

多種資源を対象とするオンライン コアロケーション手法の提案

竹房 あつ子^{†1} 中田 秀基^{†1}
工藤 知宏^{†1} 田中 良夫^{†1}

分散環境におけるデータインテンシブ計算の重要性が多分野において高まっているが、データインテンシブ計算で安定した実効性能を得るには、ネットワーク、計算機、ストレージを含めた資源とそれらの属性情報を考慮したコアロケーション手法が必要となる。本研究では、多種資源とその属性情報を考慮した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、既発表手法を拡張して整数計画モデルに新たな制約を加えることで、多種資源のコアロケーションを可能にする。また、整数計画法のソルバに対する Java インタフェースを提供する JavaILP を用いて提案手法に対して複数整数計画ソルバを適用できるように実装するとともに、シミュレーションによりその基本性能を調査した。その結果、資源の一部をユーザが指定する場合は 31.7msec と高速に処理できることと、属性指定がある場合は通常と同程度で処理することが分かり、提案手法の実用性が示された。

On-line Resource Co-allocation Scheme for Various Resources

ATSUKO TAKEFUSA,^{†1} HIDEMOTO NAKADA,^{†1}
TOMOHIRO KUDOH^{†1} and YOSHIO TANAKA^{†1}

A co-allocation scheme, which considers network, computers, and storage and these attributes, is one of key issues for constructing a virtual infrastructure for data-intensive applications over distributed environment because it is difficult to estimate the performance of applications, executed on unstable environment. We propose an on-line co-allocation scheme, which takes into account various resources and attributes, extending our integer programming-based scheme. We implement the proposed co-allocation scheme, by using the JavaILP interface, which enables to adopt multiple integer programming solvers. We also investigate the basic performance of the proposed scheme using simulation. The simulation results show that our scheme can rapidly determine a reservation plan in 31.7

msec when a user specifies a part of requested resources, and is comparable to the basic scheme when resource attributes are specified.

1. はじめに

高エネルギー物理学、地球科学、天文学からスマートグリッドをはじめとするセンサーネットワークなど、多分野において高性能な実験装置やセンサー等から生成された大規模データを収集・解析する、データインテンシブ計算の重要性が高まっている^{1),2)}。データインテンシブ計算では、希少な実験装置を用いる、観測地ごとに得られる情報が異なる、解析結果をお互いに検証するなどの理由で、分散する複数組織の資源間で大規模データを共有し、処理する必要がある。その際、ネットワーク性能がデータインテンシブ計算の処理性能を大きく左右する。よって、計算機、ストレージ等の資源とダイナミックサーキットネットワークを組み合わせ、大規模データインテンシブ計算のための基盤（以後仮想インフラと呼ぶ）を構築するための研究開発が国内外で進められている³⁾⁻⁶⁾。

我々も、仮想インフラの構築を目的とした資源管理フレームワーク GridARS^{7),8)}を開発しており、多種資源の共通予約インタフェースである GNS-WSI3⁹⁾を実装している。GNS-WSI3はG-lambdaプロジェクト⁹⁾でネットワークの帯域予約を目的として策定されたものであるが、資源を抽象的に表現することによりインタフェースを多様な資源に適用することができる。GridARSフレームワークでは、抽象資源を計算機やストレージに拡張することで、計算機の性能、台数、メモリサイズ、ストレージのサイズ、読み書き性能など多種資源の様々な属性情報を考慮した資源の確保が可能となった¹⁰⁾。

仮想インフラの構築では、多種資源からユーザの要求を満たす適切な資源を選択する必要がある。計算機とネットワークをコアロケーションするための手法としては、我々が提案した整数計画法に基づく手法¹¹⁾をはじめ、複数提案されている^{12),13)}。しかしながら、これらは計算機とその間のネットワークの帯域の確保のみに着目しており、ストレージ資源の割り当てや、利用するアプリケーションライブラリの有無など、他の資源やその属性情報を考慮した資源のコアロケーションは行っていない。さらに、実運用では計算に利用するデータの所在などの情報を仮想インフラのユーザが管理しており、ユーザが確保する資源の一部を指定したい場合にも対応する必要がある。

^{†1} 産業技術総合研究所 National Institute of Advanced Industrial Science and Technology (AIST)

