Vol.2013-HPC-140 No.11
2013/7/31

OpenFlow 連携ジョブ管理システムの実装と評価

渡場 康弘[1]　木戸 善之[1]　伊達 進[1]　阿部 洋丈[2]　市川 昊平[3]　山中 広明[4]　河合 栄治[4]　竹村 治雄[1]

概要：大規模化・分散化傾向にあるクラスタ型計算資源の利用において，ネットワーク性能が計算処理性能に与える影響は大きい．しかし，ユーザジョブの計算要求に応じて計算資源を割り当てるジョブ管理システムの多くは，ネットワーク資源情報を利用せず，CPU やメモリのような計算資源情報に基づいて各ジョブへの割当資源を決定する．本研究ではネットワーク資源と計算資源の情報に基づき，ユーザのジョブ要求に対して最適なネットワークを持つ計算資源を割り当てることができるジョブ管理システムの実現を目指している．本稿では，ネットワークを柔軟に制御することが可能な技術として注目を集めている OpenFlow の機能を応用したジョブ管理システムの概要をまとめるとともに，提案システムの有効性を検証した実験結果について報告する．

1. はじめに

物理シミュレーションや流体解析に代表される科学技術計算において，計算環境のネットワーク性能が計算処理性能に与える影響が大きくなっている．その理由の 1 つとして，多数の計算ノードが相互にネットワークで接続されたクラスタ型計算資源では，計算中に計算ノード間で頻繁に通信する必要があることがあげられる．クラスタ型計算資源は今日の計算環境における主流の構成であり，この構成の計算資源で高い計算処理性能を得るためには，複数の計算ノードを効率的に利用した分散並列計算を行う必要がある．実際，現在のスーパーコンピュータに代表される高性能計算資源の約 80%がクラスタ型計算資源であり，数百台以上の計算ノードで構成されるクラスタ型計算資源も多数存在する [1]．また，キャンパス内の余剰計算資源を活用するキャンパスグリッドのような計算環境では，計算ノードがキャンパス内に分散して配置されるため，計算処理性能がネットワーク性能に依存する度合いはより高くなる．

このようなクラスタ型計算資源の管理において，効率的な計算資源の割当を行うシステムとしてジョブ管理システム（Job Management System, JMS）がある．JMS は主に High-Performance Computing（HPC）に代表される科学技術計算環境で利用されている．このような JMS の例として NQS [2]，PBS [3]，Open Grid Scheduler/Grid Engine（OGS/GE） [4] などをあげることができる．これらの JMS はユーザジョブに適切な計算資源を割り当てるために必要な計算資源情報や利用状況を把握する機能を有する．

しかし，一般的な JMS での計算資源割当は，CPU やメモリなどの計算資源に対する利用情報およびユーザの計算資源要求に基づいて割当計算資源を決定しており，計算ノード間を接続するネットワークの効率的な利用を考慮する設計にはなっていない．このことは今日利用可能な JMS が，計算ノード間を接続するネットワークは常に十分な性能を保有していること，およびネットワークを動的に制御可能な資源とみなしていないことに起因すると考えられる．一方，計算ノードの数がますます増加し大規模化していく今日のクラスタ型計算資源の現状を鑑みれば，前述の前提を満たすネットワークを有するクラスタ型計算資源の構築は，多大なコストを必要とするため実現が困難になりつつある．本研究では，常に十分なネットワーク性能が保証されない計算環境であっても，ネットワーク資源と計算資源の情報を管理し，ユーザの計算資源要求に対して両方の資源情報を統合的に考慮することで，ジョブに最適な計算資源を割り当てることができるジョブ管理システムの実現を目指す．

以下，本稿の構成を下記に示す．2 節ではクラスタ型計算資源におけるネットワークの効率的利用に関する関連研究を紹介する．3 節でネットワーク資源の管理・割当を実現するために必要な機能について述べ，4 節で提案する

[1] 大阪大学
Osaka University
[2] 筑波大学
University of Tsukuba
[3] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
[4] 独立行政法人 情報通信研究機構
National Institute of Information and Communications Technology

