グローバルコンピューティングシステムのシミュレーションによる評価

竹房 あつ子[†1] 合田 憲人[†2] 中田 秀基[†3]
小川 宏高[†2] 松岡 聡[†2] 佐藤 三久[†4]
関口 智嗣[†3] 長嶋 雲兵[†5]

グローバルコンピューティングを実現する試みが行われる一方，グローバルコンピューティングのスケジューリングに関する明確な知見は得られていない．これは，広域ネットワークでは再現性のある大規模性能評価が非常に困難で，性能要素に関する調査が十分なされていないためである．本稿では，グローバルコンピューティングシステムにおけるスケジューリングの公正な性能評価を行うシミュレータの設計および実装と，本シミュレータを用いた性能評価実験を行った．本シミュレータではネットワークとサーバを待ち行列で表し，その組合せにより多様なグローバルコンピューティングシステムが表現可能である．本シミュレータで実際のグローバルコンピューティングシステムをシミュレーションした結果，実測とほぼ同様の結果が得られた．また，本シミュレータを用いたスケジューリング手法の性能評価では，資源状況を適切に考慮したスケジューリングが有効であることが分かった．

Performance Evaluation of Global Computing Systems by Simulation

Atsuko Takefusa,[†1] Kento Aida,[†2] Hidemoto Nakada,[†3]
Hirotaka Ogawa,[†2] Satoshi Matsuoka,[†2] Mitsuhisa Sato,[†4]
Satoshi Sekiguchi[†3] and Umpei Nagashima [†5]

While there have been several proposals of high performance global computing systems, scheduling schemes for the systems have not been well investigated. The reason is difficulties of evaluation by large-scale benchmarks with reproducible results. This paper describes design and implementation of the simulator that evaluates scheduling schemes on a typical high-performance global computing system. The simulator can simulate various features of global computing systems by adopting a queueing model. Effectiveness of the simulator was verified by the simulation results, which showed very similar results to the experimental results on a real global computing system. This paper also shows simulation results of simple scheduling schemes by the simulator. Results show it is important to consider resource conditions appropriately for overall system performance.

1. はじめに

ネットワーク技術の発展により，広域ネットワーク上に分散した計算資源や情報資源を積極的に活用して大規模計算を実現する，グローバルコンピューティングが可能となった．近年これを目的としたシステムがNinf[1]をはじめ複数提案されている[2)~6)]．各システムでは，グローバルコンピューティングを効率的に行うために，Ninf: MetaServer，NetSolve: Agent, Condor: Matchmaking[7)] およびGlobus, Legionの資源管理フレームワーク[8),9)]に代表される，独自のスケジューリングフレームワークを実装している．また，AppLeS[10)]，Prophet[11)]といったグローバルコンピューティングシステム上でスケジューリングを専門に行うシステムや，ネットワークの通信スループットや通信レイテンシ，および計算サーバの負荷など，広域ネットワークにおける資源状況を予測するNWS[12)]が提案されている．

グローバルコンピューティングシステムにおける各スケジューリング手法を比較し，その特性を調査するためには，様々な

- ネットワークトポロジおよびバンド幅

†1 お茶の水女子大学
Ochanomizu University
†2 東京工業大学
Tokyo Institute of Technology
†3 電子技術総合研究所
Electrotechnical Laboratory
†4 新情報処理開発機構
Real World Computing Partnership
†5 物質工学工業技術研究所
National Institute of Materials and Chemical Research

- ネットワークの混雑度とその変動
- 各計算資源のアーキテクチャと性能
- 各計算資源の負荷とその変動

等を想定した，再現性のある大規模環境での評価が求められる．一方，現在提案されているスケジューリングシステムでは，限られた計算機とネットワークトポロジにおける小規模の評価実験に留まっており，その有効性が明らかにされていない．また，広域ネットワークを用いた実験では，各計測ごとに対象とするネットワークのスループットが変動し，計算サーバの負荷も異なるので，各スケジューリング手法の比較が非常に困難である．グローバルコンピューティングシステムの各スケジューリング手法の正当性を実証するには，実システムにおける評価だけでは不十分であるにも関わらず，性能評価環境に関する議論はほとんどなされていなかった．

本稿では，グローバルコンピューティングシステムにおけるスケジューリング手法の性能評価を行う枠組を提供するシミュレータの設計および実装を行った．シミュレータを用いることにより，各グローバルコンピューティングシステムのスケジューリング手法の特性の抽出を容易にし，その比較を公正に行うことができる．

本シミュレータでは，ネットワークとサーバを待ち行列を用いてモデル化し，その組合せにより多様なグローバルコンピューティングシステムを表現することを可能とした．また，本シミュレータにより実際のグローバルコンピューティングシステムをモデルとしたシミュレーションを行った結果，実測とほぼ同様の結果が得られ，その有効性を確認した．さらに，本シミュレータを用いたスケジューリング手法の性能評価実験を行った．以後 **2**ではグローバルコンピューティングの概要について述べ，**3**でグローバルコンピューティングシステムのシミュレーションモデルについて説明する．**4**では本シミュレータの基本性能の評価を行い，**5**でスケジューリング手法を評価した結果を報告する．