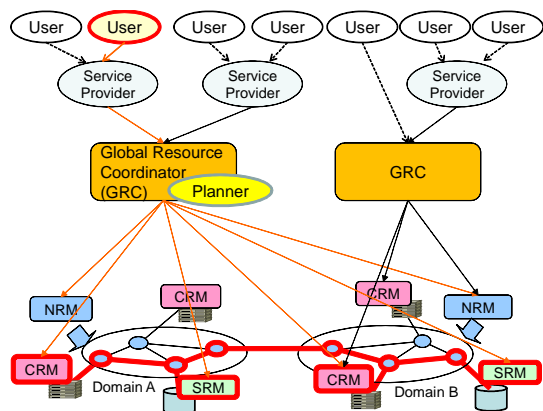


図 1 コアロケーションシステムの概要

本研究では、ネットワーク、計算機、ストレージおよびその属性情報を考慮した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、我々の手法¹¹⁾を改良し、既存の整数計画モデルに対して新たな制約を加えることで、多種資源とその属性情報を考慮するとともにユーザの資源指定を反映させたコアロケーションを実現する。また、提案手法の実装では整数計画法のソルバに対する Java インタフェースを提供する JavaILP¹⁴⁾を用いた。これにより、提案手法に対して複数整数計画ソルバを適用可能にした。さらに、シミュレーションを用いて提案手法の基本性能を調査した。実験では、通常のコアロケーションと属性情報を考慮する場合、資源の一部がユーザにより指定される場合の 3 通りについてコアロケーションの求解時間を比較した。実験から、資源指定がある場合は 31.7msec と高速に処理できること、属性指定がある場合は通常と同程度で処理できることを示す。

2. 多種資源の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

多種資源のコアロケーションでは、広域ネットワークを含めた資源を扱うため、複数の異なる事業者(ドメイン)から予約ベースで資源が提供されることを想定する。我々は、複数ドメインから提供される資源を確保・提供するための資源管理フレームワーク GridARS を開発しており、GridARS および本研究で提案するコアロケーション手法が前提とするコアロケーションシステムの概要を述べる。

図 1 に、コアロケーションシステムの概要を示す。図中の User は最終的な利用者を表し、Service Provider は確保された資源の上にサービスを構築して利用者に提供するサービスプロバイダを表す。コアロケーションシステムの主なユーザは、サービスプロバイダとなる。サービスプロバイダは、構築するサービスに必要な資源をコアロケーションシステムに要求し、コアロケーションシステムが複数ドメインを跨る資源を確保、提供する。

コアロケーションシステムは、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 はスケジューリング機能(プランナ)をもち、ユーザの要求に応じて各 RM に利用可能な資源を問い合わせることで資源の予約プランを決定する。決定した予約プランに従って、GRC が関連する RM に資源確保を要求し、予約時刻に各 RM がユーザに資源を提供する。

2.2 対象資源とその属性情報

確保対象となる資源は、計算、ストレージ、ネットワーク資源を想定する。ただし、コアロケーションではユーザの要求に応じて各資源が提供するサービス、すなわち冗長性の有無、読み書き性能等の属性情報も考慮する必要がある。属性情報としては、以下のようなものが考えられる。

計算資源 プロセッサ/コア数、プロセッサアーキテクチャ、メモリサイズ、ディスクサイズ、IP アドレス(レンジ)、OS、アプリケーションライブラリ。

ストレージ資源 スペース(サイズ、読み書きスループット)、スペース/ファイルアクセス(許可、スループット、アクセスモード)、冗長性。

ネットワーク資源 帯域、遅延、スイッチング手法、VLAN タグ ID、冗長性。

ストレージ資源では、スペースの他、アクセスの可否も考慮する必要がある。これは、一般にデータインテンシブ計算やコンテンツ配信等のサービスでは、スペースを確保するユーザと確保したスペースにアクセスするユーザは 1 対多になる。よって、アクセスの可否も考慮することで安全かつ高品質なサービスの構築をサポートする必要がある。

2.3 資源要求の指定

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、各サイトでのプロセッサ数とその他の属性情報を、ネットワーク資源要求では要求する拠点間の帯域、遅延、その他の属性情報を指定している。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。予約プラン、すなわちコアロケーション結果では、通信経路は実際のネットワークのルーティング情報がドメイン

