Grid 計算環境における 予定ガント図を用いたジョブスケジューリング

上田清詩[†,☆] 本多弘樹[†] 弓場敏嗣[†]

クライアント・サーバ型構成により，ユーザに Grid 計算環境を提供する GridRPC システムが複数提案されており，ユーザのジョブを適切なサーバに割り当てるスケジューリング手法に関する議論が続いている．これらのシステムにおけるスケジューラが用いるスケジューリング手法の多くは，ジョブの所要時間を予測し，その値を基にジョブを割り当てるサーバを決定する．したがって，適切なサーバを選択するには所要時間の予測精度を向上させることが重要となる．既存の GridRPC システムにおけるスケジューリング手法で利用されている所要時間算出手法では，Grid 計算環境上の各資源から得られるパフォーマンスの変動の予測に際し，変動が生じる時点を考慮していない．本論文では，Grid 計算環境上の各資源から得ることのできるパフォーマンスの変動が生じる時刻をガント図を用いることにより求め，所要時間の予測精度を向上させる所要時間算出手法を提案する．提案手法を実装し，所要時間が最短となるサーバを選択するスケジューリング手法において，従来手法と提案手法の比較を実機を用いた評価環境で行った結果，従来手法に比べて最大で約 14%所要時間が短縮された．

A Job Scheduling Using Look-ahead Gantt-Chart for Computational Grids

Kiyoshi Ueda,[†,☆] Hiroki Honda[†] and Toshitsugu Yuba[†]

GridRPC systems have been proposed to provide users with resources on the Computational Grid. These systems are implemented as client-server architecture, and the problem of scheduling strategies for allocating resources appropriately in these systems is still open. Because most of the scheduling strategies in these systems select a server based on the predicted processing time of the job, it's important for improving its accuracy. In predicting the resource performance which varies dynamically, the prediction schemes used by existing GridRPC systems do not consider when the changes occur. In this paper, we propose a new resource performance prediction scheme which considers it using the Look-ahead Gantt-Chart. On a pseudo Computational Grid built on actual machines, we compared the processing time of jobs with the conventional prediction schemes. As a result, proposed prediction scheme reduce it by approximately 14%.

1. はじめに

近年，地理的に分散した多数の計算資源を広域ネットワークで接続した Grid 計算環境が注目を集めており，この計算環境をエンドユーザに提供する Grid 計算システムの研究や開発がさかんに行われている[1)~3)]．Grid 計算システムには，クライアントからサーバに対してジョブの実行を依頼するクライアント・サーバ型アーキテクチャを持ち，RPC ベースのプログラミングモデルを提供する GridRPC システムがある．GridRPC システムとしては，Ninf[4)] や NetSolve[5)] などが提案されている．

これらのシステムが提供する計算資源やネットワーク資源は，広域ネットワーク上に分散する多数のユーザによって共有される．そのため，各資源では，その資源を同時に使用するジョブ数（共有ジョブ数）の増減により，各ジョブが得ることのできる計算能力や通信スループットといったパフォーマンスが変動することが予想される．このように各資源で複数のジョブが競合する環境下で，ユーザから発行されるジョブを適切な資源に割り当てる必要があるため，ジョブのスケジューリングが重要な課題となる．

既存の GridRPC システムは，一般にクライアント，サーバ，スケジューラから構成され，以下の手順でジョブを実行する．

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
☆ 現在，NEC ソリューションズ
Presently with NEC Solutions

(1) クライアントはジョブを発行すべきサーバをスケジューラに問い合わせる．
(2) スケジューラは適切なサーバを選択してクライアントに紹介する．
(3) クライアントは紹介されたサーバにジョブを発行する．
(4) サーバはジョブを処理し，クライアントに結果を返す．

これらのシステムのスケジューラが用いるスケジューリング手法の多くは，クライアントがサーバにジョブを発行してから，当該クライアントに結果が戻ってくるまでに要する時間（以下，所要時間）の予測値を基に割り当てる資源を決定する．たとえば，所要時間が最短となるサーバにジョブを割り当てるという手法であれば，各サーバにジョブを割り当てた場合の所要時間を予測し，それが最短となるサーバを選択する．この場合，所要時間の予測精度が低いと，実際には所要時間が最短とはならないサーバにジョブを割り当ててしまう可能性が大きくなる．したがって，この種のスケジューリング手法の精度を向上させるには，問合せがあった時点における，当該ジョブの所要時間の予測精度を向上させることが重要である．

Ninf や NetSolve など既存の GridRPC システムで用いられる所要時間算出手法〔以下，この手法を TIPO（time point）と呼ぶ〕では，問合せのあったジョブ a の所要時間算出に際して，サーバから得ることのできる計算能力，および，クライアント・サーバ間の通信スループットに関して，以下のように将来の変動の予測を行っている．

すなわち，ジョブ a の処理中には，サーバにこれまで割り当てられ，かつ，終了していないすべて（ジョブ a の所要時間算出時点で実行開始されていないものも含め）のジョブの処理が実行中であると予測する．一方ジョブの終了に関しては，実際にジョブの実行が終了したときの通知によって知るのみで，スケジューリング時にその予測は行わない．

通信スループットの予測についても同様である．

しかしながら，このような手法では，サーバから得ることのできる計算能力が減少する時刻や，あるジョブの実行が完了してサーバから得ることのできる計算能力が増大するという予測，および，その時刻，といった時間軸に沿った変動についての予測が行われていない．

これに対し本論文では，問合せのあったジョブが各資源から得ることのできるパフォーマンスの時間的推移を，ガント図を用いてスケジューリング時点で求めることにより，予測精度を向上させる所要時間算出手法「**SKED**」を提案し，その実装，評価について述べる．

