Ninf システムにおけるジョブスケジューラの実装と予備的評価

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

ネットワーク技術の発展により，グローバルコンピューティングシステムが複数提案されているが，その計算リソースを有効に利用するための議論が十分になされていない．特に，広域ネットワーク経由の計算固有の変動や不安定さを考慮して，計算リソースを適切に割り当てるスケジューリング手法が必要である．本稿では，グローバルコンピューティングにおい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を統合するための階層化されたスケジューリングフレームワークを提案するとともに，それに準拠した Ninf システムのメタサーバスケジューリングフレームワークの実装について述べる．さらに，実環境と性能評価モデルを用いてスケジューリング手法の評価を行い，グローバルスケジューリングにおける課題を明らかにした．

Implementation and Preliminary Evaluation of Global Scheduling Framework in Ninf

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

Rapid progress in networking technology is now making global computing systems feasible. Although there have been proposals of global computing systems, it is still a research issue as to how to achieve efficient usage of computing resources in global computing. In particular, we need to devise appropriate scheduling strategies/algorithms of computing resources over wide-area networks, which are often dynamic and unstable in nature. This paper presents our preliminary scheduling framework for unifying application and job scheduling in global computing. The proposed framework establishes a layer of scheduling and resource allocation subframeworks. We show our software framework Ninf metaserver which provides low-level scheduler and resource monitor. We also evaluate some scheduling strategies by actual environment and our performance evaluation model.

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)~5)]．一方，グローバルコンピューティングにおいて計算資源を有効利用するための議論が十分になされていない．特に，不安定かつ変動する広域ネットワーク環境では，適切なスケジューリング手法の実現が必要不可欠である．

グローバルコンピューティングシステムでのスケジューリング技術に関する過去の研究は，**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**の 2 つに分けられる．アプリケーションスケジューリングは，単一のプログラムの応答時間を短縮することを目的とし，アプリケーションプログラムの特性にしたがってプログラムを複数並列タスクに分割してタスクをスケジューリングする．一方，ジョブスケジューリングではグローバルコンピューティングシステム全体の複数ジョブの総実行時間 (スループット) を向上させることを目的として，各ジョブをスケジューリングする．

アプリケーションスケジューリングは AppLeS[6], Prophet[7] などに代表されるが，これらのシステムはいずれも 1 つのクライアントプログラムのスケジューリングのみを考慮し，複数クライアントのプログラムが競合するようなジョブスケジューリングの側面は考えられていない．ジョブスケジューリングは MPP のように密に結合した環境では広く研究されてきた[8] が，計算リソースやネットワークが変動し，不安定である点でそのアルゴリズムや利用目的がグローバルコンピューティングシ

†1 お茶の水女子大学 Ochanomizu University
†2 電子技術総合研究所 Electrotechnical Laboratory
†3 東京工業大学 Tokyo Institute of Technology
†4 物質工学工業技術研究所 National Institute of Materials and Chemical Research

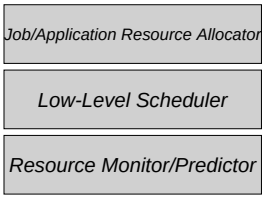

図 1 グローバルスケジューリングシステムの概要

ステムとはかなり異なる．Condor[9] や Nimrod/G[10] のように，ある種のジョブスケジューラを実現しているものもある．Condor の **Matchmaking** メカニズムはクライアントに適切なサーバを紹介する機構であるが，通常のジョブスケジューラとは異なり，計算リソースの負荷などの動的情報を考慮していない．また，Nimrod/G は Globus 上に構築されたシステムであり，ジョブのデットラインスケジューリングを行うものであるが，技術的詳細は明らかでない．

もっとも重要な点は，これまでの研究では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は協調して論じられていないことである．クライアントからの計算要求の応答時間を最短にするためにアプリケーションスケジューリングを行うことは重要であるが，グローバルコンピューティングシステムを有効利用するためには，システムを共有する複数のジョブの平均応答時間の向上を図るジョブスケジューリングが重要である．したがって，個々のアプリケーション性能とシステムの総スループットの双方をバランスよく採り入れたアルゴリズムとソフトウェアフレームワークを考慮し，アプリケーションスケジューリング，ジョブスケジューリングともに言及しなければならない．