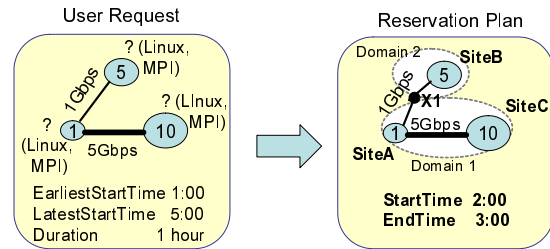


図2 ユーザの資源要求とプランナによる予約プラン

ン毎に隠ぺいされた抽象化されたトポロジで表される。

GRCは図2のようなユーザからの資源要求に基づいてプランニングするが、利用したい計算機やストレージ、ネットワークなどをユーザが明示的に指定したい場合がある。例えば、コンテンツ配信サービスでは、サービスプロバイダがどのストレージに何のコンテンツを格納しているか、サービスの利用者がどこからアクセスしているかを知っているが、コアロケーションシステム側にはその情報は開示されていない。よって、仮想インフラ上で品質の保証されたサービスを構築するには、コアロケーションシステムのユーザが利用するデータの所在や最終的にデータを転送したいストレージや計算機の場所などを、自身の管理する情報に基づいて部分的に指定できる必要がある。よって、本研究のコアロケーション手法でも、そのような部分的に資源が指定されている資源要求も、通常の資源要求と同じ手続きで行えるようにする。

3. 多種資源を対象とするオンラインコアロケーション手法

我々の既発表手法である0-1整数計画法(以降IPとする)に基づく計算資源とネットワークのコアロケーション手法¹¹⁾を拡張し、多種資源を対象とするオンラインコアロケーション手法を提案する。

3.1 既存コアロケーション手法の概要

既発表手法では、以下の手順で予約プランを作成する。本研究でも同様の手順で行い、ステップ(3c)を拡張する。

- (1) 静的資源情報をあらかじめレジストリから取得する。
- (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。

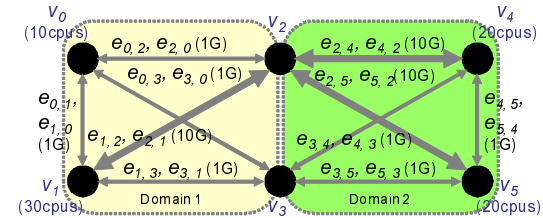


図3 資源グラフ

- (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、IPモデルを用いて N' ($N' \leq N$) 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。

3.2 既存IPモデル

コアロケーションをモデル化するため、資源群および資源要求をグラフで表す。資源群は、図3のように双方向の有向グラフ $G = (V, E)$ で表す。 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_q 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 $e_{m,n}$ または $e_{n,m}$ は各NRMが提供する資源間のパスを表す。図3の v_2, v_3 はドメイン間交換点を示す。グラフの点および枝の括弧内は、提供可能なプロセッサ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。また、提供するプロセッサ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、 $e_{m,n}, e_{n,m}$ で共有する。

資源要求は完全グラフ $G_r = (V_r, E_r)$ 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 に必要なプロセッサ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、図4のように計算資源サイト $x_{i,j}$, パス $y_{k,l}$ の値を以下のIPモデルで解くことで予約プランを決定する。 $x_{i,j}, y_{k,l}$ は資源要求に対して選択される資源の値は1, その他は0となる。

	Resources																												
Request	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="border: none;"></td> <td style="border: none; text-align: center; color: green;">v_0</td> <td style="border: none; text-align: center; color: green;">v_1</td> <td style="border: none; text-align: center; color: green;">v_2</td> <td style="border: none; text-align: center; color: green;">v_3</td> <td style="border: none; text-align: center; color: green;">v_4</td> <td style="border: none; text-align: center; color: green;">v_5</td> </tr> <tr> <td style="border: none; text-align: right;">vr_0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center; color: red;">1</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> </tr> <tr> <td style="border: none; text-align: right;">vr_1</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center; color: red;">1</td> <td style="border: none; text-align: center;">0</td> </tr> <tr> <td style="border: none; text-align: right;">vr_2</td> <td style="border: none; text-align: center; color: red;">1</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> </tr> </table>		v_0	v_1	v_2	v_3	v_4	v_5	vr_0	0	1	0	0	0	0	vr_1	0	0	0	0	1	0	vr_2	1	0	0	0	0	0
	v_0	v_1	v_2	v_3	v_4	v_5																							
vr_0	0	1	0	0	0	0																							
vr_1	0	0	0	0	1	0																							
vr_2	1	0	0	0	0	0																							

