広域計算システムのシミュレーションによる評価
—Ninf システムの広域分散環境でのジョブスケジューリング実現に向けて —

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

高性能広域計算を実現する試みが行われる一方，高性能広域計算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再現性のある大規模性能評価は非常に困難で性能要素に関する調査が不十分なためである．本稿では，高性能広域計算システムにおけるジョブスケジューリングの性能評価を行うシミュレータの設計及び実装と，本シミュレータを用いた性能評価について述べる．本シミュレータでは，待ち行列モデルを用い多様な広域計算システムをシミュレーションすることが可能である．本シミュレータにより実際の広域計算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，その有効性を確認した．本稿ではまた，本シミュレータを用いたジョブスケジューリング手法の性能評価結果についても述べる．

Performance Evaluation of Global Computing Systems by Simulation

Atsuko Takefusa,[†1] Kento Aida,[†2] Hirotaka Ogawa,[†2]
Hidemoto Nakada,[†3] 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, job 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 job scheduling schemes on the high-performance global computing system. The simulator can simulate various features on global computing systems by adopting queueing model. Effectiveness of the simulator was verified by the simulation results, which showed same results as experimental results on a real global computing system. This paper also shows simulation results of simple job scheduling schemes by the simulator.

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，科学技術計算の分野での大規模計算要求に応えることが期待されている．近年これを目的とした高性能広域計算システムが Ninf[1] を筆頭に複数提案されている[2)~6)]．このような高性能広域計算システムにおける計算サーバはネットワーク上に分散する多数のユーザにより共有される．そのため，各ユーザのリクエストに対し一定の要求性能をみたすよう適切なサーバを選択し，利用させる必要がある．一方，従来の WS クラスタとメッセージ通信ライブラリにより実現される LAN 内の分散並列処理と異なり，高性能広域計算では計算サーバの性能 / 負荷及びネットワークのトポロジ / 性能が不均質である．さらに TCP/IP で実現される広域ネットワークでの通信性能は変動が大きく予測が困難である．従って，高性能広域計算の実現には従来の LAN 環境でのスケジューリング手法とは異なる適切な広域ジョブスケジューリング手法が要求される．

高性能広域計算システム上でのジョブスケジューリングは，Ninf, NetSolve[2], RCS[5] などのシステムでその試みが行われている他，AppLeS[7]，Prophet[8] といった高性能広域計算システム上でジョブスケジューリングを専門に行うシステムも提案されている．また，通信スループットやレイテンシ等，広域ネットワーク上での資源情報を予測するためのシステムとして NWS[9] が提案されている．これらのシステムのいくつかは既に広

†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

域ネットワーク上に実装され，実システムによる実験結果が得られている．

高性能広域計算システムでの最適なジョブスケジューリング手法を検討するためには，様々な

- ネットワークトポロジおよびバンド幅
- ネットワークの混雑度及びその変動
- 各計算資源の性能
- 各計算資源の負荷及びその変動

等を想定した再現性のある大規模環境での評価が必要である．しかし，現在行われているような実システムによる実験のみでは，利用可能な計算資源やネットワークトポロジが制限されるため，様々な環境を想定した再現性のある大規模な性能評価を行うことは非常に困難である．そのため，広域計算環境での最適なジョブスケジューリング手法に関する十分な評価は行われていない．

本稿では，これらの問題を解決するため，高性能広域計算システムにおけるジョブスケジューリング手法の性能評価を行うシミュレータの設計および実装を行った．本シミュレータでは，広域計算システムを待ち行列を用いてモデル化することにより，多様な広域計算システムをシミュレートすることが可能である．本シミュレータにより実際の高性能広域計算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，実測結果とほぼ同様の結果が得られ，その有効性を確認した．本稿ではまた，本シミュレータを用いたジョブスケジューリング手法の性能評価結果についても述べる．

以後 **2**では高性能広域計算の概要について述べ，**3**で広域計算システムのモデルについて述べる．**4**では本シミュレータの基本性能の評価を行い，**5**でジョブスケジューリング手法を評価した結果を報告する．