2. グローバルコンピューティングの概要

スケジューリングフレームワークを提供するグローバルコンピューティングシステムは，基本的に次のコンポーネントにより構成される (図 **1**)．

クライアント (*Client*)　グローバルコンピューティングシステムのユーザへの API を提供する．

サーバ (*Server*)　計算資源および計算ライブラリ等の情報資源を提供する．

スケジューラ (*Scheduler*)　各グローバルコンピューティングシステムのポリシーに従ったスケジューリングを行う．

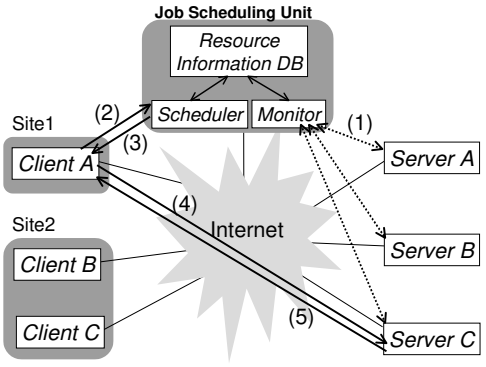

図 **1**　グローバルコンピューティングの実行の流れ
Fig. 1　Execution process of global computing.

資源情報 **DB** (*Resource Information DB*)　サーバの性能や負荷，およびサーバで提供するライブラリなど，グローバルコンピューティングシステムにおける資源情報を管理する．

モニタ (*Monitor*)　サーバおよびネットワークのモニタリングと，それらの資源状況の予測を行う．

具体的には，グローバルコンピューティングは図 **1**に示す手順で実行する．

(1)　モニタはサーバ，ネットワークの状況をモニタしている．サーバ，ネットワークの資源情報は資源情報 DB により管理される．

(2)　ユーザの計算要求が発生すると，クライアントは計算を実行する適切な計算サーバをスケジューラに問い合わせる．

(3)　スケジューラは資源情報 DB に問い合わせ，サーバ，ネットワークの資源情報とユーザの要求する計算に関する情報により，適切なサーバを選定してクライアントに紹介する．

(4)　クライアントは紹介されたサーバに対して計算要求を発行する．

(5)　サーバは計算を実行した後，その計算結果をクライアントに返す．

3. グローバルコンピューティングシステムシミュレータ

ネットワークのトポロジ / 混雑度や各計算サーバの性能 / 負荷など，グローバルコンピューティングにおける様々な環境設定と，再現性のある大規模評価実験を可能とする性能評価手法として，グローバルコンピューティングシミュレータの設計・実装を行った．本シミュレータにより，グローバルコンピューティングシステムを実現するための問題点を抽出するとともに，グローバルコンピューティングにおける最適なスケジューリング手法を検討することができる．

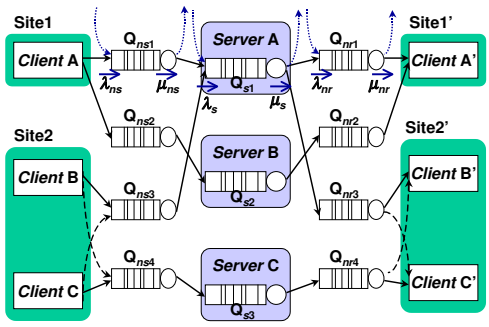

図 2 グローバルコンピューティングシステムのシミュレーションモデル
Fig. 2 Simulation model of global computing system.

3.1 シミュレーションモデルの概要

本シミュレーションモデルでは，グローバルコンピューティングシステムはクライアント，サーバ，クライアントとサーバ間のネットワーク（往路・復路），スケジューラ，資源情報 DB，モニタにより構成される．このうち，サーバとネットワークは図 2に示す待ち行列を用いて表現する．図 2中の，Q_{ns}, Q_{nr}, Q_s はそれぞれクライアントからサーバへのネットワーク（往路），サーバからクライアントへのネットワーク（復路），サーバを待ち行列で表したものである．また，$ClientA$ と A' は同一クライアントを表し，$ClientB$ と $ClientC$ は同一サイトにあるクライアントを示す．同一サイト内のクライアントが同じサーバに対してデータを転送する場合，ネットワークを共有すると考えられる．よって，このような場合は同じネットワークの待ち行列にデータを転送するものとする．

グローバルコンピューティングシステムでは，広域ネットワーク上に対象とするグローバルコンピューティングシステム以外のシステムから転送されるデータが存在する．また，その計算サーバと同一マシン上にグローバルコンピューティング以外のジョブが存在する．これらはグローバルコンピューティングの計算要求（以後 Request とする）の応答時間に大きな影響を与える．よって，シミュレーションではこれらのデータやジョブ（外乱）が想定可能であることが要求される．本シミュレータでは，これらの外乱を Q_{ns}, Q_{nr}, Q_s に到着するデータまたはジョブにより表現する．具体的には，図 2に示すように，Q_{ns}, Q_{nr}, Q_s には，Request によるものとは別に，外乱を表すデータまたはジョブが到着する．

