

## グリッド計算環境でのデッドラインを考慮したスケジューリング手法の性能

竹房あつ子<sup>†</sup> 松岡 聡<sup>††</sup>

Performance of a Deadline-Scheduling Scheme on the Computational Grids

Atsuko TAKEFUSA<sup>†</sup> and Satoshi MATSUOKA<sup>††</sup>

あらし

広域ネットワークに接続された計算資源を共通の目的を持つコミュニティで効率的に活用するグリッドが注目されている。グリッドシステムとして、ネットワーク上で計算資源とともにサービスの提供を可能にする Network-enabled Server (NES) が複数提案されている。NES は一般的にクライアント・サーバ型アーキテクチャであり、分散したグリッド計算資源上にサーバを用意する。一方、複数サーバ、複数のクライアントを想定したグリッドのスケジューリングに関する議論が十分に行われていない。また、将来の NES システム運用での課金に伴い、ユーザはジョブ実行時間を最短にすることから最小コストの資源群を利用して規定時間内に処理を終了させることを要求するようになる。本稿ではデッドラインスケジューリングに着目し、その性能特性をグリッドの評価用シミュレータ Bricks で調査した。まず、複数サーバ、複数クライアントを想定した、デッドラインスケジューリングアルゴリズムを紹介するとともに、その性能を高めるメカニズム、Load Correction と Fallback を提案する。次に、Bricks を用いた評価より、グリッド上での NES システムのデッドラインスケジューリングの有効性を示す。

キーワード グリッド、デッドライン、スケジューリング、性能評価、シミュレーション

### 1. はじめに

広域ネットワークに接続された計算資源を共通の目的を持つコミュニティで効率的に活用するグリッド計算技術の研究・開発が盛んに行われている [1] ~ [7]。グリッド上で計算資源とともにサービスの提供を可能にする Network-enabled Server (NES) [3], [7] ~ [9] は、クライアント・サーバ型アーキテクチャで RPC ベースのプログラミングモデルを提供する。科学技術計算では独立した多数のタスクからなる Parameter Sweep 型アプリケーションが多数あり [10], [11]、これらは NES の提供するプログラミングモデルと適合しやすい。

従来、グリッドのスケジューリング [12] では、並列アプリケーションの実行時間 (makespan) の短縮など、1 アプリケーションをいかに速く処理するかに重点がお

かれていた [13] ~ [17]。一方、NES の提供する計算資源群を同時に複数のユーザが利用する状況では、複数クライアントジョブのスケジューリングが重要な課題となる。広域ネットワーク上の特定多数のユーザに対して提供するグリッド上の計算資源はその性能や質が多岐に渡るため、将来的なグリッドシステムの運用における課金方法もそれに応じてバラエティに富むようになる。よって、ユーザは従来の計算機利用のように単純な経験則に基づいて最短実行時間を目標としてジョブを投入するのではなく、規定時間 (デッドライン) 内に処理を終える最小コストの計算機群を要求するようになる。既にこのような経済モデルの概念を利用したスケジューリング手法が複数提案されている [18] ~ [20] が、その有効性や利用可能性は明らかでない。

グリッド環境におけるデッドラインを考慮したスケジューリング (以後本稿ではデッドラインスケジューリングとする<sup>(注1)</sup>) は経済モデル下でのスケジューリング

<sup>†</sup> お茶の水女子大学理学部

〒112-8610 東京都文京区大塚 2-1-1

Faculty of Science, Ochanomizu University

Otsuka 2-1-1, Bunkyo-ku, Tokyo 112-8610, Japan

<sup>††</sup> 東京工業大学学術国際情報センター / 国立情報学研究所

〒152-8550 東京都目黒区大岡山 2-12-1

Tokyo Institute of Technology, GSIC / National Institute of Informatics

Ookayama 2-12-1, Meguro-ku, Tokyo 152-8550, Japan

(注1): グリッド環境では多様な組織・計算資源から成り、各ユーザの要求を満たしながら全体の資源稼働率を向上させることが重要である。よって、本稿で言及するデッドラインスケジューリングは厳密にリアルタイム性を要求するものではなく、一般のリアルタイムシステムで行われている応答時間保証や特定プロセスの QoS を目的とするものとは一線を画す。

手法の1つである。NESシステムのNimrod[21]では、性能や計算資源の需要の高さから計算資源に優先度をつける。はじめは優先度の低い資源にタスクを割り当て、各タスクの実行時間と計算資源との関係から計算資源に対する評価を改め、多数のタスクからなるアプリケーションの全体の実行時間がユーザの求めるデッドラインを超えないように次のタスクを割り当てていくデッドラインスケジューリングを実現している。

本稿では経済モデル下でのデッドラインスケジューリングに着目し、デッドラインスケジューリング手法の提案とその性能特性を調査した。評価では、グリッドユーザが発行するジョブにはあらかじめデッドラインが指定されているものとし、スケジューリングの失敗（指定されたデッドラインまでに計算が終了しない）率の低下を目的とした。また、評価ではBricks[22], [23]グリッドシミュレータを改良して用いた。

まず、柔軟かつ容易にスケラブルな評価環境を設定可能にするBricksのシミュレーションモデルについて説明する。次に、デッドラインスケジューリングの失敗率の軽減を目指す簡単なスケジューリングアルゴリズムを紹介するとともに、そのアルゴリズムの性能を高めるメカニズム、Load CorrectionとFallbackを提案する。Load Correctionはグリッドの資源情報をスケジューリング結果を用いて補正するものであり、Fallbackはサーバにジョブの入力データが到着した際にそのジョブがデッドラインを超えるかどうか予測し、超えると判断した場合は別のサーバにジョブを再投入させるメカニズムである。そして、計算資源の性能をその資源のコストとみなす簡単な経済モデルを適用し、特定多数のユーザがグリッドの計算資源群に対してシングルタスクのジョブを複数投入するモデルを想定した実験を行い、グリッドの負荷状況、計算資源コスト、性能予測の見積もり、および各スケジューリングメカニズムに対するデッドラインスケジューリングの性能を示す。