評価では，ソフトウェアシミュレータよりも，より実環境に近い，実機による評価環境を用いた．この評価環境は，帯域制御技術との組合せにより，擬似的な Grid 計算環境を作り出せる．この環境を用いて TIPO と SKED を比較し，SKED の有効性を示す．

以後，2 章では SKED について説明し，3 章では実装について述べ，4 章でその評価結果を示す．

2. 所要時間算出手法（SKED）

2.1 SKED の概要

SKED とは，問合せのあったジョブが各資源から得ることのできるパフォーマンスの時間的推移を，ガント図を用いてスケジューリング時点で求めることにより，当該ジョブの所要時間を算出する手法である．つまり，TIPO で行われていなかった，各資源から得ることのできるパフォーマンスが変動する時刻と，変動後の値の予測を取り入れたうえで，ジョブの所要時間を予測する手法である．ただし，SKED で考慮するのは，問合せがあった時点までに，資源の割当てが済んでいる他のジョブとの競合によってもたらされる変動のみであり，将来発生するジョブや，外乱によってもたらされる変動ついては考慮しない．つまり，スケジューリング時に確定している情報のみから予測可能な変動だけを考慮する☆．本論文では，SKED が対象としているこのような予測を**予定**と呼ぶ．

Grid 計算環境の資源としては，計算サーバと広域ネットワークが考えられる．計算サーバにおいて競合するジョブは，スケジューリング履歴から取得可能と考えられるが，広域ネットワークにおいて競合するジョブの検出は困難である．SKED はそれを容易にするために，広域ネットワークの通信経路（以後，パスと呼ぶ）上に存在する個々のリンク（隣接する 2 つのノード間を接続する物理ネットワーク）を，それぞれ個別の資源と見なす．また，SKED は後述する資源モデルを利用して，ある資源で競合している各ジョブが，その時点でその資源から得ることのできるパフォーマンスを計算する．これらにより，各ジョブが資源から得ることのできるパフォーマンスの変動予定を考慮した所要時間を算出する．

図を用いて説明する．**図 1** は，3 つのサイト（*Site*1,

☆ したがって，将来ジョブが発生したり，外乱によってその時点で予測できなかった変動が生じたりした場合には，その時点で算出していた予測所要時間どおりには終了しなくなる．

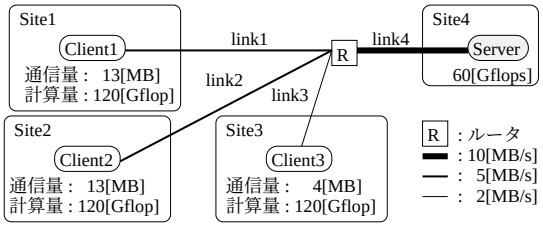

図 1 Grid 計算の例
Fig. 1 Example of computation on the computational grid.

*Site*2，*Site*3）にそれぞれ 1 つずつクライアント（*Client*1，*Client*2，*Client*3）が属しており，3 つのクライアントが *Site*4 に属するサーバ（*Server*）へ同時にジョブを発行した場合を表している．各クライアントが発行したジョブをそれぞれ *job*1，*job*2，*job*3 とし，各ジョブの通信量はそれぞれ 13 [MB]，13 [MB]，4 [MB] で計算量はすべて 120 [Gflop]，*Server* の性能が 60 [Gflops] であると仮定している．従来は広域ネットワークを end-end，あるいは各サイト間の通信を提供するネットワーク資源としか見なしていなかったため，3 つのジョブの通信が *link*4 で競合していることを検出するのが困難であった．しかし，各リンクを個別の資源と見なすことで，ネットワーク資源で競合するジョブの検出が容易となる．また，モデルによって競合を考慮した各ジョブの通信スループットを計算できるため，仮に *job*3 の通信が最初に終了した場合，他のジョブの通信スループットがどのように変動するかが算出可能となる．計算サーバにおいても競合するジョブが検出可能で，かつ，モデルによって各ジョブが得ることのできる計算能力を計算できる．このため，SKED は各ジョブが各資源から得ることのできるパフォーマンスの変動予定を考慮することが可能となる．

2.2 資源のモデル化

2.2.1 資源モデルの概要

ジョブの所要時間は，そのジョブが各資源から得ることのできるパフォーマンスの時間的推移に依存する．全資源を 1 つの GridRPC システムによって占有できたと仮定すると，あるジョブが各資源から得ることのできるパフォーマンスは，資源の最大性能とその資源を共有しているジョブ数に依存すると考えられる．本論文では，各ジョブが資源から得ることのできるパフォーマンスは，資源の最大性能とその資源の共有ジョブ数のみによって決定される，という資源モデルを採用する．資源モデルは，ネットワーク資源をモデル化した広域ネットワークモデルと，計算資源をモデル化したサーバモデルに分けられる．

2.2.2 広域ネットワークモデル

あるジョブの通信に使用中のパス上に複数のリンクが存在する場合，当該通信では，パス上の全リンクをボトルネックリンク☆から得られる通信スループット分だけ使用するとモデル化する．

また，複数のジョブが 1 つのリンクを同時に使用する場合，各ジョブはリンクの最大帯域幅を等分して使用する．ただし，パス上の他のリンクがボトルネックとなり，等分した帯域を使いきれないジョブが存在する場合，より大きな帯域を要求する他のジョブが余った帯域をさらに等分して使用するとモデル化する．

また，上記広域ネットワークモデル上でのジョブの通信時間 t [sec] は，ジョブの通信量 D [MB] と，そのジョブがボトルネックリンクから得ることのできる通信スループット T [MB/s] を用い，次式で定義する．

$$t = \frac{D}{T(0:t)}$$

ただし，$T(0:t)$ は，時刻 0〜t [sec] の間にそのジョブがボトルネックリンクから得ることのできる通信スループットである．また，そのジョブがボトルネックリンクから得ることのできる通信スループットが，通信開始から n $(n \geq 0)$ 回変動し，それぞれの変動時刻が t_n [sec] の場合は，次式で定義する．ただし，$t_k = 0$ $(k \leq 0)$ とする．

$$t = t_n + \frac{D - \sum_{k=0}^{n} \{(t_k - t_{k-1}) * T(t_{k-1} : t_k)\}}{T(t_n : t)}$$