JMS について説明する．5 節では提案 JMS のプロトタイプシステムにおける遅延および帯域を考慮した計算資源の割当について検証実験を行い，6 節でまとめを行う．

2. 関連研究

クラスタ型計算資源におけるネットワークの効率的利用についてはこれまで様々な研究が行われている．ネットワークの効率的利用に対するアプローチは大きく分類すると，計算資源管理においてネットワーク資源も併せて管理する計算資源管理側からのアプローチと，アプリケーションが割り当てられた計算資源のネットワークを効率的に利用するユーザ側からのアプローチの 2 つに分けられる．

前者のアプローチにおいて，ネットワーク資源も併せて資源管理を行う JMS の 1 つとして，G-lambda プロジェクトのグリッドスケジューリングシステム [5] があげられる．この JMS は拠点間を接続する光パスネットワーク資源の割当を目的として設計されているため，各拠点内のネットワーク資源を考慮しない．また，ネットワーク資源も含めた計算資源をユーザが要求できるように，グリッド環境上に仮想インフラを作成する記述言語 VXDL（Virtual Resources and Interconnection Networks Description Language）[6] が提案されている．VXDL では，物理ネットワーク資源ではなく、仮想ネットワーク資源要求の記述方法を定義している．そのため，仮想インフラとしてユーザに割り当てられた計算資源が要求に適した物理ネットワーク構成を持つ保証はない．

一方，並列計算処理での後者のアプローチによる研究の 1 つとして，ネットワークの遅延やトポロジ情報を取得し，その情報に応じて適した MPI 集合通信アルゴリズムを使用する研究がある [7]．この手法は割り当てられた計算資源内でより高い処理性能を実現する．しかし，収集できるネットワーク情報は割り当てられた計算資源内だけであるため，他のユーザジョブのネットワーク利用による影響は考慮できない．

3. ネットワーク資源に対する機能要件

従来の JMS はネットワーク資源に対する管理機能を有していない．本研究では，JMS でネットワーク資源情報を利用した計算資源割当を実現するためには，次の 4 つの機能を JMS に拡張する必要があると考える．

(1) ネットワーク資源の情報収集機能
(2) ネットワーク資源の割当管理機能
(3) ネットワーク資源を考慮した割当計算資源決定機能
(4) ネットワーク資源に対するユーザ要求機能

（1）の機能では，ネットワーク資源を考慮した計算資源割当処理で必要となるネットワーク情報の収集を実現する．収集するネットワーク資源情報としては，計算ノード間の通信経路の情報および利用状況を把握する必要があるため，計算環境のネットワークトポロジ，計算ノードとスイッチおよびスイッチ間の各リンクにおける遅延や利用可能帯域が考えられる．

（2）の機能は，計算資源割当処理で決定したネットワーク資源割当に基づき，ユーザジョブに割り当てられた通信経路を管理するために必要な機能である．割り当て済み通信経路の管理情報は，共用資源であるネットワークがどのようにユーザジョブに割り当てられているかを把握するために必要となる．また，Fat-Tree で構築されたクラスタ型計算資源のように計算ノード間に複数の通信経路がある場合には，各ジョブが利用する通信経路を制御するために本機能が必要となる．

（3）の機能は，収集されたネットワーク資源情報および計算資源情報から資源割当ポリシに従って割り当てる計算資源を決定するために必要である．従来 JMS における割当計算資源の決定機能ではネットワーク資源情報を利用することができないため，本機能で計算資源およびネットワーク資源の両方の情報を統合的に利用した計算資源割当を実現する．

（4）の機能は，計算資源と同様にユーザがネットワーク資源に対して要求を出すための機能である．この機能は，ユーザが利用するアプリケーションの通信特性を理解している場合などに，その特性にあったネットワーク資源を要求するために必要となる．

4. 提案システム

4.1 概要

3 節で述べたネットワーク資源に対する機能（1），(2) を実現するため，我々はネットワークの制御を一元的かつ動的に行うことができる OpenFlow 技術 [8] に着目した．OpenFlow は，従来のネットワークでは個々のネットワークスイッチで行われていたネットワークパケットの制御機能と転送機能を分離し，制御機能を 1 つのコントローラに集約する Software-Defined Network（SDN）を実現する技術の 1 つである．