評価では、従来の実行時間を最短にすることを目的としたスケジューリング手法はデッドラインスケジューリングよりユーザのジョブが利用する計算資源のコストが高くなる、保守的にジョブ処理時間を見積もるとスケジューリングの失敗率が下がる一方資源コストが高くなる、Fallbackメカニズムは失敗率の軽減に非常に有効であることが明らかになった。

## 2. Bricks システムとその拡張

Bricksはグリッドのスケジューリング手法およびそ

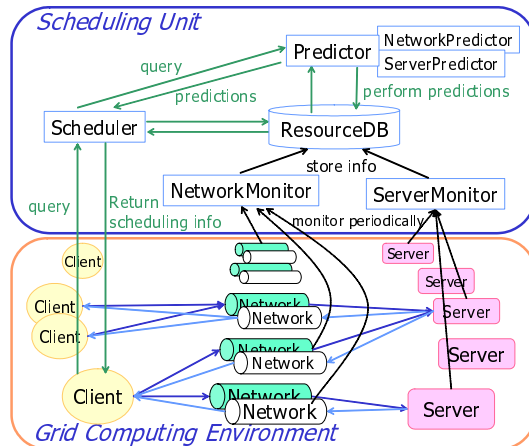


図1 Bricksシステムアーキテクチャ  
Fig.1 The Bricks Architecture.

のフレームワークの評価基盤を提供するJavaで実装された離散シミュレータである。図1にBricksのシステムアーキテクチャを示す。Bricksはグリッドをシミュレートするグリッド計算環境とスケジューリングユニットからなる。グリッド計算環境では、グリッドトポロジ、資源モデル（負荷トレース、待ち行列等）、クライアントモデル（ジョブ要求到着時間等）のシミュレーションコンポーネントセットを提供する。スケジューリングユニットでは、スケジューラ、プレディクタ、資源モナ、資源情報データベースなどグリッドでのスケジューリングに要するモジュール群を提供する[12]。BricksのユーザはBricksが提供する宣言的なスクリプトにより、柔軟に環境設定を行い、多様なスケジューリングアルゴリズムの再現性のある性能評価実験を行うことができる。また、スケジューリングユニットの各モジュールはユーザの実装したスケジューリングアルゴリズムや既存スケジューリングモジュールと容易に置換可能であり、その評価や機能試験をBricks上で実施できる[22]。

ただし、従来のBricksではスケラビリティが十分でない。よりスケラブルで現実的なグリッド環境での評価を可能にするため、Bricksシステムを改良し図2のような木構造のネットワークモデルを取り入れた。クライアント・サーバ間の各ネットワークは中間ノードとLAN、WANの異なる通信バンド幅のネットワークで構成される。この拡張により、ネットワークシミュレーションのための通信バンド幅およびトポロジを生成する既存システムGT-ITM[24], tiers[24], BRITE[25]等で生成されたネットワークトポロジが利用可能となる。

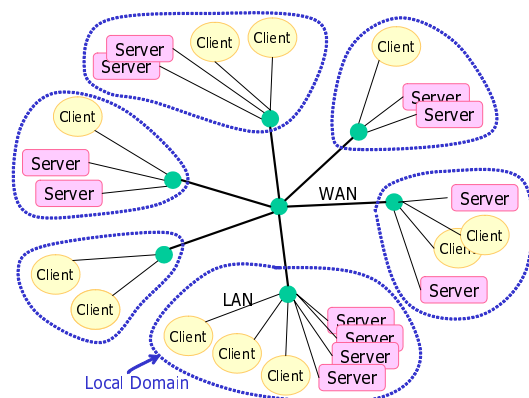


図2 ネットワークトポロジ  
Fig. 2 An Example of Network Topology.

### 2.1 グリッドシミュレーションモデル

Bricks では、グリッド上の計算資源（サーバ）と、クライアント・サーバ間のネットワークをそれぞれ待ち行列を用いて表現する。待ち行列で表現されるネットワーク、サーバモデルには以下の2つがある。

外部データ/ジョブモデル ネットワーク/サーバの混雑度をそのネットワーク/サーバに到着するグリッドシステムの利用者以外からの発行される外部データ/ジョブの流量により表す。

トレースモデル 実環境の通信スループット/サーバ負荷の観測値のトレースにより、ネットワーク/サーバの混雑度を表す。

外部データ/ジョブモデルは、パラメータの設定のみでシミュレーションが行えるものの、特にネットワークにおける通信スループット値の分散の調節が難しい。また、シミュレーションの粒度を細かくして精度を上げるとシミュレーションコストが非常に高くなってしまふ。一方、トレースモデルは実環境の変動をそのまま表現でき、シミュレーションコストが低いという利点があるが、事前にトレースデータを観測する必要があるため、シミュレーションで想定するネットワークトポロジが制限される。また、大規模評価環境の実験では大量のネットワーク/サーバのトレースデータを用意する必要がある。

本稿の評価実験では、サーバにはポアソン到着を用いた外部ジョブモデル、ネットワークにはトレースモデルを用いた。ネットワークのトラフィックモデルには、Paxsonにより提案された Fractional Gaussian Noise (FGN) による Self-Similar モデル [26] を採用した。トラフィックモデルでは一般にポアソン到着を想定す