	Resources																																
Request	<table border="1" style="border-collapse: collapse; text-align: center;"> <tr> <td style="border: none;"></td> <td style="border: none; text-align: center; color: green;">$e_{0,1}$</td> <td style="border: none; text-align: center; color: green;">$e_{1,0}$</td> <td style="border: none; text-align: center; color: green;">$e_{1,2}$</td> <td style="border: none; text-align: center; color: green;">$e_{2,1}$</td> <td style="border: none; text-align: center; color: green;">$e_{2,4}$</td> <td style="border: none; text-align: center; color: green;">$e_{4,2}$</td> <td style="border: none; text-align: center;">...</td> </tr> <tr> <td style="border: none; text-align: right;">$er_{0,1}$</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center; color: red;">1</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center; color: red;">1</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">...</td> </tr> <tr> <td style="border: none; text-align: right;">$er_{0,2}$</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center; color: red;">1</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">...</td> </tr> <tr> <td style="border: none; text-align: right;">$er_{1,2}$</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">0</td> <td style="border: none; text-align: center;">...</td> </tr> </table>		$e_{0,1}$	$e_{1,0}$	$e_{1,2}$	$e_{2,1}$	$e_{2,4}$	$e_{4,2}$...	$er_{0,1}$	0	0	1	0	1	0	...	$er_{0,2}$	0	1	0	0	0	0	...	$er_{1,2}$	0	0	0	0	0	0	...
	$e_{0,1}$	$e_{1,0}$	$e_{1,2}$	$e_{2,1}$	$e_{2,4}$	$e_{4,2}$...																										
$er_{0,1}$	0	0	1	0	1	0	...																										
$er_{0,2}$	0	1	0	0	0	0	...																										
$er_{1,2}$	0	0	0	0	0	0	...																										

図4 求める変数 $x_{i,j}$, $y_{k,l}$

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

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

Minimize

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

Subject to

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

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

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

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

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

$$\forall l = (o, p) \in E_r, \forall m \in V,$$

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

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

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

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2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 において要求されるプロセッサ数は、提供されるプロセッサ数以下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 (5) および式 (7) の制約は、計算資源要求を別々のサイトに割り当てるものであり、同じサイトに割り当てることを許す場合はこの制約を省略する。式 (9) は、流量保存則¹⁵⁾ を応用して $x_{i,j}$ と $y_{k,l}$ を関連付ける制約となる¹¹⁾。

式 (10), (11) は、IP の探索範囲を削減して求解時間を短縮するための枝刈り条件となる制約である。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。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 P_{max} を指定するものであり、 P_{max} はネットワークドメインの数などをもとに適宜決定する。

3.3 IP モデルの拡張

本研究では、3.2 節を拡張して、多種資源とその属性情報への対応、部分的な資源要求指定がある場合の対応方法について述べる。

多種資源とその属性情報への対応

3.2 節のモデルでは、計算サイト、パスに対して式 (6), (8) でプロセッサ数および帯域を満たすかどうかのアロケーション対象資源に関する制約を加えて予約プランを求める。これに、2.2 節で対象とする資源のアロケーションを行うため、以下のような拡張を行う。

まず、ストレージに関する資源は、計算サイトの変数 $x_{i,j}$ に関する属性情報と考え、対象とする資源の属性情報は計算サイト $x_{i,j}$ またはパス $y_{k,l}$ のいずれかの属性情報とする。次に、各属性情報に関する制約条件は、スカラ値で表すものとブール値で表すもの、各要求資源に対して制約式を解くものと各資源に対して制約式を解くものに分類することができる。例えば、 $x_{i,j}$ に関する資源ではプロセッサ数は制約条件がスカラ値で表され、各資源に対して制約式を解く。一方、アプリケーションライブラリの有無は、ブール値で表され、各要求資源に対して制約式を解く。 $y_{k,l}$ では、ネットワークの帯域はスカラ値で表された制約式を各資源に対して解き、冗長化の有無はブール値で表された制約式を各要求資源に対して解く。