本稿では，グローバルコンピューティングにおい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を統合するための基本的なスケジューリングフレームワークについて述べるとともに，それに準拠した Ninf システムのメタサーバスケジューリングフレームワークの実装とその評価について述べる．以後 **2** では，グローバルコンピューティングでのソフトウェアスケジューリングフレームワークについて述べるとともに，Ninf メタサーバフレームワークについて説明する．**3** では，いくつかのスケジューリング手法について，メタサーバのプロトタイプを用いた実環境での評価と，性能評価モデルを用いた評価について報告し，**4** で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考察およびまとめを述べる．

2. Network Task スケジューリングのためのソフトウェアフレームワーク

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をバランスよく採り入れるためのソフトウェアフレームワークとして，**図 1** に示すような階層化したスケジューリングとリソース割り当てのサブフレームワークを提案する．

2.1 ソフトウェアスケジューリングフレームワークの概要

グローバルコンピューティングシステムでのアプリケーションの実行は，次の 2 段階に分けられる．

(1) 単一アプリケーションを複数の並列タスク (これを Network Task とする) に分割する．
(2) Network Task をサーバへスケジューリングする．

図 1 に示すフレームワークでは，ジョブ / アプリケーションリソースアロケータでは (1) を行い，AppLeS と同様の手法でトップレベルのスケジューラとして機能する．低レベルスケジューラでは (2) を行い，サーバやネットワークの現在の負荷や予測された負荷情報を考慮し，個々の Network Task を適切な計算ノードに割り当てる．また，最下位のリソースモニタ / プレディクタは，低レベルスケジューラにリソース情報を提供する．プレディクタは NWS[11] のように単純な時系列の補間を行うのではなく，待ち行列でリソースの可用性を予測するグローバルコンピューティングの性能評価モデル[12] を用いる．

このスケジューリングフレームワークでもっとも着目すべき点は，高レベルの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の問題を低レベルの Network Task に統一化するところにある (Network Task スケジューリング)．これは低レベルスケジューラが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の要求をバランスよく採り入れ，スケジューリングを決定できることを意味する．また，グローバルコンピューティングの基本性能モデルを単純化するため，計算サーバやネットワークの負荷情報を，スケジューリング決定時の本質的なパラメータとすることができる．そのように簡易に扱うことは，グローバルコンピューティングプラットフォーム固有の不安定性の上でスケジューリングを行うには必須である．また，これは性能モデルとそのモデルに基づくシミュレータを作ることを可能とするため，ロバストな予測が可能となる．

2.2 Network Task スケジューリング

低レベルスケジューラは広域に分散した計算・通信リソースの稼働性に関する情報を収集し，その情報をもとに現在のリソースの利用状況を予測しなければならない．そのような予測を促進するため，収集された情報はすべて広域 DB マネージャで管理する．

具体的に，低レベルスケジューリングのためのサブフレームワークは以下のモジュールから構成される．

- 計算リソースのための負荷モニタ
- ネットワークのスループットモニタ
- リソース情報 DB マネージャ
- リソース状況プレディクタ
- リソーススケジューラ

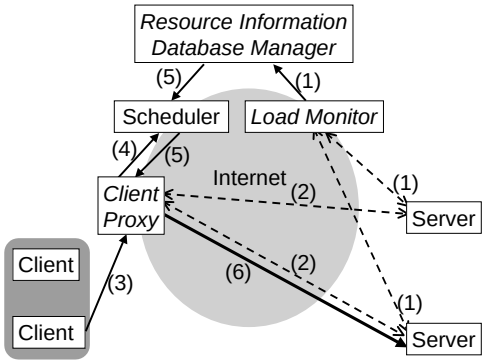

図 2 Ninf メタサーバアーキテクチャ

スループットモニタでは，あるクライアントと計算ノード間のスループットが各クライアントによって計測される．これは，現状の WAN 環境下では，ノード間のスループットを他のスループット情報から統計的に調べるのは困難なためである．一方，負荷モニタは 1 つのノードで複数計算サーバの情報を収集することができる．