ることが多いが、LAN および WAN でのトラフィックの性質はむしろ Self-Similarity があるといわれている [27]。シミュレーション環境下でより現実に近い性質をもつネットワークトラフィックを表現するためにこのモデルを採用した。これにより、少数のパラメータ設定のみで様々なトレース生成が可能になる。一方、高精度のサーバ負荷モデルはないため、サーバには外部ジョブモデルを用いてランダムに外部ジョブを生成させて負荷をシミュレートした。

### 3. デッドラインスケジューリング

既存 NES システム (Ninf [3], NetSolve [7]) では、一般にユーザのジョブに対してその計算時間が最短となるようにサーバを割り当てる。これはユーザのデッドラインの要求を考慮しない積極的なアプローチ (以後 Greedy とする) である。しかしながら、NES システムがグリッド上で広く利用されるようになると、デッドラインを考慮したサーバ選択が重要になる。デッドラインスケジューリングはユーザの要求するデッドラインまでにジョブ処理を終了させることを目的とした手法である。

#### 3.1 スケジューリングアルゴリズム

次に示す本研究で用いたデッドラインスケジューリングのアルゴリズムでは、複数ユーザの発行するジョブに対して適切なサーバを選択し、デッドラインを超える総ジョブ数を軽減させることを目的とする。

(1) 利用可能なサーバ  $S_i$  ( $0 < i \leq n$ ) にジョブを投入した場合の処理時間の予測値  $T_{S_i}$  を算出する。

$$T_{S_i} = W_{send}/P_{send} + W_{recv}/P_{recv} + W_s/P_{serv}(1)$$

$W_{send}$ ,  $W_{recv}$ ,  $W_s$  はそれぞれジョブの送信/受信データ量、論理演算数を表し、 $P_{send}$ ,  $P_{recv}$ ,  $P_{serv}$  はクライアントからサーバ、サーバからクライアントへのネットワークバンド幅の予測値およびサーバでの処理速度の予測値を表す。グリッド環境ではネットワークやサーバ計算機の負荷の変動が大きく、一般にそれらの状況を定期的にモニタリングし、ジョブが割り当てられる際のネットワーク/サーバの負荷を予測して、その予測値をスケジューリングに利用する [28]。

(2) ジョブに定められたデッドラインの時刻  $T_{deadline}$  と現在の時刻  $T_{now}$  から、ジョブのデッドラインまでの時間  $T_{until\ deadline}$  を算出する。

$$T_{until\ deadline} = T_{deadline} - T_{now} \quad (2)$$

(3) 処理時間の目標値  $T_{target}$  を算出する。

$$T_{target} = T_{until\ deadline} \times Opt \quad (0 < Opt \leq 1) \quad (3)$$

パラメータ  $Opt$  はグリッド上での性能予測に対する精度の期待度を表す。グリッドモニタリングシステムの提供する予測値には予測エラーが含まれているため、どの程度保守的に予測の失敗を見積もるかを  $Opt$  により調節する。 $Opt$  を小さく設定することは予測失敗時のリスクを大きく見積もることを意味し、 $Opt$  を 1 に設定することは予測の精度が非常に高いと判断したことを意味する。

(4) 利用可能なサーバから処理時間の目標値を超えず、目標値に最も近い処理時間となるサーバ  $S_i$  を選出する。

$$Diff = T_{target} - T_{S_i} \geq 0 \text{ かつ, } Diff \text{ が最小} \quad (4)$$

ただし、 $Diff \geq 0$  となる  $S_i$  が 1 つもない場合は、 $Diff$  の絶対値が最小となるサーバを選出する。

### 3.2 Load Correction メカニズム

グリッド上のスケジューラは一般に NWS [28] 等から得られたリソースの負荷の測定値、予測値を利用する。前節で述べたアルゴリズムでもこれらの情報を用いている。しかしながら、実際にはサーバの利用状況に関する付加的な情報も保持している。すなわち、スケジューリングによりジョブが割り当てられたサーバの負荷は、そのジョブが投入されると高くなることがあらかじめ予測できる。一方、グリッドのモニタリングシステムは定期的にモニタリングを行うため、即座にその負荷の変化を察知することができない。

スケジューリングアルゴリズムの性能を改善するため、スケジューリングの履歴情報を利用してモニタリングシステムから得られた負荷予測値を補正し、それをスケジューリングの際に用いる（以後 Load Correction と呼ぶ）。この手法は NES システムのオンラインスケジューラに適用できる。

Load Correction では、モニタリングシステムから得られたサーバ  $S_i$  の負荷予測値を  $Load_{S_i}$  とすると、その補正值  $Load_{S_i, corrected}$  を次のように求める。

$$Load_{S_i, corrected} = Load_{S_i} + numjobs_{S_i} \times pload \quad (5)$$

$numjobs_{S_i}$  はそのグリッドシステムのスケジューラにより  $S_i$  に割り当てられたジョブの総数、 $pload$  は補正の大きさを調整する任意の値である。本稿の評価では、すべて  $pload = 1$  としている。これは、評価で想定するアプリケーションは計算主体であり、一般に計算が主体となるタスクはその計算機の CPU 負荷を 1 つ高くする

ためである（例えば、計算主体の 3 つのタスクを実行している計算機の負荷はほぼ 3.0 となる）。

### 3.3 Fallback メカニズム

ジョブに要する入力データの送信時間の見積もりに失敗する（実際のネットワークバンド幅が予測値より低かった場合）と、入力データがサーバに到着した後にそのジョブの実行がデッドラインまでに終了しないと予測できる場合もある。さらに、本稿の評価ではサーバは First Comes First Served (FCFS) でのジョブ処理を想定しており、サーバにジョブが到着する順序が異なったために実行時間の見積もりを誤る場合もある。この問題を解決するため、サーバ自身にスケジューリングをサポートする機能をもたせる（以後 Fallback とする）。