2. 高性能広域計算の概要

ネットワーク技術の発展を前提に，広域に分散した計算機や記憶装置といった計算資源を積極的に活用して大規模計算の要求に応えることが可能となった．これを目的とした計算技術を高性能広域計算 (Global Computing) と呼ぶ．この高性能広域計算を有効に運用するための試みが現在複数行われている．

高性能広域計算システムは基本的に以下のコンポーネントにより構成される (図 **1**)．

クライアント　高性能広域計算システムのユーザへの API．

サーバ　計算ライブラリ等の資源を提供するシステム

ジョブスケジューラ　各高性能広域計算システムのポリシーに従ったジョブスケジューリングを行うシステム

ディレクトリサービス　計算サーバやサーバで提供するライブラリ等の情報の集中ストレージ

モニタ　計算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予測を行うシステム

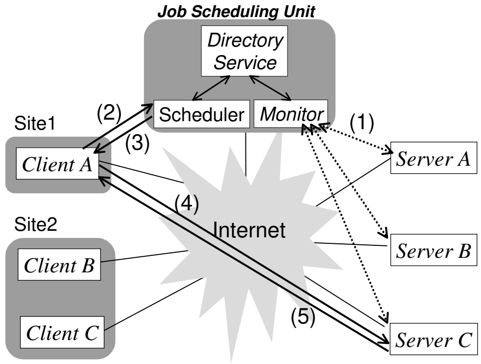

図 1　高性能広域計算の実行の流れ

高性能広域計算システムにはこれらの機能をすべて提供しているシステム (Ninf, NetSolve, RCS) と，実行環境を提供するシステム (Legion, Globus)，スケジューリングを専門に行うシステム (Prophet, AppLeS)，モニタを専門に行うシステム (NWS) がある．

具体的には高性能広域計算は図 **1**に示す手順で実行される．

(1) モニタはサーバ，ネットワークの情報をモニタしている．サーバ資源の情報はディレクトリサービズにより管理される．

(2) 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サーバをジョブスケジューラに問い合わせる．

(3) ジョブスケジューラはディレクトリサービスに問い合わせたクライアントの要求する計算とサーバ，ネットワークなどの資源情報により，適切なサーバを選定し，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計算要求を発行する．

(5) サーバは計算結果をクライアントに返す．

3. 高性能広域計算システムシミュレータ

高性能広域計算の実現には，広域ネットワークに存在する計算資源を有効に利用し，かつクライアントの計算要求の応答時間を短縮することが求められる．既に実際の広域ネットワーク上で高性能広域計算システムの実装とその評価が行われている[2),3),5),6),10)] が，広域ネットワークで再現性のある大規模な性能評価を行うことは非常に困難であり，評価で利用可能な計算資源やネットワークトポロジも限定される．また，今後の広域ネットワークの整備を前提とした高性能広域計算システムの挙動を調査する必要があり，実システムによる評価だけでは各高性能広域計算システムにおけるジョブスケジューリング手法の正当性を実証するには不十分である．

そこで，ネットワークのトポロジ / 混雑度，各計算資源の性能 / 負荷など，高性能広域計算における様々な環境を想定可能な性能評価手法が必要であり，この性能評

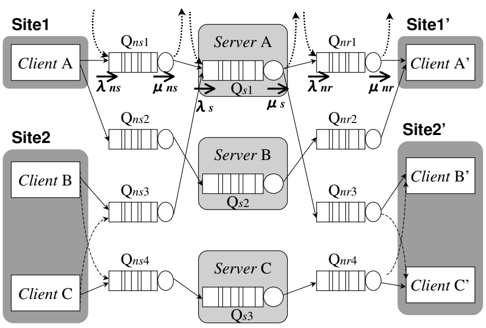

図 **2**　高性能広域計算システムのシミュレーションモデル

価手法として，高性能広域計算シミュレータを設計，実装した．本シミュレータを用いることにより，高性能広域計算システムを実現するための問題点を抽出するとともに，高性能広域計算を実行する際の広域ジョブスケジューラによる最適な計算サーバの選択（ジョブスケジューリング）手法等を検討することができる．

