 基<sup>†</sup>  
工 藤 知 宏<sup>†</sup> 田 中 良 夫<sup>†</sup>

利用者の要求する計算機性能やネットワーク帯域を保証した分散実行環境を構築して提供する場合、それらの資源の同時割り当て（コアロケーション）が重要な課題となる。本稿では、分散する資源が事前予約で複数の組織から提供されることを前提として、オンラインコアロケーション手法を提案する。提案手法では、我々の開発する資源管理フレームワークから利用可能な資源情報を取得し、最適化問題にモデル化したコアロケーションアルゴリズムを用いて複数の予約プランを作成する。また、提案手法では、ユーザや資源管理者の資源の割り当て方針を反映することができる。シミュレーションによる実験を行い、提案するオンラインコアロケーション手法の機能性、実用性に関する評価を行う。機能性の評価では、提案手法の予約成功率の高さと、資源管理者の観点で資源割り当て方針が反映できることを示す。実用性の評価では、最適化問題の求解時間を制約条件とソルバの違いにより比較するとともに、実用化に向けて議論する。

## On-line Resource Co-allocation Algorithm for Performance-guaranteed Distributed Execution Environments

ATSUKO TAKEFUSA,<sup>†</sup> HIDEMOTO NAKADA,<sup>†</sup> TOMOHIRO KUDOH<sup>†</sup>  
and YOSHIO TANAKA<sup>†</sup>

Resource scheduling is one of the key technologies to provide computing and network resources to the users, while meeting the QoS with the users. We propose an advance reservation-based on-line co-allocation algorithm for both computing and network resources on distributed execution environments. In the proposed algorithm, we solve the co-allocation problem as a generic optimization problem. The goal of our algorithm is to create reservation plans satisfying user resource requirements. Also the algorithm takes co-allocation options in user and resource administrator views into consideration. We evaluate the proposed algorithm in our advance reservation-based co-allocation model with extensive simulation. Our experimental results show that our algorithm can take co-allocation options and seems practical as an on-line method.

### 1. はじめに

グリッドとネットワーク資源管理技術により、利用者の要求する性能を保証する計算機、ネットワーク、ストレージなどの資源を動的に組合わせて“分散実行環境”を構築することが可能になった。分散実行環境を提供する実証実験は、国内外における産学官プロジェクトとして複数行われている<sup>1)~5)</sup>。これらの試みでは、帯域が保証されたパスネットワークも占有可能な資源の一つとして扱い、他の資源と同時に提供する点で従来のグリッドとは異なる。

分散実行環境では、ネットワークも資源の一つとし

て扱うため、各資源は別々の組織またはVO（仮想組織）により管理・提供される。また、各ユーザに対して要求された性能を保証するため、事前予約を前提とする場合が多い。計算資源のみを対象としたマルチクラスタの同時確保では、KOALA<sup>6)</sup>グリッドスケジューラやQBETS<sup>7)</sup>バッチキュー予測サービスのように、各ローカルスケジューラのキューの利用状況の取得やその予測により、事前予約機能がない状況での同時確保を実現している。しかしながら、これらの手法は多くの資源の利用をブロックしたり、複数資源が同時に利用可能になることを保証するものではない。よって、複数資源の同時確保を実現するには事前予約が有効な手段と言える。

分散実行環境の構築では、メタスケジューラ（またはスーパースケジューラ）により多様な資源群からユー

<sup>†</sup> 産業技術総合研究所 National Institute of Advanced Industrial Science and Technology (AIST)

ザの要求を満たす適切な資源群を選択し、確保すること、すなわちコアロケーションが重要な課題となる。特に、分散実行環境では計算資源だけでなく、計算資源をつなぐネットワークの帯域も確保するため、コアロケーションでは計算機とネットワークのトポロジの両方を考慮しなければならない。よって、従来の計算機のみを対象とした様々なヒューリスティック手法<sup>8),9)</sup>や、ネットワークのルーティングなどの最短経路問題で用いられるダイクストラ (Dijkstra) 法に基づく手法<sup>10)</sup>等をそのまま利用することはできない。バックトラック法を用いて計算機とネットワーク両方の資源を確保していく手法<sup>11)</sup>も提案されているが、資源確保に要する時間が長くなり、手続きも煩雑になるなどの問題もある。

また、ユーザの性能要求を満たす資源群を確保するだけでなく、ユーザや資源管理者の資源の割り当て方針を反映させる必要がある。すなわち、ユーザの観点では指定した性能を保証する資源群を (a) 早い時刻に確保する、(b) 価格の安い資源群を確保する、(c) 価格が高くても品質 (可用性等) 優先で確保する、等の選択肢がありうる。一方、資源管理の観点では、上記を考慮することに加え、(A) 複数資源提供者に対して平等に負荷を割り当てる、(B) 省エネや連携関係から、特定の資源に優先的に割り当てる、(C) ユーザのサービスレベルにより、高レベルのユーザが確実に資源確保できるように配慮する、等の方針が考えられる。

さらに、バッチキューイングシステムと異なり、オンライン処理ではプランニングに要する時間をできるだけ短くすることが求められる。一方、本研究で扱うような資源のスケジューリング問題は NP 困難であるため、最適化問題として解くと、実在する資源サイトの数や要求する資源の数の増加によって求解時間が指数的に長くなってしまふ。

本研究では、資源が複数の組織から提供されることを前提とし、計算・ネットワーク資源を同時に確保するための、オンラインコアロケーション手法を提案する<sup>12),13)</sup>。提案手法では、ユーザの計算機及びネットワークに関する要求に対して、各資源を管理する資源マネージャからある時間帯における動的資源情報を入力し、その情報から複数予約プランを作成する。各時間帯における予約プランの作成は、最適化問題にモデル化して行う。また、本研究では提案手法が機能性、実用性の面で有効であることを、シミュレーションによる評価実験で示す。