モニタは定期的にリソース情報 DB マネージャにリソース情報を報告する． DB に格納された情報に基づき，リソース状況プレディクタはそのリソースの現在と未来の状況を予測する．ここで，予測はグローバルコンピューティングの任意の性能モデルに基づく．また，スケジューラは DB とプレディクタの情報を踏まえて，各 Network Task に対して適切な計算ノードを割り当てる．

2.3 Ninf メタサーバアーキテクチャ

Ninf システムは， RPC ベースのグローバルコンピューティングシステムである． Ninf における Network Task スケジューリングを行うフレームワークとして， **2.2**に基づくメタサーバスケジューリングフレームワークのプロトタイプを設計・実装した．

メタサーバは以下のモジュールによって構成される．

- 計算リソースの負荷モニタ
- クライアントプロキシ
- リソース情報 DB マネージャ
- リソーススケジューラ

ここで，クライアントプロキシには 2 つの機能がある．主な機能として，ネットワークのモニタリングがある．クライアントプロキシとサーバ間のスループットを調べることで，サイト内の複数ユーザに対するモニタリングコストが削減することができる．また，ネットワーク保護のための firewall としても機能する．

具体的にメタサーバは次のように処理を行う (**図 2**)．

(1) ロードモニタは計算サーバの負荷情報をモニタし，それをリソース情報 DB に格納する．

(2) クライアントプロキシはクライアントプロキシと各サーバ間のネットワークの混雑度等の通信情報をモニタし，その結果をクライアントプロキシで管理する．

(3) 計算要求 (Request) が発生すると，クライアントはクライアントプロキシに Network Task のスケジューリング要求を送信する．

(4) クライアントプロキシはそのサイト内から発生したスケジューリング要求をスケジューラに発行する．その際，クライアントプロキシとサーバ間の通信情報を通知する．

(5) スケジューラはリソース情報 DB から計算サーバの負荷情報を得ると， Network Task の特性に依存したタスクに対して最適なサーバを決定し，クライアントに紹介する．新たな Network Task がスケジュールされたため，スケジューラはリソース情報 DB のサーバの負荷情報を更新する．

(6) クライアントプロキシは，クライアントの Request を紹介されたサーバに転送する．

これらのモジュールは Java で実装されており，コアスケジューリングモジュールのスケジューリングポリシはネットワーク経由で動的に変更可能となっている．

3. スケジューリング手法の予備的評価

グローバルコンピューティングが想定している異機種計算機・ネットワークの性能とその負荷が変動するような環境下で，メタサーバスケジューリングフレームワークが有効に機能することが求められる．そこでその有効性を調査するため，以下の基本的なスケジューリング手法の予備的性能評価を 1) 地理的に離れた 2 つのサイト (電総研，東工大) を用いた実環境と， 2) 性能評価モデル[12] で行った．

Round Robin : RR 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．

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，

$$(L+1)/P$$

が最小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，リソース情報 DB で管理されている情報をもとに，サーバが決定される．評価実験では，60[sec] ごとにサーバ，ネットワークの情報をモニタして， DB 情報の更新を行うが，複数のクライアントが同時にアクセスするような環境では，クライアントにサーバが割り当てられるたびに， DB で格納されている情報と実際のリソースの状況が大幅に変わってしまう．そこで， LOAD と LOTH では， L, T に対して，割り当て

表 1　実測によるスケジューリング手法の評価実験環境

	計算機 [性能]	サイト
Client1-4	A:UltraSPARC[143MHz, 256MB]	東工大
MetaServer ClientProxy	B:UltraSPARC [200MHz×2, 256MB]	東工大
Server1	C:J90[200MFlops×4, 512MB]	電総研
Server2	D:UltraSPARC[143MHz, 128MB]	東工大

表 2　1 回の Request の平均処理時間 (単位はすべて [sec])

Server	通信時間	計算時間	所要時間
J90	17.832	0.345	19.265
UltraSPARC	2.941	3.687	6.972

時の負荷を予測しないもの (LOAD, LOTH) と予測するもの (LOAD+L, LOTH+L, LOTH+T, LOTH+LT) を用意した．負荷の予測は単純なものを採用し，計算サーバ負荷では，Request が割り当てられた時にそのサーバの負荷に 1 を加算した値を予測値とした．また，通信スループットは，[計測された通信スループット] / [ネットワークを共有しているジョブ数] を予測値とした．