すなわち、対象資源に関する制約は以下の 8 通りの式で表すことができる。

$$\forall i \in V, \sum_{j \in V_r} param(p)_j \cdot x_{i,j} \leq ws(p)_i \quad (12)$$

$$\forall j \in V_r, \sum_{i \in V} param(p)_i \cdot x_{i,j} \leq ws(p)_j \quad (13)$$

$$\forall i \in V, \sum_{j \in V_r} (1 - param(p)_j) \cdot x_{i,j} = 0 \quad (14)$$

$$\forall j \in V_r, \sum_{i \in V} (1 - param(p)_i) \cdot x_{i,j} = 0 \quad (15)$$

$$\forall k \in E, \sum_{l \in E_r} param(q)_l \cdot y_{k,l} \leq ws(q)_k \quad (16)$$

$$\forall l \in E_r, \sum_{k \in E} param(q)_k \cdot y_{k,l} \leq ws(q)_l \quad (17)$$

$$\forall k \in E, \sum_{l \in E_r} (1 - param(q)_l) \cdot y_{k,l} = 0 \quad (18)$$

$$\forall l \in E_r, \sum_{k \in E} (1 - param(q)_k) \cdot y_{k,l} = 0 \quad (19)$$

ここで、 p, q は計算およびネットワーク資源の属性情報の集合 IX, IY の 1 つを表し、 $param(p/q)$ はスカラ値またはブール値、 $ws(p/q)$ はスカラ値のパラメータを表す。また、 $param(p)_j$ は要求する計算サイト j が p を必要とする量、 $ws(p)_i$ は計算サイト i が提供可能な p の量を表す。例えば、 p がプロセッサの場合、 $param(p)_j$ は 10、 $ws(p)_i$ は 32 のような値となる。一方、 $param(p)_i$ がブール値の場合は計算サイト i が p を満たすかどうか

表 1 対象資源と制約式の関係

制約式	スカラ/ブール	演算方向	対象資源
式 (12)	スカラ	↓	プロセッサ/コア, メモリ, ディスク
式 (13)	スカラ	⇒	スペースサイズ/読み書きスループット, アクセススループット
式 (14)	ブール	↓	
式 (15)	ブール	⇒	プロセッサアーキテクチャ, OS, ライブラリ, 冗長性 アクセス許可, アクセスモード, 冗長性
式 (16)	スカラ	↓	帯域
式 (17)	スカラ	⇒	遅延
式 (18)	ブール	↓	
式 (19)	ブール	⇒	スイッチング手法, 冗長性

を、0 または 1 で表す。例えば、指定したアプリケーションライブラリがある場合は 1、ない場合は 0 となる。

式 (12) ~ (19) に 2.2 節の対象資源を当てはめると表 1 のようになる。表中の“演算方向”は、各資源に対して制約式を解くもの (↓) と各要求資源に対して制約式を解くもの (⇒) を表している。表 1 のように、ほぼ全ての属性情報は式 (12) ~ (19) で表すことが可能であり、いずれかに適宜当てはめて 3.2 節の既存 IP モデルに追加の制約式として加えていけばよい。

ただし、計算資源の属性情報である IP アドレス (レンジ)、およびネットワーク資源の属性情報である VLAN タグ ID はこの制約式に当てはめることができない。これは IP アドレスや VLAN タグ ID そのものが資源であり、別に変数を定義しなければ解くことができないためである。本研究では、IP アドレス (レンジ) および VLAN タグ ID は $x_{i,j}, y_{k,l}$ を求めた後、ヒューリスティックに決定することにする。

また、目的関数は、以下のように修正する。

Minimize

$$\sum_{i \in V, j \in V_r, p \in IX} v(p)_i \cdot param(p)_j \cdot x_{i,j} + \sum_{k \in E, l \in E_r, q \in IY} v(q)_k \cdot param(q)_l \cdot y_{k,l} \quad (20)$$