OpenFlow ネットワークは複数の OpenFlow スイッチと 1 つの OpenFlow コントローラで構成されており，データ転送は OpenFlow スイッチの持つ *flow table* の情報に基づいて実行される．*flow table* は対象となるパケットごとに処理を定義した *flow entry* の集合として構成される．1 つの *flow entry* はパケットに対する条件と処理方法および統計情報の 3 つの情報から成る．

OpenFlow ネットワークでは，処理するパケットに該当する *flow entry* が *flow table* に存在する場合は，その *flow entry* 情報に従いパケットを処理する．しかし，該当する *flow entry* が存在しない場合には，そのパケットに対する処理を OpenFlow コントローラに問い合わせる．OpenFlow スイッチからの問い合わせに対し，OpenFlow コントロー

ラが該当パケットに対応した *flow entry* 情報を持っていればその *flow entry* 情報を OpenFlow スイッチに返し，該当無ければ．この OpenFlow コントローラが持つ *flow entry* 情報を操作することによって，従来のネットワーク環境では実施が困難であった動的なネットワーク制御を行うことが可能となる．ネットワーク資源に対する情報収集機能および割当管理機能を前述の OpenFlow の特性を利用して実現する．

3 節の機能（1）について，従来のネットワークではネットワーク資源情報が個々のネットワークスイッチで管理されていたため，情報の収集は困難であった．しかし，OpenFlow ネットワークではネットワーク資源情報も OpenFlow コントローラに集約されているので，その情報を利用することによって計算環境全体のネットワーク情報を収集できる．ただし，OpenFlow コントローラで収集される各ネットワークスイッチの情報だけでは，ネットワーク資源を考慮した資源割当に必要なすべての情報を得られるとは限らない．本プロトタイプシステムでは，計算環境のネットワークトポロジ情報を OpenFlow コントローラが持つネットワーク情報からは得ることができないため，Link Layer Discovery Protocol（LLDP） [9] を利用している．

3 節の機能（2）については，OpenFlow ネットワークでは通信経路を該当パケットに対する *flow entry* で表現するため，計算ノード間の通信経路を *flow entry* 情報として管理する．なお，ユーザジョブに割り当てた計算資源が利用する通信経路は複数あるため，1 つのユーザジョブに複数の *flow entry* が関連付けられる．

以上の OpenFlow を利用したネットワーク資源管理機能は，OpenFlow コントローラを内包する Network Management Module（NMM）と呼ぶ外部モジュールとして実装し，従来の JMS とネットワーク資源管理機能の連携を本モジュールで実現する．また，資源割当ポリシに従いユーザジョブに割り当てる計算資源を決定する 3 節の機能（3）についても NMM で実装する．

4.2 プロトタイプシステムの構成および処理フロー

提案 JMS のプロトタイプシステムの構成およびシステムの処理フローを **図 1** に示す．プロトタイプシステムでは，連携する JMS として OGS/GE，NMM に内包するソフトウェア OpenFlow コントローラとして Trema [10] を使用している．ネットワーク資源の管理機能を持つ NMM は，Brain，Network Control の 2 つのサブモジュールと Database で構成される．

Brain サブモジュールは，管理者によってあらかじめ設定された資源割当ポリシに基づき，計算資源情報およびネットワーク資源情報を基にジョブに割り当てる計算資源の決定を行う NMM のコアモジュールである．そのため，JMS から対象ジョブの情報および計算資源情報，