つまり，n 回目の通信スループットの変動前までに転送されたデータ量を元の通信量から差し引き，残りが $T(t_n : t)$ [MB/s] で転送される．

2.2.3 サーバモデル

ある計算サーバの最大性能を P [Gflops]，その計算サーバを共有しているジョブ数を n とした場合，各ジョブがその計算サーバから得ることのできる計算能力 M [Gflops] を以下のようにモデル化する．

$$M = \frac{P}{n}$$

また，ジョブの計算量 C [Gflop] が既知の場合の，サーバモデル上でのそのジョブの計算時間 t [sec] を，通信時間と同様に次式で定義する．ただし，n $(n \geq 0)$ はそのジョブがその計算サーバから得ることのできる計算能力が，計算開始から変動する回数，t_n [sec] はそのときの各変動時刻であり，$t_k = 0$ $(k \leq 0)$ とする．また，$M(a:b)$ は，時刻 a〜b [sec] の間にその

☆ あるパス上のリンクの中で，あるジョブが得ることのできる通信スループットがそのパス上の他のどのリンクよりも小さなリンク．

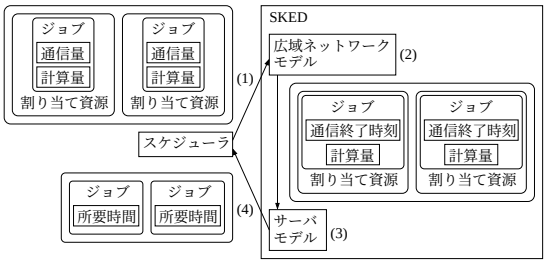

図 2 SKED の動作手順
Fig. 2 Flow of SKED.

ジョブがその計算サーバから得ることのできる計算能力である．

$$t = t_n + \frac{C - \sum_{k=0}^{n} \{(t_k - t_{k-1}) * M(t_{k-1} : t_k)\}}{M(t_n : t)}$$

2.3 SKED の動作手順

SKED は提案した資源モデルを利用するため，以下のような前提条件を必要とする．

(1) ジョブの通信量と計算量が既知
(2) ジョブに割り当てる資源（ネットワーク資源を含む）が既知
(3) 各リンクの最大帯域幅が既知
(4) 各計算資源の最大性能が既知

SKED は以下に示す手順で所要時間を算出する．以下の番号は**図 2** に対応している．なお，図 2 中で入出力のないモジュールは受け渡すデータを表す．

(1) 各ジョブの通信量と計算量の残量，および，割り当てる資源を入力として受け取る．
(2) 各ジョブの通信量を広域ネットワークモデルで処理し，各ジョブの通信終了時刻を算出する．具体的には，以下の手順を全ジョブの通信が終了するまで繰り返す．
  (a) 各リンクの共有ジョブ数より，個々のジョブが得ることのできる通信スループットを算出．
  (b) 各ジョブの通信量と (2a) の値より，最初に通信を終了するジョブ，および，その終了時刻を算出．
  (c) (2b) のジョブが使用していた各リンクの共有ジョブ数を減らし，各ジョブがそれまでに転送できるデータ量を通信量から削減．
(3) 算出された通信終了時刻に投入される各ジョブの計算量をサーバモデルで処理し，各ジョブの計算終了時刻を算出する．
  (a) 各サーバの共有ジョブ数より，個々のジョブが得ることのできる計算能力を算出．
  (b) 各ジョブの計算量と (3a) の値，および (2) で算出される通信終了時刻より，最初に共有ジョブ数が変化する時刻を算出．
  (c) (3b) のジョブが使用していたサーバの共有ジョブ数を減らし，それまでに処理される各ジョブの計算量を削減．
(4) 各ジョブの計算終了時刻と現時刻の差をとり，各ジョブの残り所要時間として出力する．

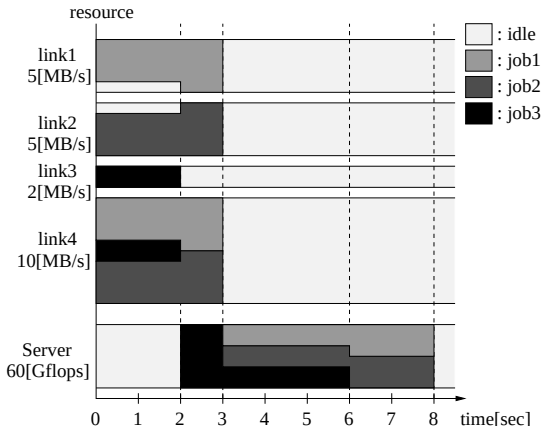

図 3 図 1 に対応する予定ガント図
Fig. 3 Look-ahead Gantt-Chart for Fig. 1.