Fallback では、ジョブの入力データがサーバに到着した際、サーバはそのジョブが要求されたデッドラインまでに終了するかどうか見積もる。サーバがそのジョブがデッドラインまでに終了しないと判断した場合、サーバはそのジョブをキャンセルしてジョブを発行したクライアントにそれを通知する。クライアントは再びそのジョブの処理に適したサーバをスケジューラに問い合わせ、改めて選択されたサーバにジョブを投入する。ただし、この Fallback のオーバーヘッドによりデッドライン内にジョブを終了できなくなる可能性があるため、本稿では再投入する回数を制限する。

以下の条件を満たす場合、サーバはジョブがデッドラインまでに処理が終了しないと判断する。

$$T_{until\ deadline} < T_{send} + ET_{exec} + ET_{recv}$$

$$N_{fallbacks} \leq N_{max.\ fallbacks} \quad (6)$$

$T_{until\ deadline}$  は式 (2) で定められた値、 $T_{send}$  はクライアントからサーバへの入力データの通信に実際に要した時間、 $ET_{recv}$  は出力データの送信時間の見積もり、 $ET_{exec}$  は入力データの到着時に見積もられたそのサーバでのジョブの処理時間を示す。また、 $N_{fallbacks}$  はそのジョブに対して行われた Fallback の総数、 $N_{max.\ fallbacks}$  はあらかじめ指定された Fallback 数の上限値である。

## 4. スケジューリング手法の評価

Bricks システムを用い、複数クライアント/サーバ環境でのデッドラインスケジューリングアルゴリズムの性能を示す。評価では東京工業大学松岡研究室の Presto II クラスタ (Dual PentiumIII 800MHz,  $\times 64$  nodes) 上で、スケジューリングアルゴリズム、クラ

表 1 シミュレーションで用いたパラメータ  
Table 1 Parameters used in the experiments.

グリッド計算環境		
Local Domain 数	10	固定
Local Domain のノード数	5-10	一様分布
クライアントとサーバの比率	1:1	固定
平均 LAN バンド幅	50-100[Mbps]	一様分布
平均 WAN バンド幅	500-1000[Mbps]	一様分布
サーバ性能	100-500[Mops/s]	一様分布
サーバの平均負荷	0.1	固定
サーバの外乱ジョブ命令数	50[Mops]	固定
論理パケットサイズ	10[Mbits]	固定
クライアントジョブ特性		
送信 / 受信データ量	100-5000[Mbits]	一様分布
ジョブ論理演算数	1.5-1080[GOps]	一様分布
ジョブ発行間隔 (Int)	60/90/120[min]	ポアソン
Deadline パラメータ (DF)	1.0-3.0	一様分布
スケジューリングユニット		
モニタリング間隔	5[min]	固定
スケジューラ	Deadline/Greedy	
Opt	0.5/0.6/0.7/0.8/0.9	
Load Correction メカニズム	on / off	
Fallback メカニズム	$N_{max. fallbacks}$ = 0/1/2/3/4/5	

クライアント, サーバ, ネットワークポロジ, Opt, およびグリッドシステムの負荷を変化させたシミュレーションを約 2500 回行った. Java 2 Runtime Environment (1.3.0) と Java HotSpot Client VM では, 1 シミュレーション (75 ノード 24 時間) あたり 30-60[min] 要した. また, 各シミュレーションのクラスタ上での実行には APST ソフトウェア [14] を用いた. APST は独立した多数のタスクを利用可能なクラスタノードに自動的に割り当てるグリッドミドルウェアである.

#### 4.1 評価環境

シミュレーションでは, まず従来のアプローチである Greedy アルゴリズムと 3.1 のデッドラインスケジューリングアルゴリズム (以下 Deadline) を比較する. 次に, パラメータ Opt, Load Correction および Fallback メカニズムのスケジューリングの失敗率と選択したサーバのコストに対する影響を比較する. 1. で述べたように, 評価ではサーバ計算機の性能をそのサーバの “コスト” とする経済モデルを適用する.

シミュレーションで用いるパラメータの詳細を表 1 に示す. 表 1 のグリッド計算環境におけるパラメータを用いて, 5 つの異なるグリッド環境を生成する. 次節の評価実験結果は, この 5 種類のシミュレーションの平均値である. シミュレーションでの LAN/WAN の通信バンド幅は, 標準偏差が各平均バンド幅の 10% となる Self-Similar トレース (2.1) で決定する. また, 各サーバ

は FCFS でジョブを処理し, その負荷は平均が 0.1 となるよう外部ジョブの平均到着間隔を設定する (2.1). 外部ジョブはポアソン到着に従うものとする.