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，高性能広域計算システムはジョブスケジューラ，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）により構成される．このうちサーバ，ネットワークは図**2**に示すような待ち行列モデルを用いて表現する．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表す．同一サイトのクライアントから同じサーバにデータを転送する場合，同じネットワークを共有すると考えられる．従って，同一サイトのクライアントから同じサーバにデータが転送される場合は同じネットワークの待ち行列にデータが転送されるものとする．

高性能広域計算システムでは，広域ネットワーク上に対象とする高性能広域計算システム以外のシステムから転送されるデータやその計算サーバと同一マシン上で実行されている高性能広域計算以外のジョブが存在し，これらが高性能広域計算要求（以後 Request とする）の応答時間に大きな影響を与える．従って，シミュレーションではこれらの高性能広域計算システム以外から発生するデータやジョブ（外乱）を想定できる必要がある．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 $\lambda_{ns}, \lambda_{nr}, \lambda_s$ は，それぞれ Q_{ns}, Q_{nr}, Q_s に到着する Request と外乱によるデータまたはジョブを合わせた到着率であり，

$$\lambda_{ns} = \lambda_{ns\_request} + \lambda_{ns\_others} \quad (1)$$
$$\lambda_{nr} = \lambda_{nr\_request} + \lambda_{nr\_others} \quad (2)$$
$$\lambda_{s} = \lambda_{s\_request} + \lambda_{s\_others} \quad (3)$$

の関係がある．ここで $\lambda_{ns\_request}$ は Request のデータの論理パケットの到着率, λ_{ns_others} は Request 以外のジョブ（データの論理パケット）の到着率を表す．$\lambda_{nr\_request}, \lambda_{nr\_others}, \lambda_{s\_request}, \lambda_{s\_others}$ 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 によってサーバ上で実行される計算ライブラリ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のデータ量を意味する．

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を以下に示す．

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 に適切なサーバを問い合わせる．サーバが決定すると計算に必要なサイズ W_{ns} のデータを論理パケット単位に分割し，λ_{packet} の確率で Q_{ns} に論理パケットを投入していく．サーバで処理が終了すると，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 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}[byte/s]，論理パケットサイズを W_{packet}[byte] とすると以下のように定める．

$$\lambda_{packet} = T_{net}/W_{packet} \quad (4)$$

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[11),12)] を採用する．Q_{ns} に M/M/1/N 待ち行列モデルを用いる利点は，

- λ_{ns_others} を適切に設定することにより，実際の広域ネットワークの特性（通信スループット）を比較的容易に表現できる.
- TCP/IP によるデータ送出時のデータの再送を表現できる．

等である．

Q_{ns} にはクライアントから発行された論理パケットと，外乱のデータが到着する．この際，待ち行列バッファが一杯ならば外乱データは破棄され，Request の論理パケットは再送される．Q_{ns} に入った Request の論理パケットは Q_{ns} の待ち行列サーバ上で [論理パケットサイズ / ネットワークのバンド幅] 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上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平均パケット数が多くなるため，Q_{ns} に到着した Request のパケットがバッファに入り切らずに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，ネットワーク上で転送される Request の通信スループットをシミュレーションで表現可能であることを意味する．

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．

$$\frac{\lambda_{packet}}{\lambda_{ns\_others} + \lambda_{packet}} = \frac{T_{act}}{T_{net}}$$
$$\Leftrightarrow \lambda_{ns\_others} = (\frac{T_{net}}{T_{act}} - 1) \times \lambda_{packet} \qquad (5)$$

ここで λ_{packet} は式 (4) により決定する．

例えば，$T_{net} = 1.0[\mathrm{MB/s}]$, $W_{packet} = 0.1[\mathrm{MB}]$ という条件で，$T_{act} = 0.1[\mathrm{MB/s}]$ となるような状況をシミュレーションで表現するには，

$$\lambda_{packet} = 1.0/0.1 = 10 \qquad (6)$$
$$\lambda_{ns\_others} = (1.0/0.1 - 1) \times 10 = 90 \qquad (7)$$

とすればよい．

また，このように λ_{ns_others} を決定すると，Q_{ns} における平均バッファ長は有限バッファ長 N とほぼ等しくなる．従って，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．本シミュレーションではこれを広域ネットワークにおける平均通信遅延 $T_{latency}$ とし，以下のように有限バッファ長 N を定める．

$$\frac{W_{packet} \times N}{T_{net}} \simeq T_{latency}$$
$$\Leftrightarrow N \simeq \frac{T_{latency} \times T_{net}}{W_{packet}} \qquad (8)$$