図中の λ_{ns}, λ_{nr}, λ_s は，それぞれ Q_{ns}, Q_{nr}, Q_s に到着する Request と外乱のデータまたはジョブを合わせた到着率であり，

$$\lambda_{ns} = \lambda_{ns_request} + \lambda_{ns_others}$$
$$\lambda_{nr} = \lambda_{nr_request} + \lambda_{nr_others}$$
$$\lambda_s = \lambda_{s_request} + \lambda_{s_others}$$

の関係がある．ここで $\lambda_{ns_request}$ は Request のデータの到着率，λ_{ns_others} は Request 以外のデータの到着率を表す．$\lambda_{nr_request}$, λ_{nr_others}, $\lambda_{s_request}$, λ_{s_others} についても同様である．

クライアントから発行された Request は，以下のパラメータを持ち，Q_{ns}, Q_s, Q_{nr} の順で各待ち行列で処理される．

- 演算数：W_s
- 転送データ量（往路）：W_{ns}
- 転送データ量（復路）：W_{nr}

W_s は Request がサーバ上で実行される際の計算の演算数を意味する．W_{ns} はクライアントからサーバへ転送されるサーバ上での計算に必要なデータの転送量，W_{nr} はサーバからクライアントへ転送される計算結果等のデータ量を意味する．

次に，クライアント，クライアント・サーバ間のネットワーク，サーバにおける具体的な処理について示す．

3.1.1 クライアント

クライアントでは $\lambda_{request}$ の確率で Request を発行する．Request が発行されると，クライアントはスケジューラに Request に関する情報を提供し，その Request に対して適切なサーバを問い合わせる．スケジューラがサーバを決定するとそれがクライアントに通知され，クライアントは計算に必要なデータをサーバに転送する．この際，データはサイズ W_{ns} の論理パケットに分割され，論理パケットは λ_{packet} の確率で Q_{ns} に投入される．また，サーバでの処理が終了すると，Q_{nr} から論理パケットに分割されたデータを受けとり，全ての論理パケットが到着した時点でその Request が終了する．

ここで論理パケットの投入率 λ_{packet} はネットワークのバンド幅を $T_{net}[\mathrm{byte/s}]$，論理パケットサイズを $W_{packet}[\mathrm{byte}]$ とすると，以下のように定める．

$$\lambda_{packet} = T_{net}/W_{packet} \quad (1)$$

このように λ_{packet} を設定すると，Q_{nr} に外乱のデータが到着しない場合は Request のデータの通信スループット T_{act} がバンド幅 T_{net} と等しくなり，外乱のデータの到着率を高くすることにより T_{act} を低くなるように表現することができる．

3.1.2 クライアントからサーバへのネットワーク

Q_{ns} には M/M/1/N の有限バッファ待ち行列[13] を採用する．Q_{ns} にはクライアントから発行された Request の論理パケットと外乱のデータが到着する．この際，待

ち行列のバッファが一杯ならば各データは再送される．Q_{ns} に入った論理パケットは，Q_{ns} 待ち行列のサーバ上で $[W_{packet}/T_{net}]$ 時間処理され，Q_s に投入される．外乱のデータは同様に処理された後，破棄される．

外乱データの到着率 λ_{ns_others} は，Q_{ns} によって表されるネットワークの通信スループット T_{act} を決定するパラメータである．具体的には，λ_{ns_others} を高く設定した場合，Q_{ns} のバッファ中に存在する外乱データの数が多くなるため，Q_{ns} に到着した Request の論理パケットがバッファに入り切らず，再送される確率が高くなる．逆に λ_{ns_others} を低く設定した場合は，Q_{ns} に到着した Request の論理パケットが再送される可能性が低くなる．即ちこれは，適切な λ_{ns_others} を設定することにより，データの再送が起こり，Request のデータの通信スループットを任意に表現できることを意味する．

バンド幅 T_{net} のネットワークで，T_{act} の通信スループットを得るための λ_{ns_others} は，以下の式により決定する．

$$\frac{\lambda_{packet}}{\lambda_{ns_others}+\lambda_{packet}} = \frac{T_{act}}{T_{net}}$$

$$\Leftrightarrow \lambda_{ns_others} = (\frac{T_{net}}{T_{act}} - 1) \times \lambda_{packet} \qquad (2)$$

ここで λ_{packet} は式 **(1)** で決定される．例えば，T_{net} = 1.0[MB/s], W_{packet} = 0.1[MB] という条件で，T_{act} = 0.1[MB/s] となるような状況を表現するには，