機能性の評価では、提案手法により計算・ネットワーク資源のコアロケーションが可能であることを示すとともに、資源管理者の方針 (A)(B)(C) が、提案手法により反映できることを示す。また、ユーザの方針 (a)(b)(c) については、本稿で反映する方法を示す。

実用性の評価では、予約プランを作成する際に要する時間を比較し、提案手法がオンラインコアロー

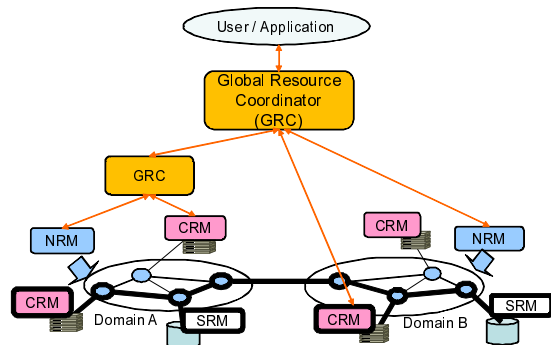


図1 資源管理フレームワークの概要

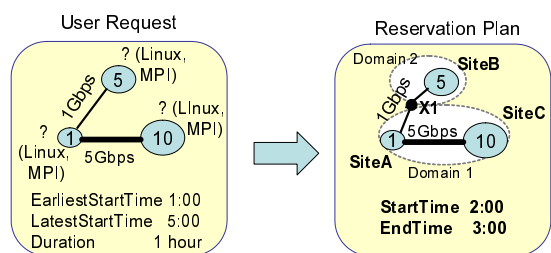


図2 ユーザの資源要求とプランナによる予約プラン

ション手法として実用的であることを示す。実験では、フリーの線形計画ソルバ GLPK (GNU Linear Programming Kit)<sup>14)</sup>と、充足可能性問題 (SAT) のソルバ MiniSat<sup>15)</sup>と Sugar++<sup>16)</sup>を組み合わせたものを用いて求解時間を比較し、提案手法がオンラインサービスとして実用的であることを示す。

## 2. コアロケーションモデル

### 2.1 コアロケーションシステムモデル

我々は、ネットワーク資源を含む多様な資源の管理フレームワーク GridARS の開発をしている<sup>17)</sup>。本フレームワークは資源管理システム (RMS)、資源レジストリ (RR)<sup>18)</sup>、資源モニタリングシステム (MS)<sup>19)</sup>で構成される。RR は各 RM の所在や各資源の資源マネージャ (RM) の管理する資源の静的な情報を提供し、MS は予約資源の稼働時の状況を提供する。

RMS の概要を図 1 に示す。RMS は、グローバル資源コーディネータ (GRC) と各資源を管理する資源マネージャ (RM) で構成され、GRC と RM が連携してユーザに資源群を提供する。図 1 の NRM, CRM, SRM は、それぞれネットワーク、計算機、ストレージの RM を表す。GRC のうち、スケジューリング機能 (プランナ) をもつものがメタスケジューラに相当し、資源の予約プランを決定する。

### 2.2 ユーザの資源要求

図 2 にユーザの資源要求 (左) と、その要求からプランナが生成する予約プラン (右) の例を示す。計算

資源の要求では、計算サイト数、各サイトでの CPU (コア) 数、属性情報 (OS, 実行環境等) を、ネットワーク資源要求では要求する拠点間の帯域、遅延、その他の属性情報 (メディアタイプ, 可用性等) を指定する。また、各資源に対して時刻の要求を指定することができる。時刻は、全資源の確保時間帯を直接、または最も早い開始時刻  $EST$ , 最も遅い開始時刻  $LST$ , 予約時間幅  $D$  により指定する。図 2 右はプランナが選定した予約プラン例で、計算資源サイト SiteA, SiteB, SiteC の計算機とその間の通信経路、資源の確保開始・終了時刻が決定している。通信経路は、抽象化されたトポロジで表われ、複数ドメインを跨る経路は複数のパスで表される。図 2 では、SiteA-SiteB 間の経路は  $X1$  を経由して Domain1 と Domain2 の提供する 2 つのパスで構成される。

### 3. コアロケーション手法

#### 3.1 コアロケーション手法の概要

本研究で提案するコアロケーション手法の概要を述べる。GRC における資源確保手順は以下の通りである。

- (1) 静的資源情報をあらかじめ RR から取得する。
- (2) ユーザの資源要求を受け取る。
- (3) GRC のプランナで資源要求に対する予約プランを複数作成する。
  - (3a) 指定された時間帯から予約時間帯候補を  $N$  個選ぶ。
  - (3b) 関連する RM から、 $N$  個の時間帯において利用可能な資源情報を取得する。
  - (3c) 取得した情報から、提案コアロケーションアルゴリズムを用いて  $n$  ( $\leq N$ ) 個の予約プランを作成する。
  - (3d)  $n$  個の予約プランの優先順位を決定する。
- (4) プランナの作成した予約プランから、GRC と複数 RM が連携して資源を確保する。
- (5) 確保が成功した場合は終了、失敗した場合はその情報をユーザに提供する。

資源を確保する時間帯が直接指定されている場合は指定時刻から資源を探索し、時間幅で指定される場合は  $[EST, LST + D]$  から資源を探索する。

手順 (3a) では、予約時間帯候補数  $N$  を大きく設定するほどコアロケーションの成功率が上がり、より適切な資源群の選択ができるが、計算パワーとのトレードオフとなる。