ただし，$N = 1$ のときの待ち行列の挙動が非常に不安定であるため，$N \geq 2$ とする．

3.1.3 サーバ

本シミュレーションでは，サーバ上のジョブは，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[11),12)] を用いる．ただし，サーバでのジョブ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，Q_s は対応する待ち行列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 Q_s のサーバ上で [外乱のジョブの演算数 /T_{ser}] 時間処理された後，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 $U[\%]$ のとしたときの λ_{s_others} を以下のように決定する．

$$\lambda_{s\_others} = \frac{T_{ser}}{W_{s\_others}} \times \frac{U}{100} \qquad (9)$$

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_{nr} のサーバ上で [パケットサイズ / ネットワークのバンド幅] 時間で処理され，クライアントに転送される．λ_{nr_others} は λ_{ns_others} と同様に決定する．

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[13)] で実装しているため，

- クライアント，サーバ，ネットワークの構成
- 広域ジョブ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)
- 乱数分布

等を容易に組み変えることができる．

また，Java で記述することにより，以下の利点が得られる．

- ガベージコレクションの機能により，メモリ管理の必要がない．
- 各待ち行列での外乱のデータの到着間隔など，個々のオブジェクトに対して独立の乱数系列が指定できるため，柔軟なシミュレーションセッティングが可能となる．

Java は C++ 等の言語と比較すると実行速度が低速ではあるが，C 言語への変換や JIT の使用が可能なため，実用上は問題ない．

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**に示す実際の高性能広域計算システムを例にしたシミュレーションを行った．図中の括弧内には各クライアントとサーバ間の FTP スループット，ping レイテンシが示されている．なお，図**3**に示す環境については，実測による実行結果が得られているため[10)]，シミュレートする高性能広域計算システムとして Ninf システムを想定する．

評価では，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，

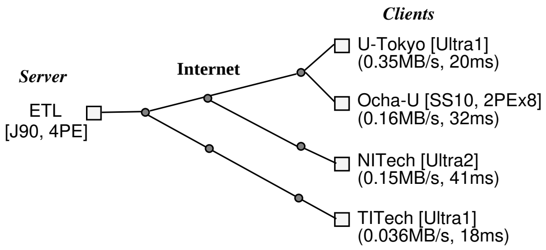

図 3　高性能広域計算の計測環境

- 問題サイズ
- 論理パケットサイズ
- 期待される通信スループット

が異なる条件でのシミュレーション結果を示す．

4.1　シミュレーション環境

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]，First Come First Served(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は，実測時の J90 の稼働率を考慮して 4% とした．

クライアン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，Ocha-U を用いた単一クライアントのシミュレーションと，4 つのサイトに分散した複数クライアントによるシミュレーションを行う．クライアント・サーバ間のバンド幅 T_{net} はいずれも 1.5[MB/s] とした．

クライアントのプログラムのモデルは，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し，リモート実行では LU 分解，後退代入を計算サーバ上で実行する．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．

クライアント
- Linpack の問題サイズ n：
  $n = 600, 1000, 1400$
- Request の発行率 $\lambda_{request}$：
  $\lambda_{request}$= 1 / （[期待される Request の所要時間] + *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]，複数クライアントでは 5[sec] とする），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]．W_{packet} は試行の間固定．到着率 λ_{packet} はそれぞれ 150.0, 30.0, 15.0（式 **(4)** 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：
  バンド幅 $T_{net} = 1.5$[MB/s] で FCFS．
- 外乱のデータのパケット：
  平均データサイズは W_{packet} と同じで指数分布に従う．$\lambda_{ns\_others}, \lambda_{nr\_others}$ は式 **(5)** で決定．ポアソン到着．

サーバ
- サーバでの処理：
  性能 $T_{ser} = 500$[Mflops] で FCFS．
- 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 4[%] となるよう式 **(9)** 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の seed 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 95[%] において信頼区間 ±10[%] 未満であった[12)]．