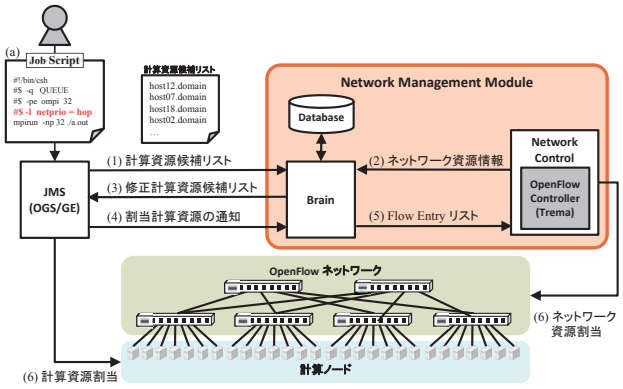


図 1 提案 JMS プロトタイプのシステム構成と処理フロー

Network Control サブモジュールからネットワーク資源情報を取得する機能を有する．また，Brain サブモジュールは Network Control サブモジュールの OpenFlow コントローラへの *flow entry* の登録・削除を命令することができる．Network Control サブモジュールは Trema を内包しており，OpenFlow コントローラの機能を有する．また，Brain サブモジュールに渡すネットワーク資源情報を収集する機能を有する．Database では Network Control サブモジュールから取得したネットワーク資源情報や Brain サブモジュールで各ジョブに割り当てられた *flow entry* 情報を格納する．特に，*flow entry* 情報は各ユーザジョブのジョブ ID と関連付けて管理し，該当ジョブの処理終了後に不要となる *flow entry* 情報を削除するために利用する．

本プロトタイプシステムにおけるユーザジョブへの計算資源割当は次の通り動作する．まず，計算資源割当を行うジョブの選択およびユーザの計算資源要求に該当する計算ノードで構成された計算資源候補リストの作成は OGS/GE が持つ従来 JMS の機能で行われる．この際，計算資源候補リストは計算ノードの CPU ロードアベレージに基づいてソートされる．従来の OGS/GE の場合，この計算資源候補リストの上位から計算ノードにジョブのプロセスを割り当てる．一方，提案 JMS では，ネットワーク資源情報を利用した資源割当ポリシを反映する必要があるため，NMM の Brain サブモジュールに計算資源候補リストを渡す（図 1 (1)）．このとき Brain サブモジュールは該当ジョブ情報および計算資源情報も JMS から取得する．次に Brain サブモジュールはネットワーク資源情報を Network Control サブモジュールから取得する（図 1 (2)）．なお，Network Control サブモジュールが持つネットワーク資源情報は定期的に収集した情報であり，本プロトタイプシステムではこの間隔を 10 秒単位にしている．取得したネットワーク資源情報および計算資源情報を基に Brain サブモジュールは資源割当ポリシに従って計算資源候補リストの計算ノードを並べ替え，修正したリストを OGS/GE に返す（図 1 (3)）．また，この修正処理において資源割当

ポリシに適さない計算ノードが含まれていれば，その計算ノードを計算資源候補リストから削除することもできる．OGS/GE は修正された計算資源候補リストに基づきジョブのプロセスを割り当てる計算ノードを決定し，確定した割当計算ノードのリストを Brain サブモジュールに渡す（図 1（4））．Brain サブモジュールは受け取った割当計算ノードで使用する *flow entry* 情報を Network Control サブモジュールに渡す（図 1（5））．その際，*flow entry* 情報は該当ジョブのジョブ ID と関連付けてデータベースにも保存される．最後に，OGS/GE は計算資源にジョブの各プロセスを割り当て，Network Control サブモジュールは OpenFlow スイッチに *flow entry* 情報を登録して計算資源の割当処理を終了する（図 1（6））．

4.3 資源割当ポリシ

本プロトタイプシステムで実装したネットワーク資源割当ポリシには，割当計算資源のネットワークにおける遅延考慮（hop）と帯域考慮（bandwidth）の 2 種類を用意した．この資源割当ポリシは qsub コマンドにおける拡張資源要求オプションの netprio で指定することで有効化することができる（図 1（a））．