3.1　実環境での評価

3.1.1　実環境での評価実験環境

表 1 に，評価実験環境を示す．LAN 内でのマシン間のバンド幅は A-B 間は 100[Mbps]．B-D 間は 10[Mbps] となっている．B-C 間は広域ネットワークである．

サーバの計算ルーチンには Linpack を採用した．Linpack では，LU 分解，後退代入を計算サーバ上で実行する．この際，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．Linpack の求解ルーチンには，Server1(J90) では libSci ライブラリの sgetrf, sgetrs を用いて 4PE で並列実行するものとした．Server2(UltraSPARC) ではブロック化等の高速化が施された glub4, gslv4 を用いた．

以上の環境において，問題サイズ 600 の Linpack の Request を 1 回発行した時の平均処理時間は**表 2**に示すとおりである．

測定では，Request が頻繁に発行される場合 (case A) と，適度に発行される場合 (case B) を想定し，各クライアントでの Request の発行間隔はそれぞれ平均値

$$I = [\text{Request}の所要時間] + interval \quad (1)$$

$$I = [\text{Request}の所要時間] \times 2 + interval \quad (2)$$

の指数分布とした．ここで Request の所要時間は**表 2**の遅いものを適用し，$interval = 5$[sec] とした．

3.2　実環境での評価実験結果

表 3に実環境での評価実験結果を示す．表の上が case A，下が case B の結果であり，それぞれ平均所要時間，計算サーバの稼働率，測定時の通信スループットを示している．

case A では，RR が非常に良い性能を示している．これは LOTH では比較的同じサーバが連続して選択されているために，全体の所要時間が長くなったと考えられる．このことから，単位時間あたりの Request 数が非常に多い場合には，Request の通信，計算の干渉が小さくなるようなスケジューリングが全体のスループットの向上するよう適切に機能することが分かる．

case B では，LOTH が最も応答時間が短かった．これは Request 発行間隔が比較的長いため，サーバ割り当て時のリソースの負荷状況を見て判断したものが非常に良い性能を示すことが分かる．LOTH+L, LOTH+T がわずかに性能が落ちているのは，今回の資源予測が非常に単純なものであったこと，各リソースの負荷をバランス良く考慮して判断を行わなかったことによるものと考えられる．また，case B においても RR がよい性能を示しており，これもリソースの競合を避けるようなスケジューリングが行われたためである．

一方，case A, case B とも LOAD の応答時間が非常に遅くなっていた．これは，計算サーバの性能，負荷値のみでスケジューリングした結果，すべての Request に対して J90 が選定されたためである．

3.3　性能評価モデルによるスケジューリング手法の評価

前節の評価実験では，計算リソースの競合が少なくなることに重点をおいたスケジューリングアルゴリズムが，より適切な資源情報予測を行うことが分かった．しかしながら，グローバルコンピューティングシステム上での実環境における評価実験では測定環境が限定されるため，これらのアルゴリズムについて論じるための部分的な解決方法となり得るが，一般的な性能について論じるには不十分である．

この章では，様々なグローバルコンピューティングシステムやその振舞いが表現可能である性能評価モデル[12]を用い，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手法について検討する．

3.3.1　性能評価モデルの概要

性能評価モデルでは，グローバルコンピューティングシステムはスケジューリングユニット (負荷モニタ，リソーススケジューラ，リソース情報 DB)，クライアント，サーバ，ネットワークで構成される．このうちサーバ，ネットワークは，待ち行列を用いて表現する．待ち行列を用いてグローバルコンピューティングシステムから発生するデータやジョブと，他のシステムから生成されるデータやジョブをシミュレートすることにより，様々なグローバルコンピューティングシステムの構成やその振舞いを表現することができる．

3.3.2　性能評価モデルによる評価環境

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 3**に示す．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 × クライアント 4 の構成となっている．$ServerA$ 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 (0.20[MB/s])，$ServerB$ は計算性能は高くない (40[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 (1.08[MB/s]) ものとする．これは実測とほぼ同様の設定であるが，クライアントは複数のサイトに分散して

表 3　実環境での Linpack を用いたスケジューリングの測定結果 (上:case A, 下:case B)