手順 (3b) の資源情報の取得では、資源予約インタフェース GNS-WSI<sup>1)</sup> を用いる。GNS-WSI は、商用サービスを前提として G-lambda プロジェクト<sup>20)</sup> が規定している。GNS-WSI では、各 RM の詳細な予約テーブルを公開しない代わりに、指定した時刻における利用可能な資源情報を提供するため、本手法では手

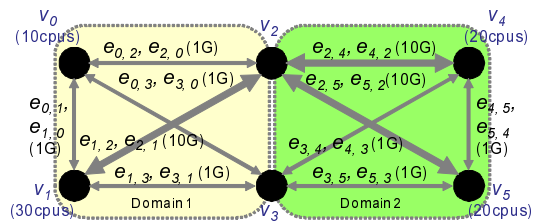


図 3 資源グラフ

順 (3a) のように予約時間帯候補を決めて問い合わせる。ここで、GNS-WSI では複数条件の問い合わせを同時に行うことが可能であり、また我々の資源管理フレームワーク GridARS では複数 RM に対する問い合わせを同時に行うことができるため、これによるオーバーヘッドは  $O(1)$  となる。

手順 (3c) では、選んだ各予約時間帯に対して次節のコアロケーションアルゴリズムを適用する。ここで、 $N$  個の予約時間帯候補に対して独立に予約プランを決定することができる。

手順 (3d) の  $n$  個の予約プランの優先順位は、3.4 節資源の割り当て方針に従って決定する。

手順 (4) における資源確保手続きは、GridARS を用いると複数 RM に対して並行して資源確保の手続きが行えるため、このオーバーヘッドも  $O(1)$  となる。

#### 3.2 コアロケーションアルゴリズム

提案するコアロケーションアルゴリズムでは、提供可能な資源群と資源要求をグラフで表し、線形計画法 (LP) を用いて予約プランを作成する。LP は、1 次の不等式または等式のみで目的関数と制約条件が定義された最適化問題のことを呼ぶ。

提案手法では、図 3 に示すような双方向の有向グラフ  $G = (V, E)$  として資源群を表す。 $V$  は  $G$  の点の集合、 $E$  は  $G$  の枝の集合であり、グラフの各点  $v_i$  は計算資源サイトまたはネットワークドメイン間の交換点を、各枝  $e_{+r}$  ( $= e_{o,p}$ ) または  $e_{-r}$  ( $= e_{p,o}$ ) は各 NRM が提供する資源間のパスを表す。グラフの点および枝の括弧内は、提供可能な CPU 数および帯域であり、それぞれ  $wc_i$  ( $i \in V$ ),  $wb_k$  ( $k \in E$ ) と表す。図 3 の  $v_2, v_3$  はドメイン間交換点であり、利用可能な CPU は存在しない。また、提供する CPU および帯域の単位あたりの重み (価格など) を、それぞれ  $vc_i$  ( $i \in V$ ),  $vb_k$  ( $k \in E$ ) と表す。 $wb_k$  および  $vb_k$  は、 $e_{+r}, e_{-r}$  で共有する。

次に、資源要求を完全グラフ  $G_r = (V_r, E_r)$  で表す。この際、各枝は有向グラフで表されるが、2 点間の枝は 1 つのみで、その方向は任意に決めることができる。 $V_r$  は必要とする計算資源サイト (点) の集合、 $E_r$  は  $V_r$  間を結ぶネットワーク (枝) の集合を示す。また、 $G_r$  に必要な CPU 数および帯域をそれぞれ  $rc_j$  ( $j \in V_r$ ),  $rb_l$  ( $l \in E_r$ ) とする。

以上のパラメータから、以下の変数を線形計画法で求めることで予約プランを決定する。

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

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

ここで、資源要求されるサイトに対して選択される計算資源サイトの要素  $x_{i,j}$  は 1, その他は 0 となる。また、資源要求されるネットワークに対して選択されるパスの要素  $y_{k,l}$  は 1, その他は 0 となる。

次に、目的関数と制約条件は以下のように表せる。

Minimize

$$\sum_{i \in V, j \in V_r} vc_i \cdot rc_j \cdot x_{i,j} \\ + \sum_{k \in E, l \in E_r} ve_k \cdot rb_l \cdot y_{k,l} \quad (3)$$

Subject to

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

$$\forall i \in V, \sum_{j \in V_r} x_{i,j} \leq 1 \quad (5)$$

$$\forall i \in V, \sum_{j \in V_r} rc_j \cdot x_{i,j} \leq wc_i \quad (6)$$

$$\forall l \in E_r, \sum_{k \in E} y_{k,l} \begin{cases} \geq 1 & (rb_l \neq 0) \\ = 0 & (rb_l = 0) \end{cases} \quad (7)$$

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

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

式 (3) の目的関数は、選択する計算・ネットワーク資源の重みの合計を最小化することを表している。

式 (4), (5), (6) は計算資源に関する制約を表す。式 (4) は、要求サイト  $j$  に割り当てられる実際のサイトは 1 つのみであること、式 (5) は、各資源サイト  $i$  は資源要求の中で 1 つ以上割り当てられないことを表す。式 (6) は、各資源サイト  $i$  が要求される CPU 数以上の CPU を提供可能であることを示している。

式 (7), (8) は、ネットワークの帯域に関する制約を表す。式 (7) は、要求サイト間のパス  $l$  の帯域の確保要求がある場合は、 $y_{k,l}$  の総和が 1 以上に、ない場合は 0 になることを表す。式 (8) は、各パス  $k$  が要求される帯域以上の帯域を提供可能であることを示す。