遅延考慮を要求した場合，提案 JMS は割当可能な計算ノードに対し，遅延が最小となる計算ノードの組み合わせを算出する．本プロトタイプシステムでは遅延を計算ノード間のホップ数で表し，割り当てる計算ノード間の平均ホップ数が最小となる計算ノードの組み合わせをジョブに割り当てる．帯域考慮を要求した場合には，割り当てる計算ノード間の帯域が最大となる組み合わせを計算資源候補リストから探す．割り当てる計算ノードの組み合わせの算出は，各計算ノード間におけるダイクストラ法による最短経路問題を遅延考慮の場合は計算ノード間のホップ数，帯域考慮の場合は計算ノード間の利用可能帯域をコスト値として実行し，最小となる計算ノードの組み合わせを求めてジョブに割り当てる．

5. 検証実験

ネットワーク資源を考慮した資源割当ポリシを指定した提案 JMS と従来の OGS/GE に対し，同一のジョブ列を投入することによってその資源割当の振る舞いと実効性能について検証実験を行う．検証実験を行う計算環境を**図 2** に示す．検証実験に使用するクラスタ型計算資源は，28 台の計算ノードで構成されており，1 台の OpenFlow スイッチに 7 台ずつ接続されている．計算ノードと接続している 4 台の OpenFlow スイッチは 1 台の上流 OpenFlow スイッチに接続されていて，計算ノードとスイッチおよびスイッチ間はすべて 1 Gbps で接続されている．検証実験では提案 JMS おける OpenFlow を利用したネットワーク資源考慮の効果を測定することを目的としているため，同一計算ノード内でのプロセス間通信が発生しないよう各計算ノードに割り当てられるプロセス数は 1 つに制限する．また，本評価に使用するジョブで実行するプログラムには，CPU 処理による処理時間差の影響が極力出さずにネットワーク利用のみを行うよう，任意のプロセスから他のすべてのプロセスにデータの転送を行う処理を繰り返すプログラムを使用する．なお，このプログラムで転送するデータのサイズには 8 KiB，16 KiB，32 KiB，64 KiB を用いる．

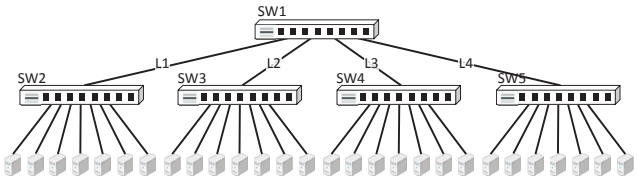
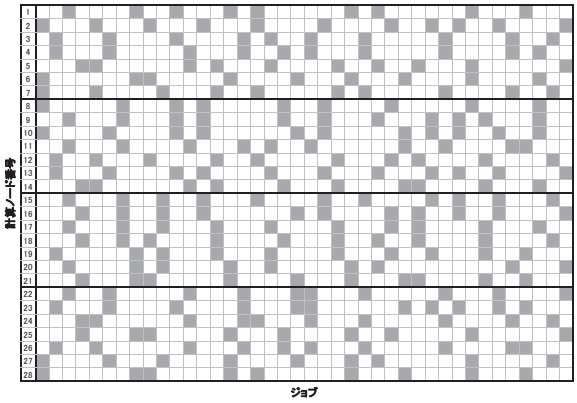