スケジューラ	資源予測		平均計算時間	平均通信時間	平均所要時間	サーバの稼働率		測定時通信スループット	
	L	T	[sec]	[sec]	[sec]	A(400Mflops)	B(40Mflops)	A [MB/s]	B [MB/s]
RR	–	–	2.879	14.821	18.269	0.151	0.353	0.169	0.687
LOAD	–	–	0.421	63.051	65.879	0.215	0.000	0.081	1.067
LOAD+L	○	–	0.409	65.956	67.935	0.199	0.000	0.035	1.074
LOTH	–	–	15.599	16.434	32.849	0.084	0.451	0.107	0.560
LOTH+L	○	–	13.396	21.050	35.264	0.129	0.449	0.121	0.680
LOTH+T	–	○	19.831	8.459	28.762	0.039	0.857	0.087	0.511
LOTH+LT	○	○	7.562	21.992	30.588	0.093	0.569	0.125	0.764
RR	–	–	2.303	10.725	13.606	0.094	0.121	0.138	0.898
LOAD	–	–	0.405	53.512	55.325	0.178	0.000	0.076	1.080
LOAD+L	○	–	0.392	54.299	57.115	0.169	0.000	0.052	1.073
LOTH	–	–	6.941	5.263	12.590	0.068	0.651	0.128	0.674
LOTH+L	○	–	4.822	12.964	18.407	0.154	0.260	0.135	0.744
LOTH+T	–	○	7.370	8.485	16.407	0.035	0.456	0.208	1.031
LOTH+LT	○	○	3.641	9.613	13.680	0.161	0.247	0.158	0.760

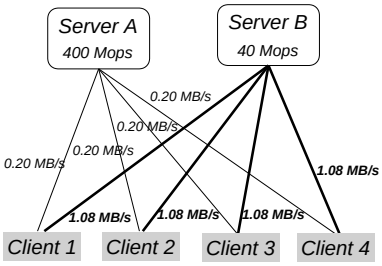

図 3　性能評価モデルによる評価環境

いることを想定して，各クライアント，サーバ間のネットワークが独立している点が実測とは異なっている．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．計算ルーチンには実測同様， Linpack を用いた．

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 $n = 600$.

Request の発行間隔は**式 1, 2**で決定．

ネットワーク　バンド幅 $T_{net} = 1.5$[MB/s].

図 3のスループットをシミュレート．

サーバ　**図 3**の性能で FCFS で処理．稼働率 0.10.

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 50 回繰り返した． 50 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 90[%] において信頼区間 ±10[%] 未満であった[13]．

3.3.3　シミュレーションによるスケジューリングの評価結果

表 4に性能評価モデルを用いたスケジューリングの測定結果を示す． **3.2**同様，上は各クライアントの発行間隔が狭いもの (case A)，下は広いもの (case B) となっている．

case A, case B とも LOTH が適切にスケジュールされていたことが分かる．これは実環境の評価と異なり，ネットワークを独立に設定して通信の競合が起こらなかったために，各リソース情報を見て判断したものが良い結果が得られたと考えられる．一方で， LOTH+T, LOTH+LT が LOTH と比較するとやや応答時間が遅くなっていたのは， Request の所要時間が短かったためにシミュレーションにおける通信スループットの性能予測が正確に行われなかったことが顕在化したと思われる．

また， RR が悪い性能を示したのは，シミュレーション環境ではネットワークの干渉がなかったため，リソースの状態を考慮してスケジューリングを行なったものより遅くなっている．

4.　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におい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を統合するための基本的なスケジューリングフレームワークについて述べるとともに，そのための Ninf システムにおけるメタサーバスケジューリングフレームワークの実装とそれを用いたスケジューリング手法の評価について述べた．

実環境および性能評価モデルによる評価実験より，グローバルコンピューティングシステムのリソースが競合するような環境では，特に Request が頻繁に発生するような場合に RR のように各 Request の競合が小さくなるようなスケジューリングが非常によく機能することが分かった．一方，ネットワークリソースが干渉しないような場合，すなわち複数サイトにクライアントが分散しているような場合や Request 間隔が比較的長い状況では， LOTH のようにリソース情報を考慮してスケジューリングを行うことが重要であることが示された．また，スケジューリング時のリソース情報を正確に予測できない場合には，リソース情報を考慮しても適切なスケジューリングが行えないことが示唆された．