$\lambda_{packet} = 1.0/0.1 = 10$

$\lambda_{ns_others} = (1.0/0.1 - 1) \times 10 = 90$

とすればよい．

また，本シミュレーションでは有限バッファ長 N を以下のように定める．

$$\frac{W_{packet} \times N}{T_{net}} \simeq T_{latency}$$

$$\Leftrightarrow N \simeq \frac{T_{latency} \times T_{net}}{W_{packet}} \qquad (3)$$

ただし，$N \geq 2$ で $T_{latency}$ は広域ネットワークにおける平均通信レイテンシとする．現状の広域ネットワークを想定して式 **(2)** のように λ_{ns_others} を決定すると，待ち行列のバッファは定常的に溢れる状態になり，Q_{ns} の平均待ち行列長は有限バッファ長 N とほぼ等しくなる．よって，Q_{ns} に投入されるあらゆるデータに対し，有限バッファにあるデータ分の遅延が生じるため，この遅延を $T_{latency}$ と考えることにした．

3.1.3 サーバ

本シミュレーションでは，サーバ上でジョブは，First Come First Served（FCFS）で処理されるものとし，Q_s には M/M/1 待ち行列[13] を用いる．ただし，サーバでのスケジューリング方式がタイムシェアリング等の異なる方式をとる場合も，Q_s は対応するモデルに適宜変更可能である．

Request のジョブは，関連する Request の論理パケットがすべて Q_{ns} から出た時点で Q_s に投入される．Request のジョブは Q_s の性能が T_{ser} のサーバ上で $[W_s/T_{ser}]$ 時間処理され，処理が終るとサイズ W_{nr} の Request のデータが再び論理パケットに分割されて Q_{nr} に投入される．外乱のジョブは同様に処理された後，破棄される．

外乱のジョブの到着率 λ_{s_others} はサーバの稼働率を決定するパラメータである．本シミュレーションでは外乱のジョブの平均演算数を W_{s_others}，稼働率を U としたときの λ_{s_others} を次のように決定する．

$$\lambda_{s_others} = \frac{T_{ser}}{W_{s_others}} \times U \qquad (4)$$

3.1.4 サーバからクライアントへのネットワーク

3.1.2と同様に Q_{nr} は M/M/1/N の有限バッファ待ち行列を採用し，Q_{nr} には Request の論理パケットと外乱のデータが到着する．Request のパケットは Q_{ns} 同様，Q_{nr} のサーバ上で $[W_{packet}/T_{net}]$ 時間処理され，クライアントに転送される．λ_{nr_others} も λ_{ns_others} と同様に決定する．

3.1.5 シミュレーションモデルのまとめ

表 **1** に，本シミュレーションモデルにおいて必要とされるパラメータをまとめた．表中の資源はグローバルコンピューティングシステムにおける各資源を表し，構成要素は本シミュレーションモデルで必要とされる要素を表す．決定パラメータは，構成要素の値等を求める際に必要となるパラメータを示す．

3.2 シミュレータの実装

本システムはオブジェクト指向言語の Java で実装した．シミュレーション環境の生成部分を Abstract Factory パターン[14] で実装しており，

- クライアント，サーバ，ネットワークの構成
- スケジューリングモデル
- 待ち行列（ネットワーク / サーバでの処理方法）

等を容易に組み変えることができる．また，ネットワーク，サーバにおける外乱のデータ，ジョブのサイズや発生間隔など，各オブジェクトに対して独立の乱数系列が指定可能であるため，柔軟なシミュレーション設定が可能である．

4. シミュレータの評価

3で提案したシミュレーションモデルの正当性を評価するため，本シミュレータを用い，図 **3**に示す実際のグ

表 1 シミュレーションモデルで必要とされるパラメータ
Table 1 The parameters for the simulation model.

資源	構成要素	モデルでの表現	決定パラメータ
トポロジ	待ち行列のトポロジ	待ち行列間のリンク	クライアントとサーバ間のリンク情報
クライアント	論理パケットの投入率	λ_{packet}	ネットワークのバンド幅，論理パケットサイズ
サーバ	ジョブの処理性能	μ_s	CPU 性能
	混雑度	λ_{s_others}	CPU 性能，外乱のジョブの平均演算数， 実際にサーバで計測された平均稼働率
	サーバの負荷の変動	到着分布	ポアソン到着など
ネットワーク	バンド幅	μ_{ns} ， μ_{nr}	ネットワークのバンド幅
	混雑度	λ_{ns_others}, λ_{nr_others}	バンド幅，論理パケットサイズ， 実際にネットワークで計測された平均通信スループット
	通信レイテンシ	N	バンド幅，論理パケットサイズ， 実際にネットワークで計測された平均通信レイテンシ
	通信スループットの変動	到着分布	ポアソン到着など

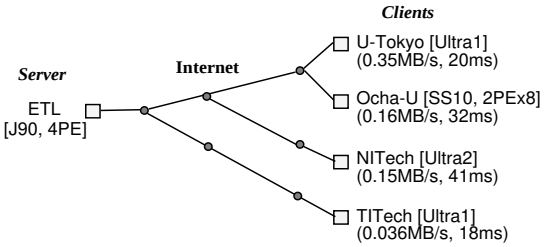

図 3 グローバルコンピューティングの計測環境
Fig. 3 Benchmarking environment of global computing.