図 2 実験クラスタの構成

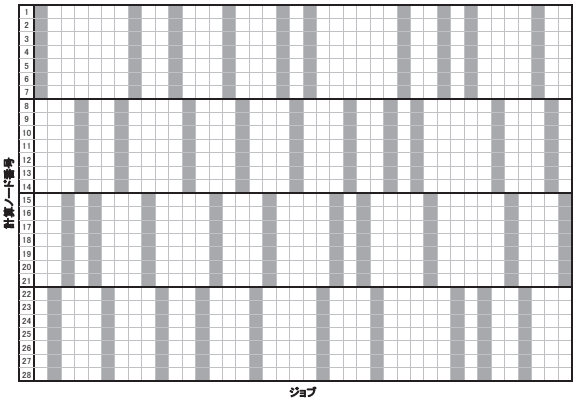
5.1 遅延考慮

資源割当ポリシに遅延考慮を指定した場合，本プロトタイプシステムでは計算ノード間のホップ数で評価するので，遅延が最小となるのは同一 OpenFlow スイッチに接続された計算ノード間となる．今回用いる計算環境では 1 つの OpenFlow スイッチに 7 台の計算ノードが接続されているので，ジョブの並列数が 7 並列までは，ジョブのすべてのプロセスが同一 OpenFlow スイッチに接続された計算ノードに割り当てられた状態が最小の平均ホップ数の割当計算資源となる．7 並列ジョブを連続して投入した場合，今回用いる計算環境は 28 台の計算ノードで構成されているため 4 つのジョブが同時に実行されるので，これを 1 ステップとして合計 25 ステップの試行を繰り返して検証を行った．この検証実験における計算資源の割当状況について，64 KiB で行った実験結果の一部を，縦軸に計算ノード，横軸にジョブを割当処理順に並べた図として**図 3** に示す．

本検証実験において，提案 JMS はネットワーク資源情報を利用して割当計算資源の平均ホップ数が最小となるよう選択しているため，**図 3(b)** に示されるようにすべてのジョブにおいて 1 つのジョブのプロセスが同じスイッチ配下に割り当てられた．一方，OGS/GE では**図 3(a)** に示されるように，各ジョブのプロセスの割当は複数のスイッチに接続された計算ノードに分散した．なお，本実験におけるすべての試行において，OGS/GE ではジョブのプロセスが同じスイッチ配下に割り当てられたジョブは 1 つも無かった．7 並列ジョブが同一スイッチに接続された計算ノードに割り当てられる組み合わせは 4 通りしかなく，この計算資源割当となる確率は 0.0003% であることからこの結果は妥当であると判断できる．以上より，提案 JMS における遅延を考慮した計算資源割当は，従来 JMS の CPU ロードアベレージに基づく計算資源割当に対してよりホップ数の少ない計算資源割当を実現できることが確認できた．



(a) OGS/GE における計算資源の割当状況 (64 KiB)



(b) 提案 JMS における計算資源の割当状況 (64 KiB)

図 3 遅延制御を指定した場合の計算資源の割当状況 (64 KiB)

次にジョブ処理性能の検証として，各データサイズにおけるジョブの平均処理時間を**図 4** に示す．図 4 において 8 KiB，16 KiB ではほとんど差は見られないのに対し，32 KiB，64 KiB では OGS/GE における処理時間の増加が確認された．この結果より，32 KiB，64 KiB で処理時間に差が出たのは，スイッチ間のリンク L1〜L4 を複数のジョブが同時に利用して輻輳が発生したためであると考える．一方，8 KiB，16 KiB でも 32 KiB，64 KiB の場合と同様に複数のジョブから転送したデータがスイッチ間のリンク L1〜L4 を同時に流れる状況は発生しているが，処理時間にはほとんど影響は出ていない．これは転送したデータのサイズが小さいため輻輳による通信性能の低下が生じなかったためと考える．

5.2 帯域考慮

次に，資源割当ポリシに帯域考慮を指定した場合の検証実験として，割り当てられた計算資源間の通信経路が他のジョブに利用されていないかを確認する．この確認のため，1 つのジョブが使用する通信経路が 1 つとなるよう 2 並列を要求するジョブを用いる．2 並列ジョブを連続して投入した場合，今回用いる計算環境では 14 個のジョブが