様々な特性をもつジョブに対するスケジューリングの性能を調べるため, クライアントの発行するジョブの論理演算量は負荷を考慮しないサーバ全体の平均性能 (300) に対して実行時間が 5-60[min] となるように 1.5-1080[GOps] に設定した. また, クライアントのジョブの平均発行間隔 Int は, グリッドシステム全体の負荷の度合いを調節するパラメータであり, Int が長くなるにつれ負荷が低くなる. 評価では, 負荷が比較的低い場合 (Int= 120), 中程度 (Int= 90), 高い場合 (Int= 60) を想定したスケジューリング性能を調査する. ネットワーク, サーバの負荷はバックグラウンドのジョブ, バンド幅に対して各ユーザから投入されるジョブとデータが加わるため, シミュレーション中の負荷は非常に高くなる. また, 各ジョブのデッドライン  $T_{deadline}$  は次のように決定する.

$$T_{deadline} = T_{start} + ((W_{send} + W_{recv})/B_{net} + W_s/B_{serv}) \times DF \quad (7)$$

$T_{start}$  はジョブが発行された時刻,  $W_{send}$ ,  $W_{recv}$ ,  $W_s$  はそれぞれジョブの送信量, 受信量, 論理演算数を表し,  $B_{net}$ ,  $B_{serv}$  は各クライアント・サーバ間のネットワークおよびサーバの最大スループットの平均値とする. また, デッドラインパラメータ DF はデッドラインまでの長さを決定する. 実際のグリッドの運用では, デッドラインは各ユーザの事情により決定される. 本評価ではデッドラインを考慮する際にクリティカルになると思われる範囲を想定し,  $1.0 \leq DF \leq 3.0$  とした.

表 1 のスケジューリングユニットにはスケジューリングに関するパラメータを示す. 評価では, 5[min] ごとにネットワーク / サーバの状況をモニタリングするモニタを用意する. 各クライアント・サーバ間のネットワークに対して定期的にプローブデータを流すことによりその混雑度を測定するが, すべてのネットワークでプローブを同時に開始するとプローブデータ同士の衝突により測定される通信スループットが著しく低下する恐れがある. よって, 本シミュレーションでは巡回してプローブするネットワークモニタを用いる. 実環境下でグリッドの負荷状況のモニタリング・予測するシステム NWS [28] でも, 同様の方式でモニタリングする.  $N_{max. fallbacks}$  は Fallback 回数の最大値である. また, 評価では資源モニタの観測値をその資源の負荷予測値とした. これは,

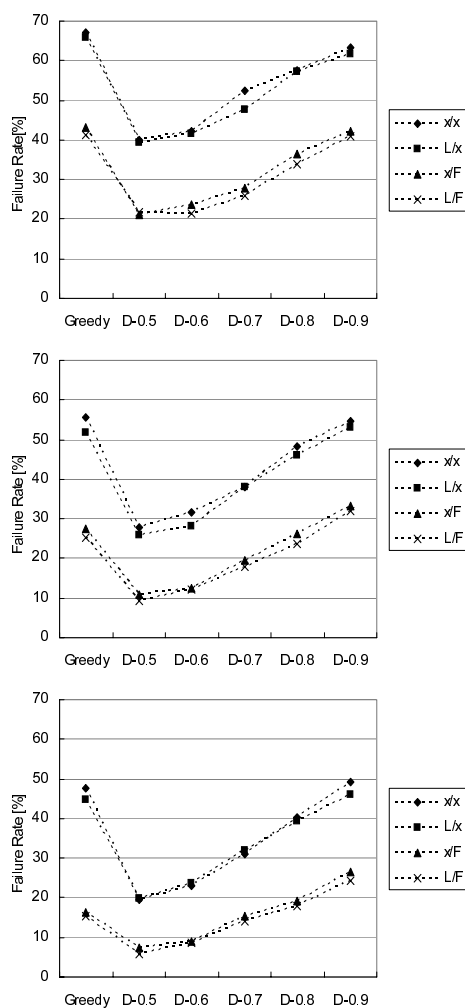


図3 GreedyとDeadlineのスケジューリング失敗率の比較 ( $N_{max. fallbacks} = 1, lnt = 60$  (上),  $90$  (中),  $120$  (下)) .  
 Fig.3 The failure rate for the Greedy algorithm and of the Deadline algorithm with different values of  $Opt(N_{max. fallbacks} = 1, lnt = 60$  (top),  $90$  (middle),  $120$  (bottom)).

現状ではグリッド環境でのネットワーク / 計算機負荷の予測の精度が保証されていないため、まず基礎データとしてモニタリングから得られた情報をもとにスケジューリングした結果を調査した。

#### 4.2 評価結果

図3にスケジューリングアルゴリズム Greedy と Deadline のデッドラインスケジューリングの失敗率を示す。Deadline では異なる Opt を設定した結果を示して

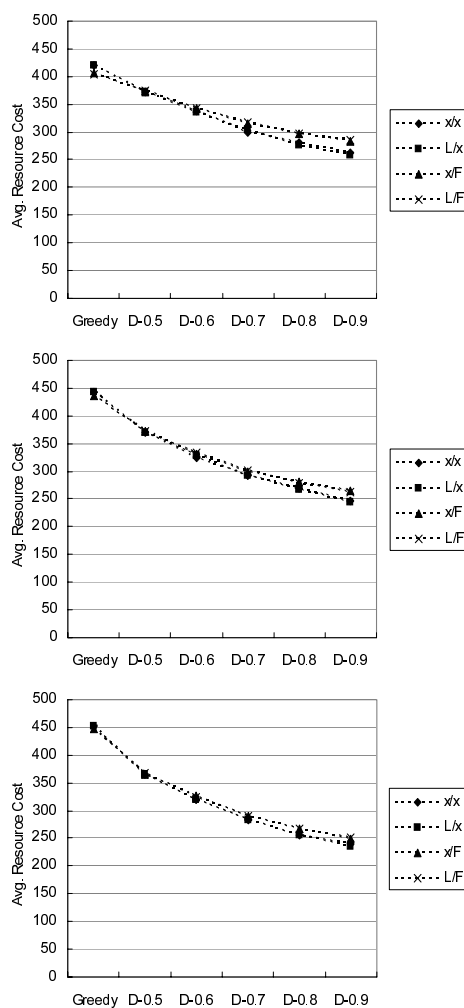


図4 平均資源コストの比較 ( $N_{max. fallbacks} = 1, lnt = 60$  (上),  $90$  (中),  $120$  (下)) .  
 Fig.4 The average resource cost over all requests ( $N_{max. fallbacks} = 1, lnt = 60$  (top),  $90$  (middle),  $120$  (bottom)).