また，以下のグローバルスケジューリングにおける課

表 4　性能評価モデルでの Linpack を用いたスケジューリングの測定結果 (上:case A, 下:case B)

スケジューラ	計算資源予測		平均計算時間	平均通信時間	平均所要時間	サーバの稼働率	
	L	T	[sec]	[sec]	[sec]	A(400Mflops)	B(40Mflops)
RR	–	–	2.314	14.339	16.654	0.538	0.130
LOAD	–	–	1.689	9.375	11.064	0.387	0.148
LOAD+L	○	–	0.711	5.499	6.210	0.309	0.155
LOTH	–	–	0.377	3.752	4.129	0.248	0.162
LOTH+L	○	–	0.423	4.010	4.433	0.256	0.161
LOTH+T	–	○	1.002	6.981	7.983	0.346	0.151
LOTH+LT	○	○	0.908	6.783	7.691	0.339	0.152
RR	–	–	2.145	11.527	13.672	0.418	0.117
LOAD	–	–	0.964	5.726	6.690	0.293	0.130
LOAD+L	○	–	0.800	5.423	6.223	0.291	0.130
LOTH	–	–	0.374	3.554	3.928	0.247	0.135
LOTH+L	○	–	0.379	3.575	3.954	0.248	0.135
LOTH+T	–	○	0.612	4.701	5.313	0.272	0.132
LOTH+LT	○	○	0.636	4.816	5.452	0.275	0.132

題が明らかになった.

計算サーバの性能の指標　今回の測定ではアプリケーションとして固定サイズの Linpack を利用したが, サーバ計算機の性能は, アプリケーションの種類, 問題サイズ等により大きく異なる. スケジューラに適切な情報を提供するためには, 問題の性質と計算サーバの性能に対応可能な指標が必要である[14].

混雑度とスケジューリング手法　Request の発生度合いによって, スケジューリングのパラメータの重みが異なってくることが明らかになった. グローバルコンピューティングシステム全体のスループットを向上させるスケジューリングを行うためには, これらを適切に判断する必要がある.

今後は性能評価モデルで様々な環境を想定した評価を行い, これらの課題に応える低レベルスケジューリングを実現したのち, Network Task を適切にリソースの割り当てるジョブ / アプリケーションリソースアロケーションのサブフレームワークを提案する予定である.

謝辞　日頃よりご討論頂く電子技術総合研究所 関口智嗣氏, 高木浩光氏, 新情報開発機構 佐藤三久氏に感謝致します.

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, *Proceedings of Supercomputing '96* (1996).
3) Grimshaw, A. S. et al.: Legion: The Next Logincal Step Toward a Natiowide Virtual Computer, CS 94-21, University of Virginia (1994).
4) Arbenz, P., Gander, W. and Oettli, M.: The Remote Computation System, Technical Report 245, ETH (1996).
5) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications* (1997).
6) Berman, F. et al.: Application-Level Scheduling on Distributed Heterogeneous Networks, *Proceedings of Supercomputing '97* (1996).
7) Weissman, J. B. and Zhao, X.: Scheduling Parallel Applications in Distributed Networks, *Journal of Cluster Computing (accepted)*.
8) Feitelson, D. and Rudolph, L.: *Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science, Springer.
9) Raman, R., Livny, M. and Solomon, M.: Matchmaking: Distributed Resource Management for High Throughput Computing, *Proceedings of HPDC-7* (1998).
10) Abramson, D. et al.: The Nimrod Computational Workbench: A Case Study in Desktop Metacomputing, *Proceedings of ACSC97* (1997).
11) Wolski, R., Spring, N. and Peterson, C.: Implementing a Performance Forecasting System for Metacomputing: The Network Weather service, *Proceedings of Supercomputing '97* (1997).
12) 竹房, 合田, 小川, 中田, 松岡, 佐藤, 関口, 長嶋: 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 –, 並列処理シンポジウム JSPP'98 論文集, pp. 127–134 (1998).
13) Jain, R.: *The art of computer systems performance analysis*, John Wiley & Sons, Inc. (1991).
14) 関口, 佐藤, 井須, 長嶋: 定量的な並列システムのスケーラビリティ評価指標, 並列処理シンポジウム JSPP'96 論文集, pp. 235–242 (1996).