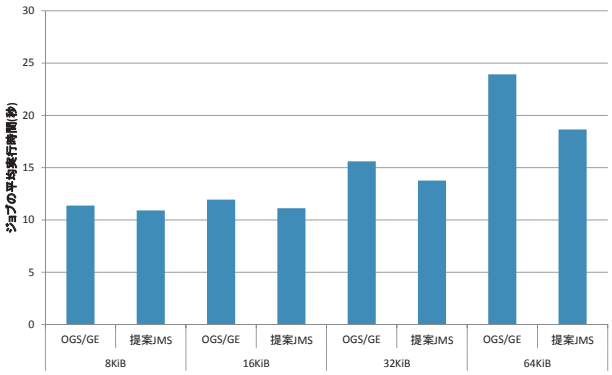


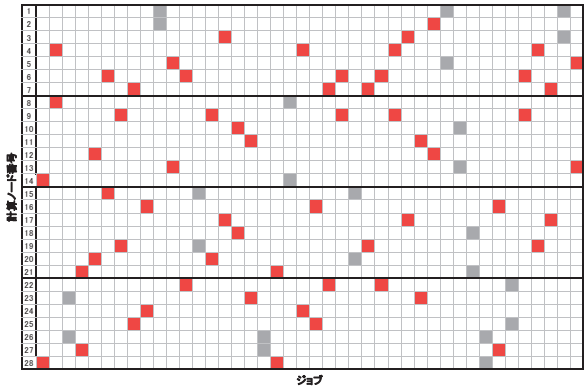
図 4 遅延制御を指定した場合の各データサイズにおけるジョブの平均処理時間

同時に実行できるため，これを 1 ステップとして合計 25 ステップの試行を繰り返して検証を行った．この検証実験における計算資源の割当状況について，64 KiB で行った実験結果の一部を，縦軸に計算ノード，横軸にジョブを割当処理順に並べた図として**図 5** に示す．なお，割り当てられた計算ノードが赤色で表示されているジョブは，スイッチ *SW1* を経由した通信を行うジョブであることを示す．

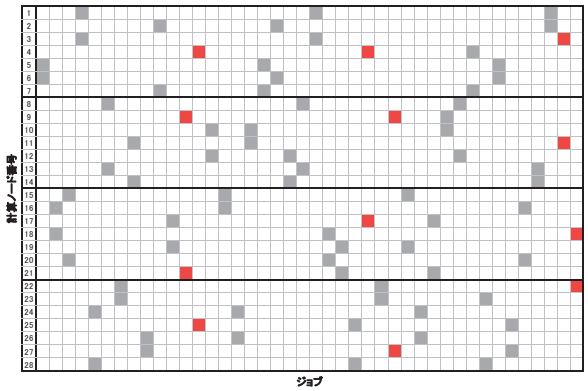
本検証実験において，通信経路の重複が無い状態とは，スイッチ *SW1* を経由する通信を行うプロセスが各スイッチ *SW2*〜*SW5* に 1 つずつ割り当てられた状態である．この場合，スイッチ間のリンク L1〜L4 のどのリンクも利用するジョブは 1 つとなる．提案 JMS では帯域の利用状況を監視して資源を割り当てているため，基本的には**図 5(b)** が示すように前述の計算資源割当の状態となったが，スイッチ間のリンクを複数のジョブが利用する状況が発生した．なお，この状態となった割合は全試行の 3.1% であった．これは先に割り当てられたジョブが通信を始める前に次のジョブの計算資源割当処理でネットワーク帯域情報を参照したため発生したと推測する．

一方，OGS/GE では**図 5(a)** に示されるように，スイッチ間のリンク L1〜L4 を重複して利用するジョブが多数発生した．どのデータサイズの試行においても，スイッチ間のリンク L1〜L4 を重複利用するジョブの平均数はリンクあたり約 5 個であり，その発生頻度は 1 試行あたり平均 10 個であった．本検証実験において，投入されたジョブが要求する帯域が理論最大値の 1/5 である 200Mbps までであれば，理論値上は大きな性能低下は発生しないと推測できる．以上より，提案 JMS における帯域を考慮した計算資源割当は，従来 JMS の計算資源割当と比較して帯域の効率的利用を実現できることが確認できた．

次にジョブ処理性能の検証として，各データサイズにおけるジョブの平均処理時間を**図 6** に示す．図 6 において，転送するデータサイズが大きくなるにつれジョブの平均処理時間に差が生じ，OGS/GE では処理性能が低下してい



(a) OGS/GE における計算資源の割当状況 (64 KiB)



(b) 提案 JMS における計算資源の割当状況 (64 KiB)

図 5 帯域制御を指定した場合の計算資源の割当状況 (64 KiB)