ローバルコンピューティングシステムを例にしたシミュレーションを行った．図 3に示す環境については，グローバルコンピューティングシステム Ninf 上での実測による実行結果が得られているため[15),16)]，シミュレートするグローバルコンピューティングシステムとして Ninf システムを想定する．図中の括弧内には，実測時における各クライアントとサーバ間の FTP スループット， ping レイテンシを示した．

Ninf システムは，サーバ・クライアントモデルに基づいており，クライアントは Ninf サーバ上に予め用意されている計算ルーチンを RPC で呼び出すことにより計算を実行する．評価では，実測と同様に Ninf クライアントから Ninf_call を重複しないように発行して各 Q_{ns}, Q_s, Q_{nr} での所要時間を測定し，通信スループットと性能を算出した．また，シミュレーションでは次のパラメータの値を変化させた．

- 問題サイズ n
- 期待される通信スループット T_{act}
- 論理パケットサイズ W_{packet}

ここで， n は Ninf_call の際の通信量及びサーバでの演算数を決定するパラメータである．異なる通信量と演算数および T_{act} を設定しても， Request の通信データおよびジョブが期待される通信スループットとサーバでの計算性能を示すかどうか調査する．ただし， T_{act} には実測での評価実験[16)] において計測された各通信スループットを入力値として与える．また， W_{packet} はシミュレーションでの通信データの粒度を左右するパラメータである． W_{packet} を大きくすると Request のデータに対する論理パケット数が少なくなり，通信スループットが適切に表現できないおそれがあるが， W_{packet} を小さくすると論理パケット数が非常に多くなり，シミュレーションコストが大きくなるという問題があり，それらの影響を調べる．

4.1 シミュレーション環境

シミュレーションでは，実測と同様にクライアントとサーバが 1 対 1 の単一クライアントを用いた評価と， 4 つのサイトに分散した 4 つのクライアントと 1 つのサーバを用いた複数クライアントによる評価を行う．

クライアントのサイトにはお茶の水女子大学（Ocha-U），東京大学（U-Tokyo），名古屋工業大学（NITech），東京工業大学（TITech）を想定し，単一クライアントの評価では Ocha-U に 1 台のクライアントを設定し，複数クライアントによる評価では，各サイトに 1 台ずつクライアントを設定した． Ninf サーバには，電子技術総合研究所（ETL）の Cray J90 （4PE 構成）を想定し，計算性能は 500[Mflops]， FCFS で全てのジョブをデータ並列に処理することを前提とする．また， Ninf サーバの平均稼働率 U は実測時の J90 の稼働率を考慮して 0.04 とした．

各クライアントは， Linpack のリモートライブラリを繰り返し呼び出すものとした． Linpack はガウスの消去法を用いて密行列の連立一次方程式を求解する．リモート実行では LU 分解，後退代入を Ninf サーバ上で実行した．演算数は $2/3n^3 + 2n^2$ ，引数を含めた通信量は $8n^2 + 20n + O(1)$ [byte] である．

以下にシミュレーションで用いたパラメータを示す．このうち，実測値と同じ条件で設定したパラメータには下線を引いた．

クライアント

- Linpack の問題サイズ n : $\underline{n} = 600, 1000, 1400$
- Request の発行率 $\lambda_{request}$:
  $\underline{\lambda_{request}} = 1$ / ([Request の所要時間の実測値] + $interval$) ($interval$ は各クライアントでの Request が重複しないように単一クライアントでは 20[sec] とし，複数クライアントでは実測との比較のため，5[sec] とした) でレギュラー到着．
- Request のデータの論理パケット :
  サイズ $W_{packet} = 10, 50, 100$[KB] で試行の間固定．到着率 λ_{packet} はそれぞれ 150.0, 30.0, 15.0 (式 **(1)** より) でポアソン到着．

ネットワーク

- ネットワークでの処理 :
  FCFS でバンド幅 $T_{net} = 1.5$[MB/s] ．
- 外乱のデータのパケット :
  平均データサイズは W_{packet} と同様に指数分布に従う．λ_{ns_others}, λ_{nr_others} は式 **(2)** に実測の通信スループット を与えて決定する．ポアソン到着．

サーバ

- サーバでの処理 :
  FCFS で性能 $\underline{T_{ser}} = 500$[Mflops] ．
- 外乱のジョブ :
  平均演算数は 10[Mflop] で指数分布に従う．λ_{s_others} は稼働率 $\underline{U} = 0.04$ となるよう式 **(4)** で決定する．ポアソン到着．

シミュレーションは異なる乱数系列を用いて 30 回行い，その平均値をシミュレーション値とした．シミュレーション値はすべて信頼レベル 90% において信頼区間 ±10% 未満であった[13]．以後，本論文中のシミュレーション値はすべて信頼レベル 90%，信頼区間 ±10% 未満とする．