$v(p/q)_{i/k}$ は、各属性情報の単価を表す。すなわち、1 プロセッサあたりの単価、冗長化にともなう価格などが入る。以上の拡張により、多種資源の属性情報を考慮したコアロケーションを IP モデルで表すことができる。

資源要求指定への対応

2.3 節で述べたように、ユーザがコアロケーション対象の資源を部分的に指定したい場合

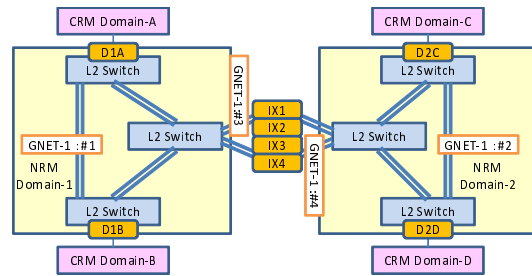


図 5 デモ環境

がある．例えば，要求計算資源サイトに対して1つめのサイトはAにしたい，またはAかBかCにしたい，という要求がある．このような場合も，制約式を加えることで表現することができる．すなわち，前者の場合は

$$x_{A,j} = 1 \quad (21)$$

$$x_{i,j} = 0 (i \neq A) \quad (22)$$

を，後者の場合は

$$x_{i,j} = 0 (i \neq A, B, C) \quad (23)$$

を制約として加えればよい．同様に，ネットワーク資源に対しても指定したパスを1，該当しないパスを0とする制約を加えることで，ユーザの部分的な資源指定が有効になる．

排他的資源

この他，実環境に実装する場合には，排他的な資源を扱うことがある．例えば，我々がGLIF2010およびSC10において実証実験を行った環境⁸⁾は図5のようになっている．利用可能な帯域としては，ドメイン内の各サイトからドメイン交換点IX1, IX2, IX3, IX4に対してそれぞれ1[Gbps]のパスが提供可能であるという情報が得られる．しかしながら，実際にはドメイン交換点では1[Gbps]のパスはそれぞれ1本ずつしか通すことができず，D1A-IX1, IX1-D2C, D1B-IX1, IX1-D2Dのような2本のパスの予約要求は失敗してしまう．このように，予め排他的な資源が分かっている場合は，ドメイン交換点を含むパス要素 $y_{k,l}$ の和が，それぞれ2以下になるようにすればよい．IXsが排他的なドメイン交換点の集合だとすると，次のように表すことができる．

$$\forall ix \in IXs, \sum_{l \in E_r, m \in V, m \neq ix} (y_{(m,ix),l} + y_{(ix,m),l}) \leq 2 \quad (24)$$

```
import net.sf.javaillp.*;
public class JavaILLPTest {
    public static void main() {
        Problem problem = new Problem();
        // 変数の設定
        problem.setVarType(<変数名>, Integer.class);
        :
        // 制約式の設定
        problem.add(<線形式>, <演算子>, <値>);
        :
        // 目的関数の設定
        problem.setObjective(<線形式>, <Minimize/Maximize>);
        // Solverの取得
        SolverFactory factory = new SolverFactoryCPLEX();
        Solver solver = factory.get();
        // 求解
        Result result = solver.solve(problem);
        System.out.println(result);
    }
}
```

図 6 JavaILLP を用いたプログラム例

4. 提案手法の実装

提案したIPモデルから $x_{i,y}, y_{k,l}$ を求めるには，IPソルバを用いる必要がある．しかしながら，IPソルバは多岐にわたり，各IPソルバ提供するライブラリやモデルの記述形式が異なる．また，フリーのものと商用のものもある．GridARSの資源管理システムはJavaで実装しているため，JavaILLP¹⁴⁾を用いて提案IPモデルを実装して複数のソルバが適用できるようにした．

4.1 JavaILLPを用いた実装

JavaILLP (Integer Linear Programming) は，複数の整数計画ソルバに対してJavaの共通インタフェースを提供している．サポートしているソルバには，GLPK¹⁶⁾，lp_solve¹⁷⁾，IBM ILOG CPLEX¹⁸⁾，Gurobi¹⁹⁾などがある．実行時にjava.library.pathで，利用するソルバのJNI (Java Native Interface) jarファイルとソルバのライブラリファイルの所在を指定することで，JavaILLPから各ソルバを実行することができる．