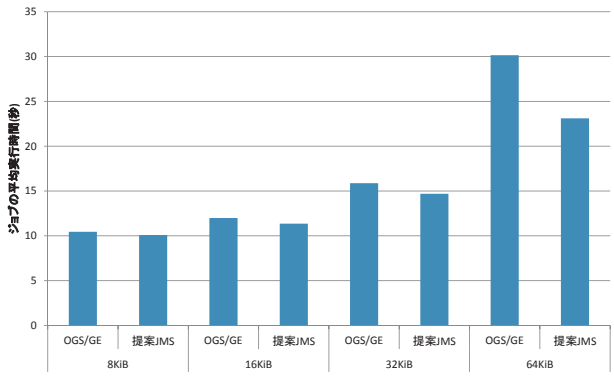


図 6 帯域制御を指定した場合の各データサイズにおけるジョブの平均処理時間

ることが確認できる．このことから，データサイズが小さい場合には使用する帯域が小さいため，同じリンクを使用するジョブが複数あっても処理時間に大きな影響を与えなかったのに対し，転送するデータサイズが大きくなるほど各リンクでの通信の衝突がジョブの処理性能に影響を与えていったと考えられる．

6. おわりに

本稿では，ユーザジョブに割り当てる計算資源を決定する際に，OpenFlow の機能を利用することによって従来の計算資源情報だけでなくネットワーク資源情報についても考慮することができるジョブ管理システムを提案した．その提案 JMS のプロトタイプシステムにおけるシステム構成および今回実装したネットワークの遅延または帯域を考慮したジョブへの計算資源割当機能について説明した．また，それぞれの計算資源割当ポリシを指定した場合における提案 JMS と従来の OSG/GE の計算資源の割当状況および処理性能を比較することで本提案システムの有効性について評価を行った．

今後の課題としては，並列数がランダムであるジョブ列や並列ベンチマークプログラム等のアプリケーションに対する評価を行う予定である．また，今回実装した資源割当ポリシにおける判定アルゴリズムや，異なる資源割当ポリシの検討を行い，実装を進める予定である．

謝辞 本研究は，独立行政法人情報通信研究機構 (NICT) の委託研究「仮想分散コンピューティング・データ流通技術の研究」の支援に基づき推進している．

参考文献

[1] TOP 500 Supercomputer Sites : http://top500.org/ .
[2] B. A. Kingsbury, “The Network Queuing System”, Sterling Software, Palo Alto, 1985.
[3] R. Henderson, “Job Scheduling under the Portable Batch System”, Job Scheduling Strategies for Parallel Processing, Springer, Vol. 949, pp. 279-294, 1995.
[4] The official Open Source Grid Engine : http://gridscheduler.sourceforge.net/ .
[5] A. Takefusa, M. Hayashi, N. Nagatsu, H. Nakada, T. Kudoh, T. Miyamoto, T. Otani, H. Tanaka, M. Suzuki, Y. Sameshima, W. Imajuku, M. Jinno, Y. Takigawa, S. Okamoto, Y. Tanaka, and S. Sekiguchi, “G-lambda: Coordination of A Grid Scheduler and Lambda Path Service over GMPLS”, Future Generation Computer Systems, Vol. 22, No. 8, pp. 868-875, Oct 2006.
[6] G. P. Koslovski, P. V.-B. Primet, and A. S. Charão, “VXDL:Virtual Resources and Interconnection Networks Description Language”, GridNets 2008, pp. 138-154, 2008.
[7] 吉富翔太，斎藤秀雄，田浦健次朗，近山隆，“自動取得したネットワークトポロジーに基づく MPI 集合通信”，情報処理学会研究報告 HPC-116 (SWoPP 2008)，pp. 7-12, 2009.
[8] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “OpenFlow: Enabling Innovation in Campus Networks”, SIGCOMM Computer Communication Review, Vol. 38, No. 2, pp. 69-74, Mar 2008.
[9] IEEE 802.1AB-2005 IEEE Standard for Local and Metropolitan Area Networks Station and Media Access Control Connectivity Discovery, May 2005.
[10] Trema: Full-Stack OpenFlow Framework for Ruby/C: http://trema.github.com/trema/ .