4.2 シミュレーション結果

Ocha-U - ETL 間での単一クライアントを用いた評価結果を表 **2**に示す．表中の n は問題サイズ，W_{packet} は Request のデータの論理パケットサイズ [KB] ，T_{sim} ，P_{sim} はシミュレーションで観測された通信スループットと性能，T_{act} ，P_{act} は実測の通信スループットと性能をそれぞれ [KB/s] ，[Mflops] で示す．

表 **2**より，各問題サイズ n において論理パケットサイズ W_{packet} が大きくなると，シミュレーション値が実測値よりやや低下しているものの，異なる W_{packet} でも通信スループット T_{sim} と T_{act} ，性能 P_{sim} と P_{act} はそれぞれほぼ等しい結果が得られた．W_{packet} を大きくするとシミュレーションの精度が下がることが危惧されたが，W_{packet} を 100[KB] としてシミュレーションコストを削減しても，遜色のないシミュレーションが行えることが示された．また，n が異なる場合，すなわち通信データ量が異なる場合でも，T_{sim} は T_{act} はそれぞれほぼ同程度のスループットを示した．

表 **3**に論理パケットサイズを 100[KB] としたときの，複数サイト，複数クライアントを想定したシミュレーション結果を示す．$W_{packet} = 100$[KB] であるため，表 **2**同様に通信スループット，性能は実測値よりやや低いものの，ほぼ同程度のシミュレーション値が得られ，本シミュレータにより文献 16) の広域ネットワークでの複数クライアントを用いた評価実験を再現することができた．また，各サイトへのネットワークに対して異なる T_{act} を想定しても，実測とシミュレーションでほぼ同じ通信スループットを示した．よって，本シミュレーション手法では各クライアントとサーバ間の通信スループットが異なる場合においても，信頼性のある挙動を示すことが分かった．

5. シミュレーションによるスケジューリング手法の評価

本シミュレータを用いたグローバルコンピューティングシステムにおけるスケジューリング手法の評価例として，以下の基本的なスケジューリング手法を用いる．

ラウンドロビン : RR　Request に対して，サーバを順番にラウンドロビンで割り当てる．

負荷 + 計算性能 : LOAD　サーバの性能 P と平均負荷 L，即ち現在処理されているジョブと実行待ちのジョブ数に関する情報を取得し，

$$P/(L+1)$$

が最大となるサーバを割り当てる．$L+1$ は，クライアントのジョブがそのサーバに投入された時の負荷を想定している．

負荷 + 計算性能 + 通信スループット : LOTH　サーバの性能 P と平均負荷 L，ネットワークの通信スループット T に関する情報を取得し，

$$\text{演算数}/(P/(L+1)) + \text{転送量}/T$$

が最小となるサーバを割り当てる．

LOAD, LOTH では，資源情報 DB で管理されている情報をもとに，サーバが決定される．シミュレーションでは，60[sec] ごとにサーバ，ネットワークの状況をモニタし，資源情報 DB の内容を更新した．

5.1 スケジューリング手法の評価環境

シミュレーションによるスケジューリングの評価環境を図 **4**に示す．サーバおよびクライアントはそれぞれ異なるサイトに分散した，サーバ 2 × クライアント 4 の

表 2 単一クライアントによるシミュレーション結果 (OCHA-U - ETL 間)
Table 2 Single-client benchmark results by simulation (OCHA-U - ETL).

	シミュレーション値			実測値	
n	W_{packet}[KB]	T_{sim}[KB/s]	P_{sim}[Mflops]	T_{act}[KB/s]	P_{act}[Mflops]
600	10	160.808	7.919	161	7.68
	50	157.743	7.771		
	100	153.038	7.543		
1000	10	131.115	10.698	131	10.50
	50	129.486	10.568		
	100	127.776	10.431		
1400	10	146.689	16.553	147	16.42
	50	146.008	16.479		
	100	145.653	16.440		

表 3 複数クライアントを想定したシミュレーション結果 ($W_{packet} = 100$[KB])
Table 3 Multi-client benchmark results by simulation for $W_{packet} = 100$[KB].

		シミュレーション値		実測値	
n	UNIV.	T_{sim}[KB/s]	P_{sim}[Mflops]	T_{act}[KB/s]	P_{act}[Mflops]
600	Ocha-U	123.473	6.102	133	6.41
	U-Tokyo	179.588	8.825	195	9.32
	NITech	84.859	4.210	95	4.36
	TITech	28.963	1.445	37	1.83
1000	Ocha-U	108.829	8.908	118	9.58
	U-Tokyo	163.020	13.225	173	13.53
	NITech	98.401	8.068	107	8.57
	TITech	30.597	2.538	37	3.07
1400	Ocha-U	130.255	14.724	135	15.05
	U-Tokyo	225.171	24.914	232	25.39
	NITech	130.480	14.743	135	15.11
	TITech	25.964	3.011	28	3.20