JavaILLPは，非常にシンプルなインタフェースを提供しており，図6のようにIPを解く

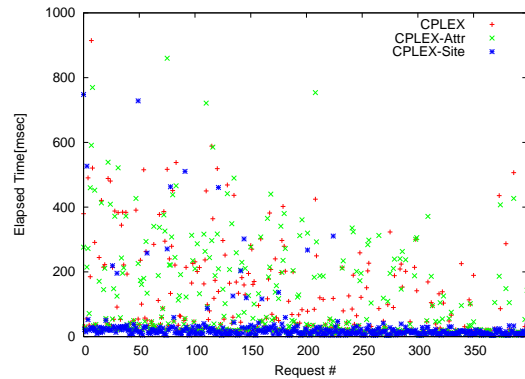


図 7 各資源要求における求解時間の比較.

手法	平均値 [msec]	最大値 [msec]	標準偏差
CPLEX	96.8	914.6	238.7
CPLEX 属性指定	106.2	860.0	269.5
CPLEX サイト指定	31.7	747.8	107.9

ことができる。まず, Problem クラスのインスタンス problem に変数, 制約条件, 目的関数, を設定する。ただし, 行列や \sum などの表現はサポートしていないため, プログラム中で式を展開する必要がある。次に, 利用するソルバの SolverFactory クラスのインスタンス factory を生成し, factory を用いてソルバクラスのインスタンス solver を生成する。solver で設定した problem を解くと, 結果として Result クラスのインスタンス result が得られる。result から目的関数や変数の値を適宜取り出すことができる。

4.2 基本性能の評価

提案手法の基本性能を示すため, 既発表研究¹¹⁾ で用いたシミュレーションの 1 シーケンスにおける IP モデルの求解時間を調査した。このシミュレーションでは, 3 つのネットワークメイン, 10 の計算資源サイト, 2 つのドメイン交換点を想定し, 2-4 サイトの計算資源とその間のネットワークをランダムに要求する。シミュレーションは, いずれも Intel Core2 Quad CPU (2.66GHz), CentOS 5.3, kernel 2.6.18 x86_64, メモリ 4GB の計算機上で実行した。ソルバには, IBM ILOG CPLEX Optimization Studio V12.2 の 32bit 版を用いた。

評価では, CPLEX を用いて資源情報のみ用いた場合, 属性情報が指定された場合, サイ

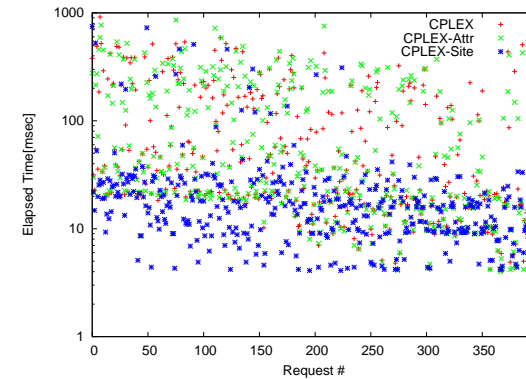


図 8 各資源要求における求解時間の比較 (ログスケール).

トが指定された場合で比較した。属性情報が指定された場合は, プロセッサアーキテクチャなどが指定された場合を想定し, 指定した属性を持つ計算資源サイトが全体の 3/4 となるようにした。また, サイトが指定された場合では, 計算資源サイトのうち 1 つ目のサイトをランダムに指定するようにした。

図 7 に求解時間をプロットした結果を, 図 8 にそれをログスケールで表したものを示す。縦軸は求解時間を msec で表し, 横軸は要求された資源要求の順序を表しており, 資源要求番号が大きくなるにつれて利用可能な資源が少なくなっていく。各プロットは, 各要求に対して予約候補数を $N=10$ とし, 10 回の計算時間の平均値をプロットしたものである。グラフから, いずれもばらつきがあるものの 1 秒以下で求解できることがわかる。また, 既発表研究におけるシミュレーション結果同様, 利用可能な資源が少なくなると求解時間が短くなる傾向がみられた。