いる (図3では、X軸に D-Opt と表記する)。なお、Greedy は  $Opt = 0$  に設定した場合と同等である。失敗率は総ジョブ数に対するデッドラインを超えたジョブ数の割合である。図3では、負荷が異なる ( $lnt = 60, 90, 120$ ) 仮想グリッドシステムの結果をそれぞれ示している。各グラフ中のスケジューリング手法 x/x, L/x, x/F, L/F は、X軸に表記した各スケジューリングアルゴリズムに対して何も用いないもの、Load Correction を用いたもの、Fallback を用いたもの、Load Correction お

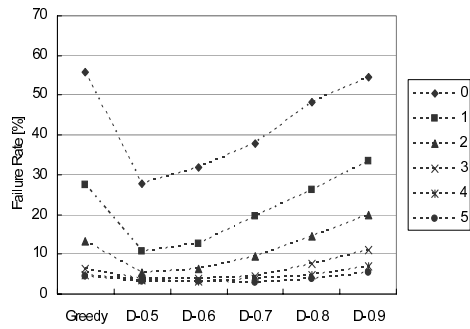


図5 Greedy と Deadline のスケジューリング失敗率の比較 ( $N_{max. fallbacks} = 1/2/3/4/5, lnt=90$ ) .

Fig. 5 The failure rate for the Greedy algorithm and the Deadline algorithm with different values of Opt with multiple fallbacks ( $N_{max. fallbacks} = 1, 2, 3, 4, 5, lnt=90$  (medium)).

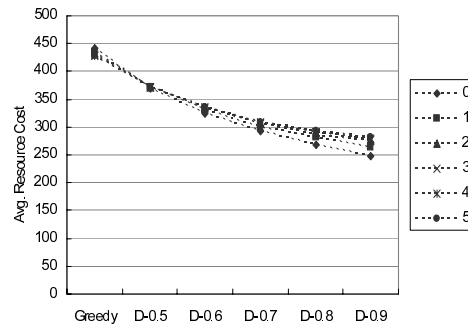


図6 平均資源コストの比較 ( $N_{max. fallbacks} = 1/2/3/4/5, lnt=90$ ) .

Fig. 6 The average resource cost over all requests ( $N_{max. fallbacks} = 1, 2, 3, 4, 5, lnt=90$  (medium)).

よび Fallback を用いたものの結果を示している (L は Load Correction, F は Fallback, x は各メカニズムを使用していない場合を表す) . Fallback での最大 fallback 数 ( $N_{max. fallbacks}$ ) は 1 とした . Greedy かつ x/x の場合が従来の手法の結果となる .

図 3 より, 負荷の異なるいずれの環境においても, Fallback が失敗率を低下させるための有効な手段であることが示された . 一方, Load Correction の失敗率低下の効果は小さかった . また, Deadline は Opt を小さく設定するとスケジューラは各ジョブに対してより厳しいデッドラインを想定するため, 失敗率を低下させている . ただし, 経済モデル下でのスケジューリングアルゴリズムの性能比較では, 失敗率だけでなく各ジョブが割り当てられるサーバの計算資源コストを考慮する必要がある .

図 4 に図 3 と同じシミュレーションでの各アルゴリズムの平均資源コストを示す . グラフより, Opt を大きく設定すると平均資源コストが低下し, Greedy は最もコストが高いことが分かる . すなわち, NES システムでは保守的なデッドラインスケジューリング (Opt が小さい場合) の方が, 既存アルゴリズムより失敗率, コストの面で有効であることが示された . また, 図 3, 4 より, Opt により Deadline アルゴリズムの保守性のレベルを変化させて失敗率とコストのトレードオフの調整が可能であることが示された .

図 5, 6 に, Load Correction を用いずに最大 fallback 数  $N_{max. fallbacks}$  を 0 から 5 に変化させた場合の失敗率と平均資源コストの比較結果を示す . ただし,

同様の傾向を示していたため,  $lnt=90$  の結果のみを示す . 図 5 より,  $N_{max. fallbacks}$  を大きくするとすべてのスケジューリングアルゴリズムにおいて失敗率が低下し, スケジューリング性能を向上させることが分かる . また, 図 6 より,  $N_{max. fallbacks}$  を大きく設定した場合でも, 資源コストが著しく増大しないことが分かる . よって, NES システムのスケジューラでは, 従来のスケジューリングメカニズムに加えて複数回の fallback を行う機構を組み込むと非常に有効であることが分かった .

## 5. 関連研究

1. で述べたように, すでに経済モデルの概念を利用したスケジューリング手法が複数提案されている . Nimrod [21] では, Parameter Sweep 型のジョブに対して self-scheduling によるデッドラインスケジューリング手法を提案・実装している . 実環境での実験 [21] では, デッドラインが近づくにつれ, スケジューラがコストの高い計算機を選択するようになる結果を示している . ただし, 1 人のユーザから発行される 1 ジョブのスケジューリングであり, 本稿の複数ユーザが複数サーバに対して同時にジョブを投入する状況とは異なる . 複数タスクからなる Parameter Sweep 型アプリケーションのデッドラインスケジューリングでは, 各タスクの処理が予測に反して処理時間が長くなっても最終的なデッドラインを超えなければよいので, スケジューリングの失敗率は低くなる . ただし, グリッド環境では単にデッドラインに注目するだけでなく, 計算資源の負荷に応じて資源割り当ての積極さを変化させることでより低いコスト