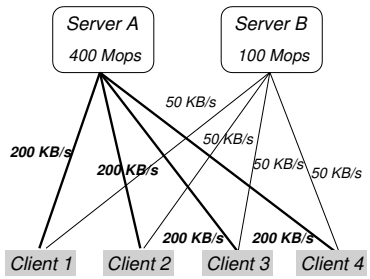

図 4 シミュレータによる評価環境
Fig. 4 Benchmarking environment by simulation

構成とする．$ServerA$ はサーバの計算性能およびクライアントからのネットワークの通信スループットが高く，$ServerB$ は計算性能，通信スループットが低いものとした．

各クライアントでは，Linpack 及び NPB[17] の EP ベンチマークプログラムを繰り返し呼び出す．EP は一様乱数 / 正規乱数を生成するプログラムで，リモート実行では通信量が $O(1)$，演算数は問題サイズ n に対し 2^{n+1} となっている．Linpack はリモート実行では Request の所要時間に対して通信時間が主体となるアプリケーションであるのに対し，EP は通信量が極めて少なく，計算時間が主体となるアプリケーションである．

シミュレーションで用いたパラメータを以下に示す．

クライアント

- 問題サイズ：Linpack $n = 600$，EP $n = 21$
- Request の発行率 $\lambda_{request}$:
  $\lambda_{request}$= 1/ ([Request の所要時間] + $interval$)
  (ここで Request の所要時間は $ServerB$ に 1 回 Request を発行した際に期待される応答時間とし，$interval$ は計算サーバでの計算時間を考慮し，Linpack:5[sec]，EP:20[sec] とした．) でポアソン到着．
- Request のデータの論理パケット :
  サイズ $W_{packet} = 100$[KB] は試行の間固定．ポアソン到着．

ネットワーク

- ネットワークでの処理 :
  FCFS でバンド幅はすべて $T_{net} = 1.5$[MB/s]．
- 外乱のデータのパケット :
  サイズは平均 W_{packet} の指数分布で決定．ポアソン到着．

サーバ

- サーバでの処理： FCFS で性能は図 4のとおり．

- 外乱のジョブ：
  演算数は平均 10[Mflop] で指数分布で決定．λ_{s_others} は稼働率 0.10 となるよう式 (4) で決定．ポアソン到着．

1 回のシミュレーションに対して全クライアントからの Request の総数が 50 回になるまでシミュレーションを行う．これを異なる乱数系列を用いて 50 回繰り返し，その平均値をシミュレーション値とした．

5.2 スケジューリング手法の評価結果

表 4にシミュレーションによるスケジューリング手法の評価結果を示す．各項目はそれぞれ平均計算 / 通信 / 所要時間，計算サーバの稼働率である．

Linpack を用いたシミュレーションでは，RR が最も悪い性能を示している．これは，$ServerA$ と $ServerB$ に交互に Request が発行されたために，遅いネットワークに接続された $ServerB$ への Request における通信時間が非常に長くなったためである．LOAD は RR より良い性能を示したものの，LOTH と比較して平均所要時間は長くなっている．所要時間のうち通信時間が主体となるアプリケーションでは，ネットワークが飽和して，通信時間が長くなったためである．よって，文献 16) で報告したように，高性能広域計算のスケジューリングでは，サーバの負荷や計算性能のみを考慮するだけでは不十分であることが本シミュレーションからも確認できた．LOTH では適切にスケジュールされていることが分かる．これは，RR や LOAD とは異なり，ネットワーク，サーバの各資源情報を適切に考慮してスケジューリングを行ったため，良い結果が得られたと考えられる．

EP を用いたシミュレーションでは，LOTH，LOAD で非常に効率良くスケジューリングが行われている．これは通信量が極めて少なく計算が主体となるアプリケーションに対し，高速な $ServerA$ を選択したためである．よって，計算が主体となるアプリケーションであれば，LOAD でも適切にサーバを割り当てられることが分かる．一方，RR では非常に所要時間が長くなっている．これは，サーバの稼働率が示しているように $ServerB$ の計算能力が飽和して，計算時間が長くなったためである．

6. まとめと今後の課題

本稿では，グローバルコンピューティングシステムにおけるスケジューリング手法を評価するためのシミュレータの設計および実装と，本シミュレータを用いたスケジューリング手法の性能評価について述べた．

グローバルコンピューティングシステムにおけるスケジューリングの評価では，ネットワークや計算サーバ等，様々な広域ネットワーク上の資源構成を想定し，大規模かつ再現性のある評価が必要である．本シミュレーションモデルでは，グローバルコンピューティングシステムを構成するネットワークおよび計算サーバを待ち行列を用いて表現することにより，これらの条件を備えた評価を可能とするシミュレータを実装した．本シミュレータを用いた実際のグローバルコンピューティングシステムを例としたシミュレーションでは，実測値とほぼ等しいシミュレーション結果が得られ，シミュレーションモデルの有効性を確認した．