表 2 に求解時間の平均値, 最大値, 標準偏差を比較した結果を示す。平均値で比較すると, サイト指定した場合が 31.7msec と最も良い結果が得られた。これにより, データの所在などに基づきサイト候補が指定される場合には, 求解時間が短縮できることが示唆された。属性指定した場合は, 平均値では通常より 10msec 程度求解時間が長くなっていったが, 最大値では属性指定がない場合よりある場合の方が 50msec 以上短くなっていった。よって, 属性情報がある場合も遜色なく求解できることが示された。

5. 関連研究

ネットワークと計算機を対象としたコアロケーション手法としては、バックトラック法による探索¹²⁾、計算サイトの組み合わせを決定してからその間のネットワークを探索する手法¹³⁾がある。ネットワークとストレージ資源を探索する手法としては、文献²⁰⁾では映像配信サービスを想定したコアロケーション手法を提案している。この方法では、コンテンツの所在情報をもとに端点を決定し、その後端点間のネットワークを探索する。

いずれの手法も、順次適切な組み合わせを探索していくのに対し、我々の手法はIPモデルを1回解くだけでよい点、多種資源とその属性情報を扱える点で異なる。

6. まとめ

本研究では、ネットワーク、計算機、ストレージおよびその属性情報を考慮したオンラインコアロケーション手法を提案した。提案手法により、既存IPモデルに対して制約式を機械的に加えることで、多種資源とその属性情報を考慮し、かつユーザの資源指定を反映するコアロケーションを可能にした。また、複数IPソルバが適用可能なコアロケーション手法の実装について述べるとともに、シミュレーションを用いて提案手法の基本性能を調査してその実用性を示した。今後は、より大規模な環境で実際のデータインテンシブ計算を想定した評価を行うとともに、GridARSフレームワークへの実装を行う。

謝辞 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) LHC: <http://lhc.web.cern.ch/lhc/>.
- 2) GEOGrid: <http://www.geogrid.org/>.
- 3) Takefusa, A., Hayashi, M., Nagatsu, N., Nakada, H., Kudoh, T., Miyamoto, T., Otani, T., Tanaka, H., Suzuki, M., Sameshima, Y., Imajuku, W., Jinnumber, 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).
- 4) 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., Hiranumber, 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).
- 5) GÉANT: <http://www.terena.org/activities/nrens-n-grids/workshop-08/slides/05-GN3-Vicinanza.pdf>.
 - 6) Internet2 Dynamic Circuit Network: <http://www.internet2.edu/network/dc/>.
 - 7) 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).
 - 8) 竹房あつ子, 中田秀基, 高野了成, 柳田誠也, 大久保克彦, 工藤知宏, 田中良夫: マルチドメインクラウド資源管理フレームワークの実証実験, 電子情報通信学会技術研究報告 (印刷中).
 - 9) G-lambda プロジェクト: <http://www.g-lambda.net/>.
 - 10) Tanimura, Y., Koie, H., Kudoh, T., Kojima, I. and Tanaka, Y.: A Distributed Storage System Allowing Application Users to Reserve I/O Performance in Advance for Achieving SLA, *Proc. Grid2010* (2010).
 - 11) 竹房あつ子, 中田秀基, 工藤知宏, 田中良夫: 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, 情報処理学会論文誌 コンピューティングシステム, Vol.3, No.3, pp.126–137 (2010).
 - 12) 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).
 - 13) 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*, Vol.25, No.18, pp.2298–2335 (2009).
 - 14) Java ILP: <http://javailp.sourceforge.net/>.
 - 15) Ahuja, R.K., Magnanti, T.L. and Orlin, J.B.: *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall (1993).
 - 16) GLPK (GNU Linear Programming Kit): <http://www.gnu.org/software/glpk/glpk.html>.
 - 17) lp_solve: <http://lpsolve.sourceforge.net/5.5/>.
 - 18) IBM ILOG CPLEX: <http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>.
 - 19) Gurobi: <http://www.gurobi.com/>.
 - 20) 山田一久, 築島幸男, 松田和浩, 谷村勇輔, 工藤知宏, 竹房あつ子, 高野了成, 清水敬司: 超高精細映像配信サービスのためのストレージ・ネットワーク統合資源管理方式の提案, 電子情報通信学会技術研究報告, CS2010-49 (2010-11), pp.71–76 (2010).