図 1 の状況をこの手順で処理し，その処理過程をガント図で表現した例を**図 3** に示す．縦軸に各資源，横軸に時間をとり，各資源の縦幅はその資源の最大性能を表す．各ジョブが資源から得ることのできるパフォーマンスを各資源のガント図に占める縦幅で表現し，その資源を使用する時間を横幅で表現する．手順 (2b) では，各リンクのガント図を作成していることになる．手順 (3b) では，各リンクのガント図から導かれる各ジョブの通信終了時刻，および，各ジョブの計算量から，*Server* のガント図を作成していることになる．

このようにして作成されたガント図は，各ジョブが各資源から得ることのできるパフォーマンスの変動予定，および，その変動予定を考慮して算出される各ジョブの残りの所要時間を情報として保持している．本研究では，このように予定を表現するガント図を**予定ガント図**と呼び，特にリンクを資源とした予定ガント図を**ネットワークガント図**，サーバを資源とした予定ガント図を**サーバガント図**と呼ぶ．

3. SKED 評価システム

3.1 システム構成

図 4 に示す評価用の GridRPC システムを作成し，このシステム上に SKED を実装した．このシステム構

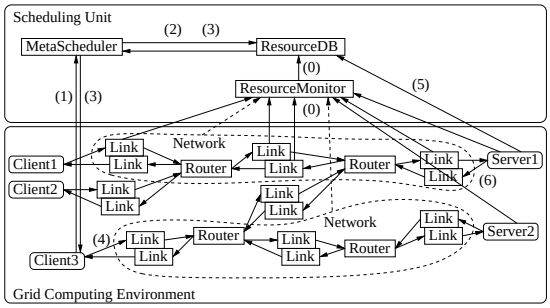

図 4 SKED 評価システムの構成
Fig. 4 Configuration of SKED evaluation system.

成は，一般的な Grid 計算システムにおけるスケジューリング手法の評価基盤を提供する離散シミュレータ Bricks[6] を参考にしている．スケジューリングユニットの各モジュールには以下の機能を持たせ，ResourceMonitor には Network Weather Service（NWS）[7] を用いた．

ResourceMonitor Grid 計算環境における Network（SKED を使用する場合は各 Link）の通信スループットとレイテンシ，および，各 Server の性能や負荷などをモニタする．得られた情報は ResourceDB に通知する．

ResourceDB ResourceMonitor からの資源情報に加え，Grid 計算環境の全トポロジや任意の end-end またはサイト間のパス，および，割当て済みジョブの残りの通信量や計算量といった処理進行状況など，SKED に必要な全情報が格納され，MetaScheduler に対してその情報を提供する．

MetaScheduler ResourceDB で管理されている情報を基に，ジョブを最適な Server に割り当てる．

3.2 動作手順

SKED 評価システムの動作手順を以下に示す．以下の番号は，図 4 に対応している．

(0) ResourceMonitor は定期的に Network または Link にプローブパケットを流して通信スループットやレイテンシを測定するとともに，Server に問い合わせて Server の負荷情報を調査する．結果は ResourceDB に通知する．

(1) Client はジョブを発行すると，MetaScheduler にジョブを投入すべきサーバを問い合わせる．その際，Client は MetaScheduler にジョブの情報と，Grid 計算環境での自分の位置情報を提供する．

(2) MetaScheduler は評価システム上でそのジョブを処理可能なすべての Server の情報，Client-Server 間の Network 情報，あるいはパスとパス上の全 Link 情報，および，割当て済みジョブの処理進行状況を ResourceDB に問い合わせ，結果を得る．

(3) MetaScheduler はこれらの情報を基にジョブを割り当てる Server を決定し，Client に通知する．このとき，そのジョブを一意に識別するためのジョブ識別子を決定し，同時に通知する．また，決定したスケジューリング情報を ResourceDB に通知する．

(4) Client は通知された Server にネットワーク経由でジョブを発行し，同時にそのジョブのジョブ識別子を通知する．

(5) Server はそのジョブの通信が終了した際，その旨をジョブ識別子を利用して ResourceDB に通知し，計算を開始する．また，計算が終了した際にもジョブ識別子を利用して ResourceDB にその旨を通知する．

(6) Server は計算結果をネットワーク経由で Client に返す．

SKED は，現時点で発行されているジョブと，その通信量や計算量の残量を基に予定ガント図を作成するため，各ジョブの処理がどこまで進んでいるかを正確に把握することが予測精度の向上につながる．実環境においては，提案モデルで考慮されていない資源の動作や外乱などにより，各ジョブが資源から得ることのできるパフォーマンスが変動するために予定がずれやすい．このずれを敏感に察知し，予定ガント図と実際の進行状況のずれを修正しやすくするために，Bricks 上にはない手順 (5) を追加した．この手順によってネットワークの共有ジョブ数減少が即座に察知できるため，SKED に限らず，この手順の追加は所要時間予測精度の向上に貢献すると考えられる．

3.3 ネットワーク上の外乱への対策

SKED は資源の最大性能を基に予定ガント図を作成する．外乱があると評価システムで利用できる最大性能がつねに変動するため，予定がずれて予測精度が低下する可能性が高いが，その値を取得できれば予測精度低下を抑えることが可能である．そこで NWS を用い，以下の手順によってネットワーク上の外乱から受ける予測精度低下の影響を抑える実装を施した．

(1) ジョブが発行されていない間，スケジューラは各リンクから得られる通信スループットを NWS より取得する．新たな値が得られたら (2) へ移る．ジョブが発行された場合は (3) へ移る．

(2) その時点で設定されているリンク i の最大性能を W_i，新たに計測されたリンク i の通信ス

ループットを T_i とし，W_i を $W_i*0.8+T_i*0.2$ によって更新する．これは，TCP/IP のタイムアウトタイマの更新式を参考にしている．ジョブが発行されていなければ (1) へ戻り，発行されていれば (3) へ移る．

(3) 通信スループットの取得を中断し，その時点で設定されている各リンクの最大性能を SKED に渡す．全ジョブの処理が終了したら (1) へ戻る．

ジョブが発行されていない間だけしか通信スループットの取得を行わないのは，ジョブの通信によるスループット低下の影響を，SKED に渡す各リンクの最大性能に反映しないようにするためである．