また，本シミュレータによる基本的なスケジューリング手法の性能評価では，資源情報を適切に考慮したスケジューリングが非常によい性能を示すことが分かった．さらに，計算主体のアプリケーションでは LAN 内で実現されるような負荷と計算性能のみを考慮したスケジューリングでも効率良くサーバを割り当てられるのに対し，通信主体の問題ではネットワークがボトルネックとなって適切なスケジューリングが行えないという実測[16] での考察を反映した結果が再現できた．

今後は，シミュレーションによる評価の信頼性を向上させるためにシミュレーションモデルの改善を行う．本シミュレーションでは，ネットワークの外乱データのモデルとしてポアソン到着を想定したが，広域ネットワークでの TCP/IP 通信はポアソン到着のみでは表せないことが報告されている[18]．よって，現在提案されている他の通信モデルを十分に調査し，様々なモデルを適用していくとともに，実際のネットワークにおける通信スループットのトレースを利用してそれを本シミュレータ上で再現することにより，より正当な評価を行う．また，計算サーバ上のジョブの処理方式として，FCFS 以外の方式を用いた場合等の評価を進める予定である．さらに，他のグローバルコンピューティングシステムで実現されているスケジューリング手法との比較と，その性能特性を抽出するために，様々なアプリケーションを想定した評価を行い，グローバルコンピューティングシステムにおける最適なスケジューリング手法を検討する．

参 考 文 献

1) Ninf. http://ninf.etl.go.jp/.
2) NetSolve. http://www.cs.utk.edu/~casanova/NetSolve/.
3) Globus. http://www.globus.org/.
4) Legion. http://www.cs.virginia.edu/~legion/.
5) RCS. http://www.inf.ethz.ch/personal/arbenz/rcs.html.
6) Christiansen, B. O., Cappello, P., Ionescu, M. F., Neary, M. O., Schauser, K. E. and Wu, D.: Javelin: Internet-Based Parallel Computing Using Java, *ACM Workshop on Java for Sci-*

表 4 シミュレーションによるスケジューリング手法の評価結果
Table 4 Simulation results of scheduling schemes.

プログラム	スケジューラ	平均計算時間 [sec]	平均通信時間 [sec]	平均所要時間 [sec]	サーバの稼働率 A(400Mflops)	サーバの稼働率 B(100Mflops)
linpack	RR	0.914	79.673	80.587	0.110	0.140
	LOAD	0.610	44.682	45.292	0.117	0.118
	LOTH	0.397	21.496	21.893	0.122	0.102
EP	RR	96.776	2.359	99.135	0.454	0.979
	LOAD	45.867	1.787	47.654	0.608	0.791
	LOTH	42.729	1.689	44.418	0.625	0.722

ence and Engineering Computation (1997).

7) Raman, R., Livny, M. and Solomon, M.: Matchmaking: Distributed Resource Management for High Throughput Computing, *Proc. 7 th IEEE International Symposium on High Performance Distributed Computing* (1998).

8) Czajkowski, K., Foster, I. and Kesselman, C.: Resource Management Architecture for Metacomputing Systems, *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing* (1998).

9) Chapin, S. J., D. Katramatos, J. K. and Grimshaw, A.: Resource Management in Legion, *submitted to Elsevier Science*.

10) Berman, F., Wolski, R., Figueira, S., Schopf, J. and Shao, G.: Application-Level Scheduling on Distributed Heterogeneous Networks, *Proc. the 1996 ACM/IEEE Supercomputing Conference* (1996).

11) Weissman, J. B. and Zhao, X.: Scheduling Parallel Applications in Distributed Networks, *Journal of Cluster Computing*, Vol.1(1) (1998).

12) Wolski, R., Spring, N. and Peterson, C.: Implementing a Performance Forecasting System for Metacomputing: The Network Weather service, *Proc. the 1997 ACM/IEEE Supercomputing Conference* (1997).

13) Jain, R.: *The art of computer systems performance analysis*, John Wiley & Sons, Inc. (1991).

14) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design patterns*, Addison-Wesley (1995).

15) Takefusa, A., Matsuoka, S., Ogawa, H., Nakada, H., Takagi, H., Sato, M., Sekiguchi, S. and Nagashima, U.: Multi-client LAN/WAN Performance Analysis of Ninf: a High-Performance Global Computing System, *Proc. the 1997 ACM/IEEE Supercomputing Conference* (1997).

16) 竹房あつ子, 小川宏高, 松岡聡, 中田秀基, 高木浩光, 佐藤三久, 関口智嗣, 長嶋雲兵: 複数クライアントによる LAN/WAN での Ninf の性能, 情報処理学会論文誌, Vol. 39, No. 6, pp. 1827–1838 (1998).

17) NPB: NAS Parallel Benchmarks. `http://science.nas.nasa.gov/Software/NPB/`.

18) Paxson, V. and Floyd, S.: Wide-Area Traffic: The Failure of Poisson Modeling, *SIGCOMM '94*, pp. 257–268 (1994).