の計算資源でジョブを実行できる可能性が高い<sup>(注2)</sup>。

より洗練された経済モデルとして、文献[18]では、様々な管理ドメイン間のリソース共有のためのチケットと通貨を用いた共有合意メカニズムを提案している。また、G-Commerce[19]では、計算資源の提供者と消費者の間をBankが仲介して資源(コモディティ)の価格を決定し、最終的にオークションの概念を用いて各コモディティを消費者に割り当てていくスケジューリング手法を提案している。ただし、実際の利用状況を想定した定量的な評価は行われていない。

また、グリッドの評価環境ではMicroGrid[29]、Simgrid[30]が挙げられる。MicroGridはクラスタ計算機上にGlobus[4]ベースの仮想グリッドシステムを構築するグリッドエミュレータである。実アプリケーションをそのままMicroGrid上で実行することが出来るため、アプリケーションのモデリングが不要だが、ジョブの実行には実時間以上を要し、システムの制御に対する負担も大きい。Simgridはトレースベースの離散シミュレータであり、グリッドシミュレーションの基本的なツールキットを提供する。Simgridでは単一の並列アプリケーションの実行時間を最短にするタスクスケジューリングアルゴリズムの評価(処理時間の予測)を目的としている。スケジューリング手法の評価を行うには、Simgridの上の階層に別のシミュレータを実装する必要がある。

## 6. まとめと今後の課題

グリッド技術に関する研究は蓄積してきたものの、そのスケジューリング手法に関する定量的な調査/解析はまだ不十分である。また、従来のグリッドのスケジューリングではいかに速くジョブを処理するかに重点がおかれていたが、広域ネットワーク上の特定多数のユーザに対して計算資源を提供する場合、計算資源に対するアクセス権限や資源のプライオリティ等を考慮し、ユーザの求める計算性能を提供することが重要な要素となる。

本稿ではデッドラインスケジューリングに着目し、簡単なスケジューリングアルゴリズムとスケジューリングアルゴリズムをサポートする2つのメカニズム、Load CorrectionとFallbackを提案した。Load Correctionはグリッドモニタリングシステムにより得られた資源情報をスケジューリング結果を利用して補正するものである。Fallbackはサーバにジョブの入力データが到着した

際にそのジョブがデッドラインを超えるかどうか予測し、超えると判断した場合は別のサーバにジョブを再投入させるメカニズムである。また、複数ユーザが複数サーバに対してジョブを投入する状況を想定し、グリッドコンピューティングのための評価システムBricks上でシミュレーションによるスケジューリングアルゴリズムの評価実験を行い、以下の知見を得た。

- GreedyアルゴリズムはDeadlineより選択されるサーバのコストが高い。
- 保守的にデッドラインを見積もる(Optが小さい場合)と、資源コストは高くなるがスケジューリングの失敗率が低下する。
- Fallbackメカニズムは失敗率を大きく低下させ、複数fallbackを行った場合でも平均資源コストが大きくならない。

よって、将来のグリッドNESシステムで複数ユーザにより計算資源の競合が起こる場合には、複数回のfallbackをサポートするデッドラインスケジューリングが非常に有効であることが明らかになった。

今後の課題として、デッドラインスケジューリング手法の性能向上を図る。具体的には、各ジョブのプライオリティを考慮し、プライオリティの高い(DFの小さい)ジョブにはOptを小さく設定することにより、失敗率の低下を図る。また、システムの利用状況を厳格にモニタリングすることにより、より高精度なLoad Correctionを試みる。さらに、より洗練された経済モデル下でのスケジューリングアルゴリズムの評価をBricksシステムを改良して行い、シミュレーションでその有効性を調査するとともに、得られた知見から実際のグリッドシステムNinf[3]におけるデッドラインスケジューラを実装し、実用に向けて実環境での挙動を調査する。

謝辞 本研究を進めるにあたりご助言いただいたUCSDのHenri Casanova氏、Francine Berman教授に深く感謝致します。本研究は文部科学省科学研究費補助金および科学技術振興財団(さきがけ21)により支援されている。

## 文献

- [1] I. Foster and C. Kesselman, editors, "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 1999.
- [2] Global Grid Forum, <http://www.gridforum.org/>.
- [3] Ninf, <http://ninf.etl.go.jp/>.
- [4] Globus, <http://www.globus.org/>.
- [5] Legion, <http://www.cs.virginia.edu/~legion/>.
- [6] Condor, <http://www.cs.wisc.edu/condor/>.