4. SKED の評価

4.1 比較対象と評価用スケジューリング手法

評価用のスケジューリング手法には，所要時間が最短となるサーバを割り当てるという手法を用いる．

所要時間算出手法の比較対象は TIPO である．サイト i に属するクライアントから発行された通信量 J_t [MB]，計算量 J_c [Gflop] のジョブを，サイト j に属するサーバ m に割り当てる場合，TIPO は以下の式により所要時間 t_m [sec] を算出すると定義する．

$$t_m = \frac{J_t}{\frac{T_{i,j}}{N_{i,j}+1}} + \frac{J_c}{\frac{P_m}{N_m+1}}$$

ただし，$T_{i,j}$，$N_{i,j}$，P_m，N_m はそれぞれ問合せがあった時点における，サイト i，j 間の通信スループット，サイト i，j 間のネットワークの共有ジョブ数，サーバ m の最大性能，サーバ m に割当て済みのジョブ数である．共有ジョブ数に 1 を加えているのは，そのジョブが投入された場合を考慮したものである．

4.2 ジョブとして使用する問題

偏微分方程式の数値解を，n 次の連立一次方程式を用いて近似的に求める問題をジョブとして使用し，連立一次方程式の解法には LU 分解法を用いた．クライアント側では $n \times n$ 行列と n ベクトルの初期化を行い，それらをサーバに転送する．サーバ側では LU 分解法を用いて求解し，結果をクライアントに転送する．

クライアントからサーバへの通信量，および計算量のオーダは，Linpack ベンチマークのそれと同様にそれぞれ $O(n^2)$，$O(n^3)$ であるが，サーバからクライアントへの通信量のオーダは，1 つの数値解であるため $O(1)$ である．現時点の SKED の実装では計算結果を返す際の通信を考慮していないため，計算結果の通信量が大きいと SKED の予測精度は低下する．ただし，サーバガント図作成後，再度ネットワークガント図を作成すればよいため，実装は複雑になるが理論上はその通信に関しても考慮可能である．

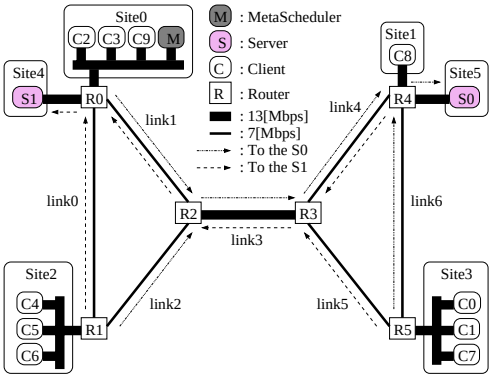

図 5 評価環境 (1)
Fig. 5 Experimental environment (1).

表 1 ノードの CPU 性能
Table 1 Node specifications.

ノード		CPU [MHz]	
Router	0〜4	Pentium III	667
	5	Pentium III	600
MetaScheduler		Pentium III	600
Server	0	Pentium III	1000
	1	Pentium II	333
Client	0〜7	SuperSPARC II	75
	8	Pentium II	266
	9	Pentium	120

4.3 評価環境

図 5 に示す環境を，実機を用いて構築した．各ノードの CPU 性能を**表 1** に示す．*Site*2，*Site*3 の各ノードは 10BASE-T 規格，それ以外のノードは 100BASE-TX/10BASE-T 規格の NIC（Network Interface Card）を搭載している．OS は *Client*0〜*Client*7 が Solaris2.6，それ以外はカーネル 2.2 系の Linux である．ルータはソフトウェアルータであり，ルーティングプロトコルは使用せず，スタティックにルーティングテーブルを設定してパスを決定している．また，帯域制御技術 TBF（Token Bucket Filter）を用いて各リンクの帯域幅を設定した．

NWS のネームサーバとメモリ（ディレクトリサーバ），および，SKED 評価システムの ResourceDB は MetaScheduler と同じマシン上で起動し，NWS のセンサは各ルータ上に起動した．さらに，各ルータ上では外乱受信プログラムを 1 つ，外乱送信プログラムを隣接するルータ数分だけ起動した．各外乱送信プログラムは指定した隣接ルータ上の外乱受信プログラムに対し，以下の仕様で UDP パケットを送信し続けることにより，各リンクの帯域を使用する．なお，外乱受

表 2 ジョブの平均所要時間（単位は [sec]）
Table 2 Processing time of jobs [sec].

Server	平均通信時間	平均計算時間	平均所要時間
Server0	12.5	17.5	30.0
Server1	12.5	44.6	57.1

表 3 各実験におけるケース（*link*3 に設定するパラメータ）
Table 3 Parameters of *link*3.

			実験 2 のみ	
case	A	B	C	D
静的な最大帯域幅 [Mbps]	13	13	1.0	6.5
外乱の平均使用帯域 [Mbps]	12	6.5	0	0

信プログラムは受信したパケットを破棄する．

パケット発行間隔 10 [msec]

パケットサイズ 帯域幅 W [Mbps] のリンクの x [%] を平均して占有する場合，平均値 $\frac{W}{100} * \frac{x}{100}$ [Mbit] の指数分布に従う．

この仕様は，外乱の変動を，各パケットのサイズを指数分布に従って変化させることで実現しようと考えたものである．後述するように，ジョブの平均所要時間は数十秒のオーダである．この仕様では，10 [msec] ごとに外乱が使用する帯域幅が変化するため，外乱の細かい変動を表現するのに妥当性があると考えられる．

*link*3 以外のリンクには，外乱送信プログラムにより，つねに平均 20 [%] の外乱を投入し，SKED 評価システムで使用できる平均帯域幅を 5.6 [Mbps] に設定した．