4.2　シミュレーション結果

Ocha-U - ETL 間でクライアント，サーバを 1 対 1 としたときのシミュレーション結果を表 **1** 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} は通信スループットと性能の実測値を示す．

表 **1** 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，T_{sim} と T_{act}，P_{sim} と P_{act} はそれぞれほぼ等しい結果が得られている．これは 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実行可能であることを示している．また，問題サイズが異なる場合においても T_{sim} は T_{act} はそれぞれほぼ同程度のスループットとなっていた．

表 **2** に論理パケットサイズを 100[KB] のときの，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 **1** 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 10) の評価実験を再現することができた．また，異なる T_{act} を想定しても実測値とシミュレーションがほぼ同じ通信スループットを示したことから，本シミュレーションの手法は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る．

5.　ジョブスケジューリング手法の評価

本シミュレータを用いた高性能広域計算システムにおけるジョブスケジューリング手法の評価例として，以下の基本的なジョブスケジューリング手法の評価結果を示す．

各クライアントで **Round Robin：** LRR　各クライアントはそれぞれ順番にサーバを選択し，Request をサーバに投入する．今回は最悪のケースを

表 1　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)

	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]
	10	159.947	7.998		
600	50	158.169	7.910	161	7.68
	100	155.188	7.760		
	10	130.436	10.873		
1000	50	129.051	10.757	131	10.50
	100	128.124	10.680		
	10	146.672	17.115		
1400	50	146.520	17.097	147	16.42
	100	145.568	16.986		

表 2　$W_{packet} = 100$[KB] のときの複数クライアントを想定したシミュレーション結果

		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]
	Ocha-U	122.372	6.121	133	6.41
600	U-Tokyo	182.928	9.147	195	9.32
	NITech	85.291	4.266	95	4.36
	TITech	29.743	1.488	37	1.83
	Ocha-U	104.670	8.725	118	9.58
1000	U-Tokyo	159.815	13.321	173	13.53
	NITech	91.476	7.625	107	8.57
	TITech	30.497	2.543	37	3.07
	Ocha-U	124.426	14.519	135	15.05
1400	U-Tokyo	218.441	25.487	232	25.39
	NITech	126.715	14.787	135	15.11
	TITech	25.879	3.020	28	3.20

考慮して，すべてのクライアントが同じサーバから順番に選択することにした．

広域スケジューラで **Round Robin：** GRR　各クライアントは Request を発行すると，広域のスケジューラにジョブを投入するサーバを問い合わせる．スケジューラは要求のあったクライアントから順番にサーバをラウンドロビンで割り当てる．

広域スケジューラで負荷+計算性能によるサーバの選択 **：** LOAD　各クライアントは Request を発行すると，広域のスケジューラに投入先のサーバを問い合わせる．スケジューラはクライアントからの問い合わせがあると，サーバの平均負荷，即ち現在処理されているジョブと実行待ちのジョブ数を調べ，[(サーバの平均負荷 $+ 1$)/ サーバの性能] が最小となるサーバをクライアントへ割り当てる．ここで，$+ 1$ はクライアントのジョブが実際にそのサーバに投入された時の負荷を想定している．

5.1　シミュレーションによるジョブスケジューリングの評価環境

シミュレーションによるジョブスケジューリングの評価環境を図 **4,5**に示す．図 **4**は均質な評価環境，図 **5**は不均質な評価環境である．サーバ及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 $2 \times$ クライアント 4 の構成となっている．均質な環境では，計算性能は 160[Mops]，通信スループットは 80[KB/s] とした．不均質な環境では，$ServerA$ は計算性能が高い (400[Mops]) が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小さく (50[KB/s])，$ServerB$ は計算性能は高くない (100[Mops]) が，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比較的高い (200[KB/s]) ものとする．また，クライアント・サーバ間のネットワークバンド幅 T_{net} はすべて 1.5[MB/s] とした．