式 (9) は、流量保存則<sup>21)</sup>を応用して  $x_{i,j}$  と  $y_{k,l}$  を関連付ける制約を表す。流量保存則では、流量を  $f$  とすると、始点の供給量は  $f$ 、終点の供給量は  $-f$ 、通過点の供給量は 0 となる。よって、流量  $f = 1$  と想

定して適用すると、右辺の値は 1, -1, または 0 となる。ここで、 $m$  が  $l$  の始点の時は  $x_{m,o} = 1$ 、終点の時は  $x_{m,p} = 1$ 、それ以外は  $x_{m,o} = x_{m,p} = 0$  となるので、式 (9) の右辺を  $x_{m,o} - x_{m,p}$  として表すことができる。

### 3.3 コアロケーションアルゴリズムの高速化

最適化問題では、求める変数の数が増えると求解時間が指数的に増大する問題がある。よって、3.2 節に以下の 2 つの制約、すなわち枝刈り条件を加えることで、求解時間を短くすることを試みる。

Subject to

$$\forall l \in E_r, \forall m, n \in V (m \neq n), \\ y_{(m,n),l} + y_{(n,m),l} \leq 1 \quad (10)$$

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

式 (10) は、求めるネットワーク (式 (9) におけるフロー) に対して、有向グラフの双方向の枝を同時に選択しないことを表す。これにより、無駄な経路の探索を防ぐことができる。式 (11) は、1 つのネットワークに対して構成する枝の数の最大値  $N_{max}$  を指定するものであり、 $N_{max}$  は経験的に与えるパラメータとなる。よって、式 (11) は最適解を求めるわけではないが、探索範囲が大幅に削減できるため、求解時間の短縮には有効であると考えられる。

### 3.4 資源の割り当て方針の反映

提案手法では、ユーザと資源管理者それぞれの観点で、資源の割り当て方針を反映させることができる。ユーザの観点では、指定した性能を保証する資源群を (a) 早い時刻に確保する、(b) 価格の安い資源群を確保する、(c) 価格が高くても品質 (可用性等) 優先で確保する、等の選択肢がありうる。一方、資源管理者の観点では、(A) 複数資源提供者に対して平等に負荷を割り当てる、(B) 省エネや連携関係から、特定の資源に優先的に割り当てる、(C) ユーザのサービスレベルにより、高レベルのユーザを優先して資源を確保するように配慮する、等の方針が考えられる。

これらの方針を反映させるため、3.1 節の手順、3.2 節の目的関数に対して次のように修正する。

(a) 時刻優先 (3c) で  $EST$  に近い順に  $n$  個の予約プランをソートする。

(b) 価格優先 CPU および帯域の重みを価格にし、式 (3) の目的関数の値が最小となる資源群を探索する。

(c) 品質優先 CPU および帯域の重みを品質の高さを表すパラメータで表し、式 (3) を修正して目的関数の値が最大化する資源群を探索する。

(A) 平等な負荷分配 提案手法のとおりに行う。

(B) 特定資源の優先 資源管理者の方針に従って資源の重み付けをし、式 (3) で目的関数の値が最小となる資源群を探索する。

(C) サービスレベルの考慮 (3b) の動的資源情報取得時に、各ユーザのサービスレベルに即した情報に

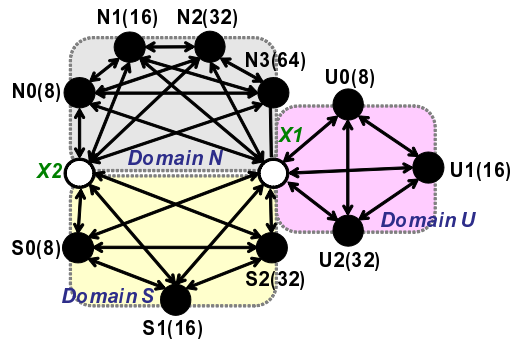


図 4 シミュレーション環境

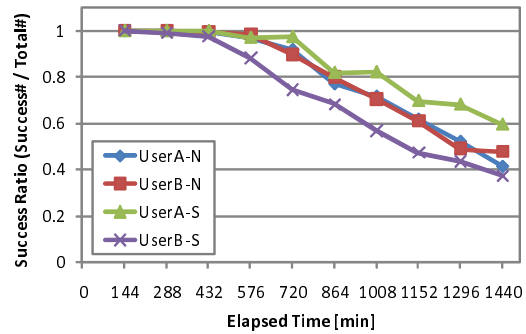


図 6 資源確保成功率の比較

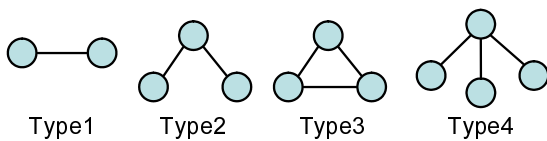


図 5 資源要求の種類

表 1 シミュレーション設定

環境設定	
資源構成	GRC 数=1, NRM 数=3, CRM 数=10
サイト数/ドメイン名	4/N, 3/S, 3/U
ドメイン交換点	X1 {N, S, U}, X2 {N, S}
CPU 数	N{8, 16, 32, 64}, S{8, 16, 32}, U{8, 16, 32}
CPU 単価	1
帯域 [Gbps]	ドメイン内 = 5, 交換点接続 = 10
帯域単価	ドメイン内 = 5, 交換点接続 = 3
資源要求設定	
ユーザ	UserA, UserB
資源要求の種類	Type1,2,3,4 (図 5)(一様分布)
要求 CPU 数	各 type の全サイトに対し, 1, 2, 4, 8 (一様分布)
要求帯域 [Gbps]	各 type の全ネットワークに対し, 1 (固定)
各ユーザの要求到着間隔	ポアソン到着
予約時間幅 [min]	30, 60, 120 (一様分布)
LST - EST(探索時間幅)	平均要求到着間隔 × 3