以上の環境において，4.2 節で述べたジョブを $n = 1024$ で *Client*0 が 1 回発行したときの平均所要時間を**表 2** に示す．

実験では，各クライアントが発行するジョブの問題サイズは $n = 1024$ とし，ジョブの発行間隔は，適度にジョブが発行される場合（実験 1）と頻繁に発行される場合（実験 2）を想定し，それぞれ平均値 446，223 の指数分布とした．発行間隔の平均値は，*Server*1 でのジョブの計算時間を 10 倍した値，そしてその半分の値である．表 2 より，図 5 の環境においてボトルネックとなるのは *Server*1 である．この場合，44.6 [sec] に 1 回ジョブが発行されれば，このシステムがジョブを処理できなくなることはない．したがって，クライアント数 10 を掛けた 446 [sec] を基準とした．

*link*3 をバックボーンとして接続されている 2 つの国家という想定の下，実験 1，実験 2 のそれぞれに対し，*link*3 を**表 3** のように設定して実験を行った．

本評価環境では，通信プロトコルとして，外乱は UDP を使用し，それ以外は TCP を使用している．これは，ジョブの通信などで使用可能な平均帯域が，TCP の輻輳制御によって，(静的な最大帯域幅) − (外乱の平均使用帯域) となることを期待したためである．つまり，我々は以下のモデルを採用した．

- UDP は輻輳制御を行わないため，パケットロスの有無にかかわらず，指定したレートでパケットを送出し続けることが可能である．
- 一方 TCP は，帯域が圧迫されてパケットロスが生じた場合，輻輳制御によりパケット送出を制限する．
- これにより，外乱（UDP）の平均使用帯域を設定することにより，ジョブの通信（TCP）で使用可能な平均帯域を制御することが可能となる．

このモデルの実環境における妥当性の検証は行っていないが，問題を単純化するために，今回はこのような単純なモデルを採用した．

ここで，SKED 評価システムの ResourceDB に手動で設定するネットワーク情報は，ネットワークトポロジと各ノード間のパス，および，全リンクの上り，下りの静的な最大帯域幅のみであり，実際に使用可能な帯域は NWS を用いて動的に取得している．

それぞれのケースは以下の想定に基づく．なお，caseC，caseD についてはさらに，両国内，および両国間に Grid 計算環境専用回線が施設されている，あるいは帯域制御技術により帯域確保が可能という想定を加え，*link*3 以外は静的に 5.6 [Mbps] に設定し，外乱は投入しない．

caseA：バックボーンが外乱によって飽和寸前
caseB：バックボーンが強く外乱の影響を受けにくい
caseC：バックボーンのコストが高く広帯域確保不可
caseD：バックボーンで広帯域を確保できる

また，*link*0，*link*6 が存在しない環境（評価環境 (2)）に対しても以上と同様の実験を行った．この場合，*Site*2 から *Site*4 へのパスは *link*2 → *link*1，*Site*3 から *Site*5 へのパスは *link*5 → *link*4 となる．

1 回の実験に対し，クライアントからの総発行ジョブ数が 100 に達するまで実験を行ってジョブの所要時間を採取し，それを異なる乱数の seed を用いて 10 回繰り返した．その平均値を実験結果とした．

4.4 評価結果

評価環境 (1) の実験結果を**図 6**，**図 7** に，評価環境 (2) の実験結果を**図 8**，**図 9** に示し，各実験で *Server*0 に割り当てられたジョブの割合を**表 4** に示す．

同じ {ネットワークトポロジ (評価環境)，ジョブ発行頻度 (実験)，バックボーンの状態 (ケース)} で SKED，TIPO を用いた場合の平均所要時間をそれぞれ s [sec]，t [sec] とし，SKED を用いたことによる TIPO に対

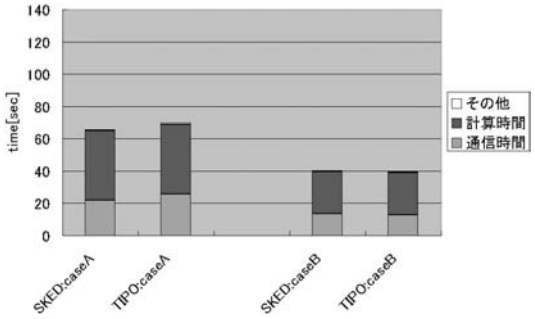

図 6 評価環境 (1) における実験 1（適度）の平均所要時間
Fig. 6 Results of experiment (1) on Experimental environment (1).

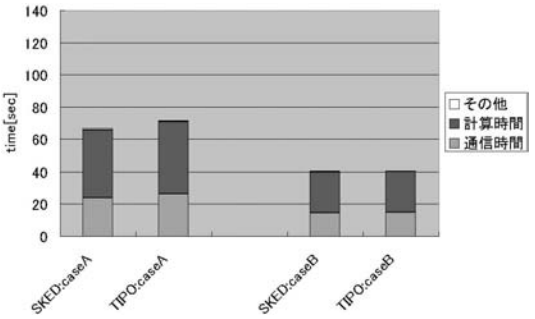

図 8 評価環境 (2) における実験 1（適度）の平均所要時間
Fig. 8 Results of experiment (1) on Experimental environment (2).

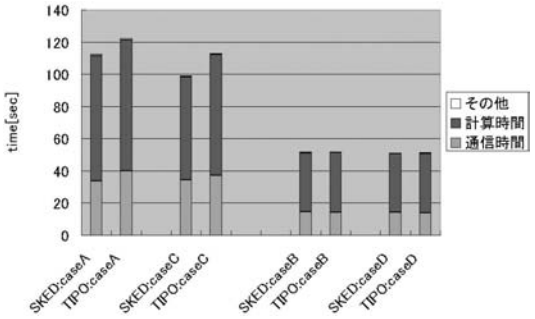

図 7 評価環境 (1) における実験 2（頻繁）の平均所要時間
Fig. 7 Results of experiment (2) on Experimental environment (1).