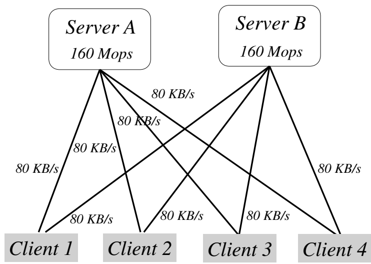

図 4　均質なシミュレーション環境

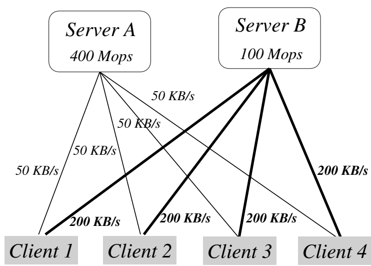

図 5　不均質なシミュレーション環境

各クライアントは，Linpack 及び NASPB[14)] の EP ベンチマークプログラムを繰り返し呼び出すものとする．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} と定められる．

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 $\lambda_{request}$：
  $\lambda_{request} = 1$ /（[最悪のケースの Request の所要時間] + $interval$）（$interval$ は Linpack:5[sec], EP:20[sec]），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]．試行の間固定．到着率 λ_{packet} は式 (4) で決定．ポアソン到着．

ネットワーク

- ネットワークでの処理：
  バンド幅はすべて $T_{net} = 1.5$[MB/s] で FCFS．
- 外乱のデータのパケット：
  平均データサイズは W_{packet} と同じで指数分布に従う．$\lambda_{ns\_others}, \lambda_{nr\_others}$ は式 (5) で決定．ポアソン到着．

サーバ

- サーバでの処理：
  FCFS．
- 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} はすべて稼働率 10[%] となるよう式 (9) で決定する．ポアソン到着．

1 回のシミュレーション (replication) 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い，それを異なる乱数の seed を用いて 30 回繰り返した．30 回の replication の平均値をシミュレーション値とした．このシミュレーション値は信頼レベル 95[%] において信頼区間 ±10[%] 未満であった[12)]．

5.2 シミュレーションによるジョブスケジューリングの評価結果

表 3にクライアントから Linpack の Request を繰り返し呼び出した場合のシミュレーション結果を示す．Linpack は演算数に対し通信量が大きいため，Request の所要時間のほとんどを通信時間が占めている．

均質な環境では LOAD がわずかに良い性能を示したものの，LRR, GRR との差異はあまり見られなかった．これはネットワークが干渉しない環境で Linpack の Request が多数発生しているため，サーバの負荷が上がらず，それぞれのスケジューリングの差が現れなかったと考えられる．

不均質な環境では，LOAD は LRR, GRR と比較して平均所要時間が非常に長くなった．これは 10) で報告したように，広域環境では通信主体の問題が計算サーバに投入されても，ネットワークが飽和してしまうことによりサーバの負荷が上がらないためである．従って高性能広域計算のジョブ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．

表 4に EP のルーチンを繰り返し呼び出した場合の実行結果を示す．EP は計算主体の問題であるため，Request の所要時間のほとんどを計算時間が占める．表 4において，通信時間が比較的大きい値を示しているが，これはシミュレーションコストを削減するために Request のデータの論理パケットサイズを 100[KB] としたためである．

均質な環境では，GRR, LRR, LOAD の順に良い性能を示した．LRR では各クライアントの Request が同じサーバに投入される確率が高くなるために GRR より性能が低下していた．LOAD が一番悪い性能を示したのは，均質な環境であるために GRR, LRR で比較的適切な負荷分散が行われていたためである考えられる．

不均質な環境では，LRR, GRR と比較すると LOAD では非常に効率良くジョブスケジューリングが行われている．これは高速なサーバ A を有効に選択したためであり，このことから，計算が主体となっているアプリケーションでは LOAD によるジョブスケジューリングで適切にサーバを割り当てられることが分かる．また，LRR と GRR では LRR の方が Request の所要時間が長かった．これは LRR では同じタイミングでサーバにジョブが投入される確率が高かったためである．

6. まとめと今後の課題

本稿では，高性能広域計算システムにおけるジョブ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ジョブスケジューリング手法の性能評価について述べた．

高性能広域計算システムにおけるジョブスケジューリングの性能評価では，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を想定し，大規模かつ再現性のある評価を行うことが必要である．本シミュレータにおけるシミュレーションモデルでは，広域計算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モデルを用いて表現することにより，これらの条件を備えたシミュレーションを行うことが可能である．