修正する。

#### 4. 評価

##### 4.1 シミュレーションモデル

評価では、機能性と実用性において提案手法が有効であることを示す。機能性の評価では、提案手法により計算・ネットワーク資源のコアロケーションが可能であることを示すとともに、資源管理者の観点で資源割り

当て方針が反映できることを示す。実用性の評価では、式 (10)、式 (11) の制約条件とソルバの違いによる求解時間の比較を行い、提案手法が実用的であることを示す。

図 4 に各実験で用いるシミュレーション環境を示す。これは、実際の実証実験で用いた計算資源のサイトとネットワークのテストベッド<sup>2)</sup>を想定し、抽象化したものである。図 4 に示すように、ドメイン内のサイト(黒丸)間は完全グラフで接続され、ドメイン内の各交換点(白丸)とドメイン内の全サイトもそれぞれ接続されている。

シミュレーション設定の概要を表 1 に示す。シミュレーションでは、2人のユーザ UserA と UserB が 1つの GRC に対して資源予約要求を繰り返し行う。資源要求の種類は、図 5 のいずれかのネットワークと計算機群の組合わせを同時に確保するものである。各ユーザにおける資源要求設定は両方とも表 1 に従う。最初の 24 時間までに各ユーザの資源要求が到着し、それぞれ次の 24 時間の時間帯の中から資源の予約を要求をする。2人のユーザの要求の平均到着間隔は、24 時間の間に計算資源に関して 100% の要求率になるように設定した。また、GRC での予約時間帯候補数  $N$  は 10 とし、時間帯候補は  $LST - EST$  の間で等間隔に選定した。候補間での優先順位は時間優先とした。

実験では、サイトの CPU 数は最大 64、ユーザが要求する CPU 数は 1~8 と比較的小さい環境を想定した。これは、提案手法に要する計算量はサイトの数に依存し、サイトの CPU 数およびユーザの要求する CPU 数には依存しないためである。実際の環境に応じてこれらのパラメータを変更しても、同様の傾向が得られると考えられる。

##### 4.2 機能性の評価

機能性の評価では、まず、予約成功率と資源管理者の要求 (C) に関する評価を行う。

図 6 に、UserA と UserB の資源要求に対するコアロケーションの成功率を示す。横軸は資源要求受付時刻であり、1440[min] のときに資源の要求率が 100% であることを示す。縦軸はコアロケーション成功率を

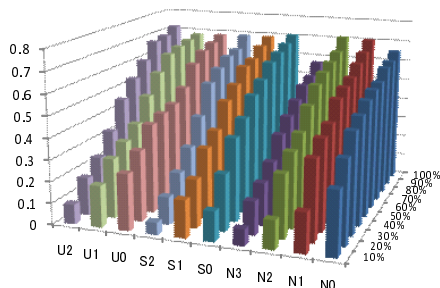


図 7 サイトの負荷の比較 ((A) . 重み付けなし) .

示しており、グラフの各点は 10 回のシミュレーションにおいて各時間帯の資源要求に対して資源が確保できた割合を表す。“UserA-”で表されているのは UserA の結果，“UserB-”は UserB の結果であり，“S” / “N” はサービスレベル (SL) の設定の有 / 無を示す。SL 設定有りの場合は，UserB が SL が低い場合を想定し，各 RM の管理する利用可能な資源のうち，UserB は半分までしか確保できないようにした。

図 6 から，コアロケーションの成功率を SL 無しの結果と比較すると，UserA, B の予約成功率は要求率が 50[%](720[min]) のときに 0.918 と 0.897 と，非常に高い結果が得られた。一方，要求率が 100[%](1440[min]) になると遊休資源がなくなるために成功率が下がるものの，0.415 と 0.478 の割合でコアロケーションが成功していた。よって，提案手法を用いることで，計算・ネットワーク資源のコアロケーションが可能であることが示された。

資源管理者の要求 (C) の SL の有無に関する比較では，SL 無しの場合は UserA と UserB の成功率はほぼ同程度であるのに対し，SL 有りの場合は要求率が低い場合に差はないものの，要求率が 100[%] になると UserA の成功率は 0.595，UserB の成功率は 0.415 となった。よって，SL の設定により低 SL のユーザの成功率を下げることが可能であり，特に負荷が高い状況では高 SL のユーザの成功率を上げることが可能であることが示された。

次に，資源管理者の要求 (A), (B) に関する評価を行う。(B) では，CPU の単価に対して重み付けすることにより，特定の資源を優先するようにした。

図 7 に (A) の方針，図 8，図 9 に (B) の方針でシミュレートした際の各サイトの平均負荷を示す。図 8 は計算サイトの所有する CPU 数ごとに 64 : 32 : 16 : 8 = 1 : 10 : 100 : 1000 のように重み付けし，大きいサイトに優先的に割り当てられるようにした。図 9 は，ネットワークドメインごとに N : S : U = 1 : 10 : 100 のように重み付けし，N, S, U の順に，優先的に割り当てられるようにした。

シミュレーション結果から，図 7 はすべてのサイ

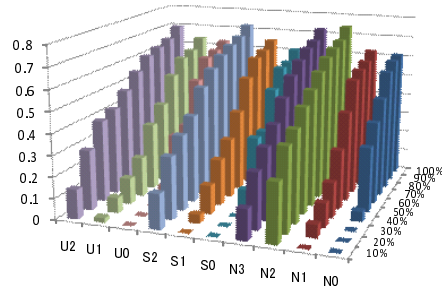


図 8 サイトの負荷の比較 ((B) . サイト CPU 数ごとに重み付け) .