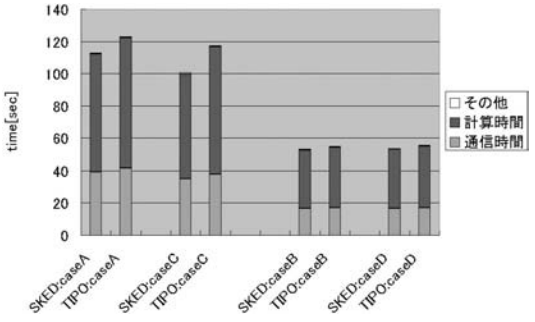

図 9 評価環境 (2) における実験 2（頻繁）の平均所要時間
Fig. 9 Results of experiment (2) on Experimental environment (2).

する性能向上率 α を次式で定義した場合の結果を**表 5** に示す．

$$\alpha = \frac{t-s}{t} * 100$$

同じ {ネットワークトポロジ (評価環境)，ジョブ発行頻度 (実験)} におけるバックボーンの状態（ケース）別では，評価システムで使用可能な平均帯域が広い場合（caseB，caseD）より，狭い場合（caseA，caseC）の方が性能向上率が高い傾向にある．使用可能帯域が狭いと各ジョブの所要時間が長くなることが多くなり，各資源においてジョブの競合が生じる可能性が高くなる．したがって，資源割当て時にそれを考慮する SKED の優位性が顕著に現れたものと考えられる．同様の理由により，同じ {ネットワークトポロジ (評価環境)，バックボーンの状態 (ケース)} におけるジョブ発行頻度（実験）別では，ジョブが適度に発行される場合（実験 1）より頻繁に発行される場合（実験 2）の方がジョブの競合が生じやすくなるために実験 2 の方が性能向上率が高く，同じ {ジョブ発行頻度 (実験)，バックボーンの状態 (ケース)} におけるネットワークトポロジ（評価環境）別では *link*1，*link*4 で競合が生じやすくなるため，評価環境 (2) の方が性能向上率が高くなる傾向にある．これらを表 4 で見ると，使用可能帯域が狭い場合は，*Server*0 に割り当てられたジョブの割合が SKED と TIPO 利用時でほぼ等しい．資源の利用率がほぼ等しいにもかかわらず性能向上が見られることから，SKED を利用することで，より適切なスケジューリングが行われていることが分かる．

次に，両ネットワークトポロジ（評価環境）でジョ

表 4 *Server*0 に割り当てられたジョブの割合（単位は [%]）
Table 4 Ratio of jobs assigned to *Server*0 [%].

	実験 1		実験 2			
case	A	B	A	B	C	D
評価環境 (1) SKED	49	94	54	86	58	86
TIPO	51	85	56	77	58	77
評価環境 (2) SKED	49	92	55	84	57	84
TIPO	51	85	55	77	58	77

表 5 性能向上率による比較（単位は [%]）
Table 5 Performance improvement ratio [%].

	実験 1		実験 2			
case	A	B	A	B	C	D
評価環境 (1)	5.6	−1.4	8.0	0.3	12.4	0.6
評価環境 (2)	6.7	0.8	8.2	1.9	14.3	3.5

ブが頻繁に発行される場合（実験 2）に限って見ると，外乱がある場合（caseA，caseB）より，ない場合（caseC，caseD）の方が性能向上率が高い．これは，外乱がある場合は評価システムで利用可能な資源の最大性能が，外乱がない場合と比較して大きく変動するため，SKED の予測精度が低下した結果だと考えられる．

このように，何らかの帯域制御技術によって外乱の影響を排除することが最も望ましいと思われるが，外乱がある場合でも性能向上率は低下するが劣るケースはほとんど見当たらないため，SKED の有効性が確認され，その利用価値は十分にあると考えられる．

5. 関 連 研 究

Grid 計算環境におけるスケジューリングの研究は，Grid 計算環境を 1 つのアプリケーションが独占して使用できることを前提とするか，複数のジョブによって共有されることを前提とするかによって，アプリケーションスケジューリングとジョブスケジューリングに大別されてきた．前述の Ninf や NetSolve などの GridRPC システムは，ジョブスケジューリングに属する．アプリケーションスケジューリングの例として，AppLeS [8]，Prophet [3] などのように，Grid 計算システム上でタスクスケジューリングを専門に行うシステムがある．これらは，非均質なハードウェア環境における単一アプリケーションのタスク分割や，分割したタスクの割当てを行うメカニズムを与え，Grid 計算環境における単一アプリケーションのパフォーマンスモデル，プログラミングモデルをユーザに提供している．

特に，AppLeS ではガント図を使用したスタティックなタスクスケジューリングを行っている．しかし，その目的は単一アプリケーションの最短実行パスを検出することであるため，単一資源に対して同時刻に複数タスクを割り当てることは考慮されていない．

本研究は，ジョブスケジューリングを対象としており，単一資源に対して複数のジョブを同時刻に割り当てた場合に生じる，ジョブ間での資源性能の奪い合いを考慮する．この状況下における各ジョブの処理時間を求め，スケジューリングに活用するためにガント図を使用している点が，AppLeS とは異なる．

6. まとめと今後の課題

本論文では，Grid 計算環境上の各資源から得ることのできるパフォーマンスの変動が生じる時刻をガント図を用いることにより求め，所要時間の予測精度を向上させるジョブ所要時間算出手法 SKED を提案し，実装，および評価を行った．

本手法は，従来資源と見なされていなかった広域ネットワーク中の個々のリンクを個別の資源と見なすとともに，各資源における共有ジョブ数の増減によって生じる，各ジョブが各資源から得ることのできるパフォーマンス変動の時間的推移を考慮する手法であり，パフォーマンス変動の時点までは予測しない従来手法と比較して，多くのジョブが競合する場合にその優位性が顕著に現れた．SKED は外乱により精度が低下するが，簡単な外乱対策を施すことによって精度低下を抑えることができ，外乱にも対応可能であることを示した．これらにより，SKED の現実的な有効性を確認した．また，帯域を確保し，外乱の影響をなくすことによって得られる利益が大きいことを示し，Grid 計算における帯域制御技術の利用を推奨した．