本シミュレータを用い，実際の広域計算システムを例としたシミュレーションを行ったところ，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正当性，有効性を確認された．また本シミュレータを用いた基本的なジョブスケジューリング手法の性能評価では，通信主体の問題では負荷と計算性能のみによりスケジューリングを行うと，Round Robin のような単純なスケジューリングよりも平均実効性能を悪化

表 3　Linpack を用いたジョブスケジューリングのシミュレーション結果

環境	スケジューラ	平均通信時間 [sec]	平均計算時間 [sec]	平均所要時間 [sec]	サーバの稼働率 A[%]	B[%]
均質	LRR	76.503	0.011	76.514	10.020	9.924
	GRR	76.910	0.011	76.921	9.992	9.941
	LOAD	75.407	0.012	75.419	10.000	9.934
不均質	LRR	80.310	0.012	80.322	10.029	9.955
	GRR	80.442	0.012	80.454	10.031	9.960
	LOAD	108.661	0.014	108.675	10.031	9.959

表 4　EP を用いたジョブスケジューリングのシミュレーション結果

環境	スケジューラ	平均通信時間 [sec]	平均計算時間 [sec]	平均所要時間 [sec]	サーバの稼働率 A[%]	B[%]
均質	LRR	2.582	83.036	85.618	87.395	92.549
	GRR	2.591	76.161	78.752	89.988	91.119
	LOAD	2.482	89.613	92.095	88.245	88.833
不均質	LRR	2.289	108.005	110.294	45.172	98.128
	GRR	2.290	97.522	99.812	46.014	97.741
	LOAD	1.812	32.685	34.497	63.671	79.117

させてしまうことが分かった．一方で，計算主体のアプリケーションに関しては 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効率良くサーバを割り当てられることを確認した．

今後は，実際のネットワークにおける通信スループット等の変動を考慮した評価を行う．本シミュレーションでは通信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，WAN での TCP/IP トラフィックはポアソン到着のみでは表せないことが報告されており，自己相似性モデル等の通信モデルを適用するつもりである[15]．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，シミュレーションモデルの信頼性をさらに向上させる予定である．さらに，本シミュレータを用いて他の高性能システムで実現されているジョブスケジューリング手法との比較を行うとともに，高性能広域計算システムにおける最適なジョブスケジューリング手法を検討する．

参 考 文 献

1) Ninf: Network Infrastructure for Global Computing. http://ninf.etl.go.jp/.
2) Casanova, H. and Dongara, J.: NetSolve: A Network Server for Solving Computational Science Problems, *Proc. of Supercomputing '96* (1996).
3) Grimshaw, A. S., Wulf, W. A., French, J. C. and Alfred C. Weaver, P. F. R. J.: Legion: The Next Logincal Step Toward a Natiowide Virtual Computer, CS94-21, University of Virginia (1994).
4) 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 Science and Engineering Computation* (1997).
5) Arbenz, P., Gander, W. and Oettli, M.: The Remote Computation System, Technical Report 245, ETH (1996).
6) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications (to appear)* (1997).
7) Berman, F., Wolski, R., Schopf, S. F. J. and Shao, G.: Application-Level Scheduling on Distributed Heterogeneous Networks, *Proc. of Supercomputing '96* (1996).
8) Weissman, J. B. and Zhao, X.: Scheduling Parallel Applications in Distributed Networks, *Journal of Cluster Computing (accepted)*.
9) Wolski, R., Spring, N. and Peterson, C.: Implementing a Performance Forecasting System for Metacomputing: The Network Weather service, *Proc. of Supercomputing '97* (1997).
10) 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, *Supercomputing '97* (1997).
11) 森村英典, 大前義次: 応用待ち行列理論, 日科技連 (1975).
12) Jain, R.: *The art of computer systems performance analysis*, John Wiley & Sons, Inc. (1991).
13) Gamma, E., Helm, R., Johnson, R. and Vlissides, J.: *Design patterns*, Addison-Wesley (1995).
14) NPB: NAS Parallel Benchmarks. http://science.nas.nasa.gov/Software/NPB/.
15) Paxson, V. and Floyd, S.: Wide-Area Traffic: The Failure of Poisson Modeling, *SIGCOMM '94*, pp. 257–268 (1994).