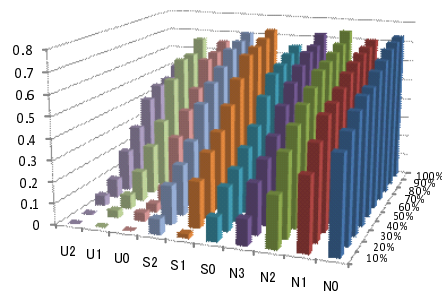


図 9 サイトの負荷の比較 ((B) . ネットワークドメインごとに重み付け) .

トにおいてほぼ均一に負荷が上がっているのに対し，図 8 では CPU 数の多いサイトに，図 9 では，N, S, U の順に資源が優先的に割り当てられていることが分かる。一方，シミュレーション時間が長くなると遊休資源が少なくなるためにいずれの結果も負荷が均一になっていく。よって，我々の手法に重み付けをすることにより，遊休資源がある状況では資源管理者の要求を反映できることが分かった。

### 4.3 実用性の評価

#### 4.3.1 ソルバの比較

実用性の評価では，制約条件による比較と，求解に用いるソルバの比較を行う。ソルバの比較では，線形計画法 (LP) のソルバである GLPK (GNU Linear Programming Kit)<sup>14)</sup> と，充足可能性問題 (SAT) のソルバ MiniSat<sup>15)</sup> と Sugar++<sup>16)</sup> を組み合わせたものを用いた。いずれもオープンソースソフトウェアである。

GLPK では，LP を解くアルゴリズムとして単体法 (simplex method) と内点法 (interior point method) を用いており，デフォルトでは単体法が適用される。本実験においても，単体法を用いている。

SAT のソルバは，乗法標準形 (CNF) で与えられた全変数に対して，すべての制約条件が満たされれば充足可能 (SAT)，満たされていない場合は充足不可能 (UNSAT) と判定するものである。我々の手法は最適化

手法	平均値 [sec]	最大値 [sec]	標準偏差
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

問題としてモデル化されているため, SAT のソルバをそのまま利用することはできない. よって, 本研究では Sugar++ を用いた. Sugar++ は Sugar<sup>22)</sup> を改良したものである. Sugar は, 制約充足問題 (CSP) を CNP に変換するためのソフトウェアであり<sup>23)</sup>, Sugar++ は CSP を最適化問題として解くために改良されたものである. Sugar++ では, 最適解を求めるために最小化または最大化する目的関数の上限, 下限を一時的に固定して CSP として解き, 複数回 CSP を解くことにより最適解を求めることができる.

#### 4.3.2 求解時間の評価結果

求解時間の評価では, 以下の 4 通りの手法を比較した.

GLPK 提案手法に対して GLPK を用い, 最適解を求める.

GLPK-st 提案手法に 3.3 節の制約 (式 (10), 式 (11)) を追加し (改良手法とする), GLPK を用い, 最適解を求める.

MiniSat-st 改良手法に対して, Sugar++ と MiniSat を用い, 最適解を求める.

MiniSat-st-1 改良手法に対して, Sugar++ と MiniSat を用い, 制約条件を満たす解の 1 つを求める (SAT の計算回数は 1 回).

求められる解の質は,  $GLPK \geq GLPK-st = MiniSat-st > MiniSat-st-1$  となる. シミュレーションは, いずれも CPU Intel Core2 Quad Q9550(2.83GHz), OS CentOS 5.0, kernel 2.6.18 x86\_64, メモリ 4GB の計算機上で実行した. 各ソルバのバージョンは GLPK v. 4.37, MiniSat v. 2.0, Sugar v. 1.14 であり, Sugar には Sugar++ 用の未公開パッチをあてた.

表 2 に, 各手法における求解時間の平均値, 最大値と標準偏差を示す. GLPK と GLPK-st で比較すると, 平均値, 最大値のいずれも GLPK-st の方が 2 倍以上短くなっていることが分かる. このことから, LP に制約条件を加えることで求解時間を短縮する手法は, 非常に有効であることが分かる.

次に, 同じ制約条件を使った GLPK-st と MiniSat-st を比較すると, GLPK-st の方が圧倒的に速い結果となった. このことから, 本研究のコアロケーションの問題の最適化問題のソルバとしては, GLPK の方が適していることが分かる. 一方, 求解時間の最大値および標準偏差ですべての手法を比較すると, MiniSat-st-1 が一番小さいという結果が得られた.

図 10, 図 11 に各資源要求における求解時間の結果を示す. 横軸は要求された資源要求の順序を表し,

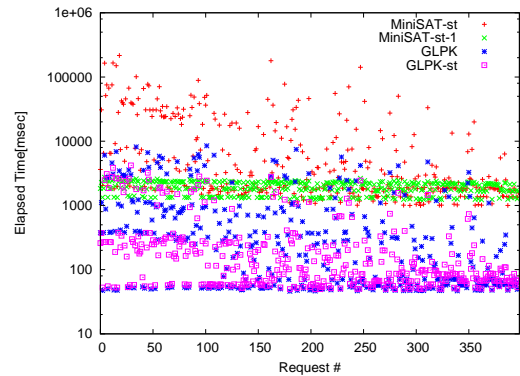


図 10 各資源要求における求解時間の比較 (ログスケール).