しかしながら，今回の評価結果が実環境において一般性を持つかどうかの確証が得られていないため，より多くの評価環境，ケースを想定した評価や，他のスケジューリング手法を用いた評価が必要と考えられる．また，LAN や WAN におけるトラフィックは自己相似性を持っていることが確認されており [9]，その特性を外乱として用い，評価することが必要である．同時に，サーバに外乱ジョブを投入し，より実環境を反映した評価環境での評価が望まれ，それに対する SKED の拡張も課題となる．帯域制御技術の利用に関しては，実環境では動的に帯域を確保する必要があるため，帯域確保までに要する時間の影響などを考慮しなければならない．

今後は，既存の GridRPC システムに SKED を実装し，評価環境はもとより，実環境における実験などを行うことにより，一般的な有効性について評価していくことが求められる．SKED を既存の GridRPC システムに実装し，実環境で利用するためには，以下のような様々な課題があげられる．

- ジョブの通信量と計算量の，より正確な見積り手法の確立
- ルータなどと連係し，ネットワークトポロジやパスを動的に取得する機能の追加
- 定期的に各リンクのスループットを計測できる機能の追加，または，NWS に限定せず，様々なスループット計測ツールへの対応
- 様々な特性を持った外乱への対応
- 通信と計算をオーバラップできるジョブへの対応
- 複数タスクに分割して並列実行が可能なジョブへの対応

これらをすべて解決するには多大な時間を要すると考えられるが，現時点の基礎研究の段階から実用レベルの研究に早く引き上げられるように，1つ1つ解決していきたい．

謝辞 本研究を進めるにあたり，数多くのご助言をいただいたお茶の水女子大学の竹房あつ子氏，産業技術総合研究所の中田秀基氏をはじめとする Ninf プロジェクトの皆様，ならびに日頃からご討論いただき，実験環境構築にご協力いただいた弓場・本多研究室の皆様に深く感謝いたします．

参 考 文 献

1) Berman, F., Wolski, R., Figueria, S., Schopf, J. and Shao, G.: Application-Level Scheduling on Distributed Heterogeneous Networks, *Proc. Supercomputing '96* (1996).
2) Raman, R., Livny, M. and Solomon, M.: Matchmaking: Distributed Resource Management for High Throughput Computing, *Proc. 7th IEEE International Symposium on High Performance Distributed Computing* (1998).
3) Weissman, J.B. and Zhao, X.: Scheduling Parallel Applications in Distributed Networks, *Journal of Cluster Computing* (1998).
4) 中田秀基, 竹房あつ子, 松岡 聡, 佐藤三久, 関口智嗣：グローバルコンピューティングのためのスケジューリングフレームワーク, 並列処理シンポジウム JSPP '99 論文集, Vol.99, No.6, pp.277–284 (1999).
5) Casanova, H. and Dongarra, J.: NetSolve: A Network Server for Solving Computational Science Problems, *Proc. Supercomputing '96* (1996).
6) 竹房あつ子, 合田憲人, 中田秀基, 松岡 聡, 長島雲兵：グローバルコンピューティングシミュレータの概要, 情報処理学会研究報告, Vol.99, No.21, pp.31–36 (1999).
7) NWS: Network Weather Service. http://nws.npaci.edu/NWS/
8) Casanova, H., Legrand, A., Zagorodnov, D. and Berman, F.: Heuristics for Scheduling Parameter Sweep applications in Grid environments, *Proc. 9th Heterogeneous Computing workshop* (*HCW '2000*), pp.349–363 (2000).
9) Sasaki, T., Kasahara, S. and Takahashi, Y.: A Generation Method of Pseudo Self-Similar Process with Wavelet Transformation, *Proc. 8th International Conference on Telecommunication Systems: Modelling and Analysis*, Nashville, TN, USA, pp.165–173 (2000).
10) 上田清詩，本多弘樹，弓場敏嗣：グローバルコンピューティングシステムにおけるネットワーク Gantt 図を用いたジョブスケジューリング手法の提案，情報処理学会研究報告，Vol.2000, No.93, pp.49–54 (2000).

(平成 14 年 6 月 6 日受付)
(平成 14 年 9 月 11 日採録)

上田 清詩

2000 年電気通信大学情報工学科卒業．2002 年同大学大学院情報システム学研究科修士課程修了．同年日本電気株式会社入社．在学時代にはグリッドの研究に従事．工学修士．

本多 弘樹（正会員）

1984 年早稲田大学理工学部電気工学科卒業．1991 年同大学大学院理工学研究科博士課程修了．1987 年より同大学情報科学研究教育センター助手．1991 年より山梨大学工学部電子情報工学科専任講師．1992 年より同助教授．1997 年より電気通信大学大学院情報システム学研究科助教授．並列処理方式，並列化コンパイラ，並列計算機アーキテクチャ，グリッド等の研究に従事．工学博士．電子情報通信学会，IEEE-CS，ACM 各会員．

弓場 敏嗣（正会員）

1966 年神戸大学大学院工学研究科修士課程修了．(株) 野村総合研究所を経て，1967 年通商産業省工業技術院電気試験所（現，産業技術総合研究所）に入所．以来，計算機のオペレーティングシステム，見出し探索アルゴリズム，データベースマシン，データ駆動型並列計算機等の研究に従事．その間，知能システム部長，情報アーキテクチャ部長等を歴任．1993 年より電気通信大学大学院情報システム学研究科教授．並列処理の科学技術一般に興味を持つ．工学博士．電子情報通信学会，日本ソフトウェア科学会，日本ロボット学会，ACM，IEEE-CS 各会員．