(注2): 経済モデルを応用し、計算機の利用率(需要)が高い場合はその計算資源の価格を上げ、低い場合には下げる方針も複数提案されている[19],[21]。

- [7] NetSolve, <http://www.cs.utk.edu/netsolve/>.
- [8] N.H. Kapadia, J.A.B. Forter, and C.E. Brodley, "Predictive Application-Performance Modeling in a Computational Grid Environment", In *Processings of the HPDC-8*, 1999.
- [9] J. Czyzyk, M. Mesnier, and J. Moré, "NEOS: The Network-Enabled Optimization System", Technical Report MCS-P615-1096, Mathematics and Computer Science Division, Argonne National Laboratory, 1996.
- [10] J.R. Stiles, T.M. Bartol, E.E. Salpeter, , and M.M. Salpeter, "Monte Carlo simulation of neuromuscular transmitter release using MCell, a general simulator of cellular physiological processes", *Computational Neuroscience*, pages 279–284, 1998.
- [11] S. Rogers, "A Comparison of Implicit Schemes for the Incompressible Navier-Stokes Equations with Artificial Compressibility", *AIAA Journal*, 33(10), Oct. 1995.
- [12] F. Berman. "The Grid, Blueprint for a New computing Infrastructure", chapter 12, Morgan Kaufmann Publishers, Inc., 1998. Edited by Ian Foster and Carl Kesselman.
- [13] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao, "Application-Level Scheduling on Distributed Heterogeneous Networks", In *Proc. of SC96*, 1996.
- [14] H. Casanova, G. Obertelli, F. Berman, and R. Wolski, "The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid". In *Proc. of SC2000*, 2000.
- [15] J.P. Goux, S. Kulkarni, J. Linderoth, and M. Yoder, "An Enabling Framework for Master-Worker Applications on the Computational Grid", In *Proc. of HPDC-9*, pages 43–50, 1999.
- [16] A. Turgeon, Q. Snell, and M. Clement, "Application Placement Using Performance Surfaces", In *Proc. of HPDC-9*, 1999.
- [17] J.M. Schopf and F. Berman, "Stochastic Scheduling", In *Proc. of SC99*, 1999.
- [18] T. Zhao and V. Karamcheti, "Expressing and Enforcing Distributed Resource Sharing Agreements", In *Proc. of SC2000*, 2000.
- [19] J.S. Plank, R. Wolski, J. Brevik, and T. Bryan, "G-Commerce: The Study and Building of Computational Economies for the Computational Grid, 2000. Workshop on Clusters and Computational Grids for Scientific Computing <http://www.cs.utk.edu/~dongarra/lyon2000/lyon-2000.html>.
- [20] R. Buyya, D. Abramson, and J. Giddy, "An Economy Driven Resource Management Architecture for Global Computational Power Grids", In *Proc. of PDPTA2000*, June 2000.
- [21] D. Abramson, J. Giddy, and L. Kotler, "High Performance Parametric Modeling with Nimrod/G: Killer Application for the Glocal Grid?", In *Proc. of IPDPS2000*, 2000.
- [22] 竹房, 合田, 松岡, 中田, and 長嶋, "グローバルコンピューティングのスケジューリングのための性能評価システム", 情報処理学会論文誌, 41(5):1628–1638, 5 2000.
- [23] Bricks, <http://ninf.is.titech.ac.jp/bricks/>.
- [24] K. L. Calvert, M. B. Doar, and E. W. Zegura, "Modeling Internet Topology", In *IEEE Communications*, 1997.
- [25] A. Medina, I. Matta, and J. Byers, "On the Origin of Power Laws in Internet Topologies", *Computer Communication Review*, 30(2):18–28, 2000.
- [26] V. Paxson, "Fast, Approximate Synthesis of Fractional Gaussian Noise for Generating Self-Similar Network Traffic", *Computer Communication Review*, 27(5):5–18, 1997.
- [27] V. Paxson and S. Floyd, "Wide-Area Traffic: The Failure of Poisson Modeling", In *SIGCOMM '94*, pages 257–268, 1994.
- [28] NWS, <http://nws.npaci.edu/NWS/>.
- [29] H. J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien, "The MicroGrid: a Scientific Tool for Modeling Computational Grids", In *Proc. of SC2000*, 2000.
- [30] H. Casanova, "Simgrid: A Toolkit for the Simulation of Grid Application Scheduling", In *Submitted to CCGRID2001*, 2001.

(平成年月日受付, 月日再受付)

## 竹房あつ子

昭和 48 年生。平成 8 年お茶の水女子大学理学部情報科学科卒業。平成 10 年同大学大学院理学研究科情報科学専攻修士課程修了。平成 12 年同大学院人間文化研究科複合領域科学専攻博士課程修了。博士(理学)。同年日本学術振興会特別研究員。平成 14 年お茶の水女子大学理学部助手に就任。並列分散処理, グローバルコンピューティング, スケジューリングに興味を持つ。ACM, 情報処理学会会員。

## 松岡 聡

昭和 38 年生。昭和 61 年東京大学理学部情報科学科卒, 平成元年同大学大学院博士課程中退。同大学情報科学科助手, 情報工学専攻講師を経て, 平成 8 年より東京工業大学情報理工学研究所数理・計算科学専攻助教授。平成 10 年より科学技術振興財団のさきがけ研究員を併任。理学博士。オブジェクト指向言語, 並列システム, リフレクティブ言語, 制約言語, ユーザ・インタフェースソフトウェアなどの研究に従事。平成 8 年度情報処理学会論文賞, 平成 11 年情報処理学会坂井記念賞受賞。ソフトウェア科学会, ACM, IEEE-CS 各会員。IEEE Concurrency Magazine の Editor。

# Performance of a Deadline-Scheduling Scheme on the Computational Grids

Atsuko TAKEFUSA  
Ochanomizu University

Otsuka 2-1-1, Bunkyo-ku, Tokyo 112-8610, Japan

Satoshi Matsuoka

Tokyo Institute of Technology, GSIC / National Institute of Informatics  
Ookayama 2-12-1, Meguro-ku, Tokyo 152-8550, Japan

## **Abstract**

A number of projects have addressed the idea of software as a *service* on the computational Grid. These systems implement client-server architectures with many servers running on distributed Grid resources and have commonly been referred to as *Network-enabled servers* (NES). The Bricks Grid simulation framework has been developed and extensively used to evaluate scheduling strategies for NES systems. Using Bricks, we discuss a *deadline-scheduling* strategy that is appropriate for the multi-client multi-server case. We propose a simple deadline-scheduling algorithm and then augment it with Load Correction and Fallback mechanisms which could improve the performance of the algorithm in our context.

## **Keyword**

Grid, Deadline, Scheduling, Performance Analysis, Simulation