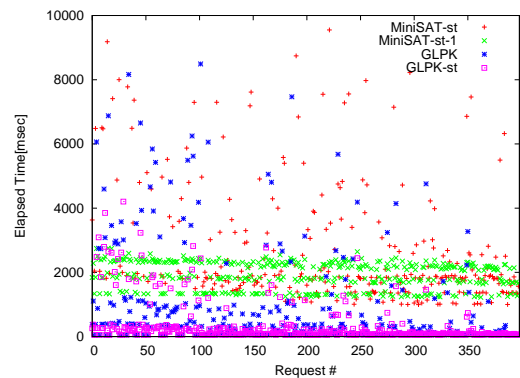


図 11 各資源要求における求解時間の比較 (0~10[sec] まで結果).

各プロットは各要求に対して予約候補数  $N = 10$  個の計算時間の平均値をプロットしたものである. この  $N$  個の計算は独立して計算できるため, 実環境で利用する場合において実際にかかる時間を想定した.

図 10 は  $y$  軸をログスケールで表したものであり, 図 11 はそのうちの 0-10[sec] までの結果を示したものである. グラフから, 最適解を求める手法では, いずれも問題によって求解時間に大きなばらつきがあるのに対し, MiniSat-st-1 の結果ではばらつきはみられない. また, 図 11 では, MiniSat-st-1 の結果において 3 本の筋が確認できる. これは図 5 の要求計算サイト (点) の数の違いによるものである. すなわち, 最適解を求めない手法においては, 多項式時間で求解できることが分かる.

また, 図 10 において最適解を求める手法の結果は, いずれも要求数が増えるほどばらつきが減っていくことが分かる. これは利用可能な資源の数が少なくなっていくにつれて, 最適解の探索範囲が狭まるためである.

#### 4.3.3 議論

従来のスケジューリングに関する研究では, NP 困難であるために様々な複雑かつヒューリスティックな手法が提案されてきた. 一方で, 近年の計算機性能の向

上と LP ソルバの研究成果により、LP の実求解時間は短縮されている。また、LP は制約条件を加えていくほど求解時間が短くなるという性質を持つ。CPLEX<sup>24)</sup>などの商用ソルバでは、前処理で制約条件を加えることにより、求解時間の短縮を図っているといわれている。さらに、最適解を求めなければ多項式時間で求解できるため、その点でも求解時間の性能改善の余地がある。

提案手法では、3.2 節のコアロケーションアルゴリズムにおいてネットワークを完全グラフで表現しなければならないため、サイト数の 3 乗のオーダーで変数の数が増加してしまう。しかしながら、以下のような性質もある。

- 図 1 で示したように、プランニングするスケジューラ (GRC) が階層的に構成されており、探索範囲の局所化が可能である。
- モデル化したグラフの点の数が、計算機の台数ではなくサイトの数でスケールする。
- ネットワークの遅延や計算の実行環境など、今後様々な要求を受け入れることを前提としており、さらに制約条件を加えることが可能である。
- バックトラック法等のヒューリスティック手法では、遊休資源が少なくなると探索時間が長くなる傾向があるのに対し、LP では遊休資源が少なくなるほど探索時間が短くなる。

よって、本研究のスケジューリング問題に対して、LP を適用することは有効なアプローチであると考えられる。

さらに、提案コアロケーション手法は、利用可能な資源群の情報とユーザが必要とする資源群の情報が得られれば、事前予約を前提としない環境においても適用可能である。

## 5. 関連研究

事前予約を前提とした広域分散環境の計算・ネットワーク資源のコアロケーション手法がいくつか提案されている。

Roblitz は、我々同様最適化問題に基づいた手法を提案している<sup>25)</sup>。この研究では、ネットワークは単純なモデルでトポロジは想定していない、メタスケジューラが各 RM での資源の空き時間を把握できるのに対し、我々の手法では商用サービスを想定し、各 RM で管理されている資源の予約情報の一部を利用してプランニングを行っている点で異なる。また、ソルバに CPLEX を用いており、求解時間の短縮に関する考察は行っていない。

安藤と合田の提案する手法は、資源群を無向グラフで表し、バックトラック法により資源群を探索する<sup>11)</sup>。この研究では、バックトラック法でインクリメンタルに予約手続きを行う手法で高い予約成功率を示してい

るが、手続きが煩雑になる、全資源の確保に要する時間が長くなり多くの資源をブロックしてしまうという問題もある。

Elmroth と Tordsson は、NordGrid<sup>5)</sup> を前提としたコアロケーション手法を提案している<sup>26)</sup>。Elmroth らの手法では、予約時間帯をずらしながら、まず計算資源を探索し、次に決定した計算資源サイト間のネットワークを探していく。我々の資源管理フレームワーク GridARS の GRC における初期実装も同様のアプローチをとっていたが、このような手法では後から探索する資源の制約が強い場合 (ネットワークの帯域が小さいなど) には予約プランの作成に時間がかかる。

上記 2 つの手法では、最初に見つかった資源群を選択するため、冗長なパスが選択されるなど、必ずしも適切な予約プランが得られないという問題がある。また、いずれの手法においても、資源管理者の資源の割り当て方針を反映させることはできない。

## 6. まとめ

本研究では、資源が複数の組織から提供されることを前提とし、計算・ネットワーク資源を同時に確保するための、オンラインコアロケーション手法を提案した。提案手法では、LP にモデル化して資源を選択し、ユーザや資源管理者の資源の割り当て方針を反映させることができる。

機能性の評価では、提案手法により計算機とネットワークのコアロケーションが可能であり、資源管理者の観点での資源の割り当て方針が反映できることを示した。実用性の評価では、現在の想定環境でも実用レベルであることを示した。また、将来的にサイト数がスケールする状況においても、LP の求解時間の短縮の余地があり、我々の手法が実用的であることが示唆された。

今後は、提案手法に実環境における様々な資源条件を反映させ、スケーラビリティに関して評価するとともに、現実的な経済モデルや SLA モデルを調査し、ユーザの割り当て方針の適用に関する評価を行う。また、我々の開発する資源管理システム GridARS への組み込みを行い、提案手法を実用化する。

謝辞 Sugar を利用するにあたり、貴重なご意見を下さった神戸大学の田村直之先生、Sugar++ で MiniSat を用いて最適化問題を解くために必要な未公開パッチを提供して下さった丹生智也様に感謝いたします。また、最適化問題に関してご助言いただいた中央大学の藤澤克樹先生、安井雄一郎様に感謝いたします。

本研究の一部は、科研費 (21700047) および情報通信研究機構 (NICT) の委託研究「ダイナミックネットワーク技術の研究開発」の助成を受けたものである。

## 参考文献

- 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. and Sekiguchi, S.: G-lambda: Coordination of a Grid Scheduler and Lambda Path Service over GMPLS, *Future Generation Computing Systems*, Vol. 22(2006), pp. 868–875 (2006).
- 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. and Jinno, M.: G-lambda and EnLIGHTened: Wrapped In Middleware Co-allocating Compute and Network Resources Across Japan and the US, *Proc. GridNets2007* (2007).
- 3) Barz, C., Pilz, M., Eickermann, T., Kirtchakova, L., Waldrich, O. and Ziegler, W.: Co-Allocation of Compute and Network Resources in the VIOLA Testbed, TR 0051, CoreGrid (2006).
- 4) 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>.
- 5) NorduGrid: <http://www.nordugrid.org/>.
- 6) Mohamed, H. and Epema, D.: Experiences with the KOALA Co-Allocating Scheduler in Multiclusters, *Proc. 5th IEEE/ACM Int'l Symp. on Cluster Computing and the GRID (CCGrid2005)* (2005).
- 7) Nurmi, D., Brevik, J. and Wolski, R.: QBETS: Queue Bounds Estimation from Time Series, *Proc. 13th Workshop on Job Scheduling Strategies for Parallel Processing* (2007).
- 8) Casanova, H., Zagorodnov, D., Berman, F. and Legrand, A.: Heuristics for Scheduling Parameter Sweep Applications in Grid Environments, *Proc. the 9th Heterogeneous Computing Workshop*, pp. 349–363 (2000).
- 9) Castillo, C., Rouskas, G. N. and Harfoush, K.: Resource Co-Allocation for Large-Scale Distributed Environments, *Proc. HPDC2009*, pp. 137–150 (2009).
- 10) Taesombut, N. and Chien, A. A.: Evaluating Network Information Models on Resource Efficiency and Application Performance in Lambda-Grids, *Proc. SC07* (2007).
- 11) 安藤誠士郎, 合田憲人: 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, 情報処理学会研究報告 2007-HPC-113, pp. 37–42 (2007).
- 12) 竹房あつ子, 中田秀基, 工藤知宏, 田中良夫: 高品質分散実行環境のための計算・ネットワーク資源のグローバルスケジューリング手法, 情報処理学会研究報告 2008-HPC-119, pp. 55–60 (2009).
- 13) 竹房あつ子, 中田秀基, 工藤知宏, 田中良夫: 性能を保証する計算・ネットワーク資源のコアロケーション手法の評価, 情報処理学会研究報告 2009-HPC-121.
- 14) GLPK (GNU Linear Programming Kit): <http://www.gnu.org/software/glpk/glpk.html>.
- 15) MiniSat: <http://minisat.se/>.
- 16) Tanjo, T., Tamura, N. and Banbara, M.: Sugar++: A SAT-based Max-CSP/COP Solver, *Proc. the Third International CSP Solver Competition*, pp. 144–151 (2008).
- 17) 竹房あつ子, 中田秀基, 工藤知宏, 田中良夫, 関口智嗣: 多様な資源を事前予約で同時確保するためのグリッドコアロケーションシステムフレームワーク GridARS, 情報処理学会論文誌コンピューティングシステム (ACS20), Vol. 48, No. SIG18, pp. 32–42 (2007).
- 18) 小島功, 木本正裕, 的野晃整: データ型別索引に基づく異種情報の検索システムとそれを用いた地球観測情報レジストリの実現, 情報処理学会等「データ工学と情報マネジメントに関するフォーラム 2009」 (2009). <http://db-event.jpn.org/deim2009/proceedings/files/B2-4.pdf>.
- 19) Takefusa, A., Nakada, H., Yanagita, S., Okazaki, F., Kudoh, T. and Tanaka, Y.: Design of a Domain Authorization-based Hierarchical Distributed Resource Monitoring System in cooperation with Resource Reservation, *Proc. HPC Asia 2009* (2009).
- 20) G-lambda: <http://www.g-lambda.net/>.
- 21) Ahuja, R. K., Magnanti, T. L. and Orlin, J. B.: *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall (1993).
- 22) Naoyuki Tamura, T. T. and Banbara, M.: System Description of a SAT-based CSP Solver Sugar, *Proc. the Third International CSP Solver Competition*, pp. 71–75 (2008).
- 23) Tamura, N., Taga, A., Kitagawa, S. and Banbara, M.: Compiling Finite Linear CSP into SAT, pp. 254–272 (2009).
- 24) ILOG CPLEX: <http://www.ilog.co.jp/product/opti/cplex/cplex.html>.
- 25) Roblitz, T.: Global Optimization For Scheduling Multiple Co-Reservations In The Grid, *Proc. CoreGRID Symposium*, pp. 93–109

- (2008).
- 26) Elmroth, E. and Tordsson, J.: A standards-based Grid resource brokering service supporting advance reservations, coallocation and cross-Grid interoperability, *Concurrency and Computation: Practice and Experience* (2009).