Vol. 39 No. 6 情報処理学会論文誌 June 1998

複数クライアントによるLAN/WANでのNinfの性能

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

広域ネットワークの整備につれ，高性能広域分散計算を実現する試みが我々のNinfを含めていくつか行われている．しかしこのような広域計算システムの，特にWANにおいて複数のクライアントが複数のサイトに分散している状況下での性能特性に関する議論は十分になされていない．本稿では，Ninfおよび類似のシステムの実現可能性を調査するため，LAN/WAN環境でLinpack/EPベンチマークを実施し，次のような結果を得た．1）十分なバンド幅があれば，Ninfを用いた方がLocal実行するより高速になる．2）既存の高性能計算機は性能や耐久性の点で広域計算システムの運用に十分なプラットフォームである．3）ベクトル並列計算機（Cray J90）では，高性能並列ライブラリが有効利用できる，すなわち既存の高性能ライブラリの再利用性がある．4）計算主体の計算（EP）では現状の広域計算システムで十分に運用できる．5）通信主体の計算（Linpack）では，LAN環境ではサーバの稼働率が性能を支配し，WAN環境では通信性能と設置条件によって性能に与える影響に一定の傾向がある．

Multi-client LAN/WAN Performance Analysis of Ninf

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

Rapid increase in speed and availability of network of supercomputers is making high-performance global computing possible, including our Ninf system. However, critical issues regarding system performance characteristics in global computing have been little investigated, especially under multi-client, multi-site WAN settings. In order to investigate the feasibility of Ninf and similar systems, we conducted benchmarks under various LAN and WAN environments, and observed the following results: 1) Given sufficient communication bandwidth, Ninf performance quickly overtakes client local performance, 2) current supercomputers are sufficient platforms for supporting Ninf and similar systems in terms of performance and OS fault resiliency, 3) for a vector-parallel machine (Cray J90), employing optimized data-parallel library is a better choice compared to conventional task-parallel execution employed for non-numerical data servers, 4) computationally intensive tasks such as EP can readily be supported under the current Ninf infrastructure, and 5) for communication-intensive applications such as Linpack, server CPU utilization dominates LAN performance, while communication bandwidth dominates WAN performance, and furthermore, aggregate bandwidth could be sustained for multiple clients located at different Internet sites; as a result, distribution of multiple tasks to computing servers on different networks would be essential for achieving higher client-observed performance.

1. はじめに

ネットワーク技術の発展により，広域に分散した計算資源や情報資源を積極的に活用して科学技術計算の分野での大規模計算の要求に応えることが可能となる．近年これを目的とした高性能広域計算システムがNinf[1),2)]を筆頭に複数提案されている[3)~7)]．Ninfシステムは Fortran，C，Java 等の言語から比較的粗粒度の Remote Procedure Call（RPC）により，ネットワーク透過に複数のリモート計算サーバやデータベース

†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

サーバおよびストレージ資源を利用したアプリケーションを作成可能とする．Ninf の用意するメタサーバにより，この RPC を単位としたマクロデータフローによる並列実行や信頼性の保証を行う[8]．

一方，高性能広域計算を実現するためのシステムの性能特性や指標に関する議論は十分になされていないため，広域計算の構成技術に関する様々な特性を調査する必要がある．

複数クライアントによる共有　計算サーバはネットワークを通じて任意の複数クライアントに共有される．さらに，広域計算におけるトランザクションはデータベースの query とは異なり長くなりがちである．したがって，広域計算システムと現状の超高性能計算機 OS の組合せで集中した計算要求をロバストに処理することが求められる．

リモートライブラリの設計と再利用性　超高性能並列機で Ninf サーバを運用する際，順次到着する計算要求に対して計算資源を分割して複数タスクを**並列に処理するか**，全資源を 1 つのタスクに割り当てて**直列に処理するか**という処理指針がある．この選択は提供するライブラリの設計段階と再利用性の有無にかかわる．

通信性能　通信バンド幅の増大は見込めるが，それが十分か判断するための定量的尺度は提供されていない．また，レイテンシの短縮は難しいため，大きいレイテンシが性能向上の妨げとならないかどうか明らかでない．

計算サーバの選択と計算性能　超高性能計算機サーバは複数のリモートクライアントに共有された状況で高性能ライブラリを実行する．計算サーバの選択にはどのような指標を用いるべきか，また現状のスーパコンピュータや MPP がそのような利用形態に適しているかが明らかでない．

本稿では，Ninf システムの概要について述べるとともに，これらの高性能広域計算の技術的問題を明らかにするため，Ninf システムを用いた性能評価実験結果を示す．評価では LAN/WAN 環境で複数のクライアントを設定した Linpack/EP ベンチマークを実施し，以下の結果を得た:

(1) ローカル実行より Ninf を用いた方が高速であることから，高性能広域計算の有効性を実証した．

(2) サーバにベクトル並列計算機 Cray J90 4PE や Sun SMP WS を用い，複数クライアントが並行にアクセスする環境での実験結果から，並列計算機用 OS と Ninf システムの組合せで堅牢なネットワークサービスが提供できることを確認した．

(3) 並列計算機（Cray J90）ではピーク性能を実現するように最適化されたライブラリで Ninf サービスを提供することで，LAN/WAN の両環境において十分効率良く運用できることが判明した．SMP では，複数のクライアントから要求がある状況では高スレッド化による性能低下が見られ，クライアント数（サーバの稼働率）に対して適切なスレッド数で計算する必要があることが分かった．

(4) 計算主体の計算（EP）においては LAN/WAN 環境の差なく，計算サーバの稼働率が向上した．このような計算（並列レンダリング / イメージング，パラメータ感度解析）では現状の高性能広域計算システムで十分に運用できることを確認した．

(5) 通信主体の計算（Linpack）では，LAN 環境ではサーバの稼働率が性能を支配することが分かった．一方，WAN 環境では通信スループットと設置条件に大きく依存し，サーバの負荷や稼働率による影響はほとんど見られなかった．

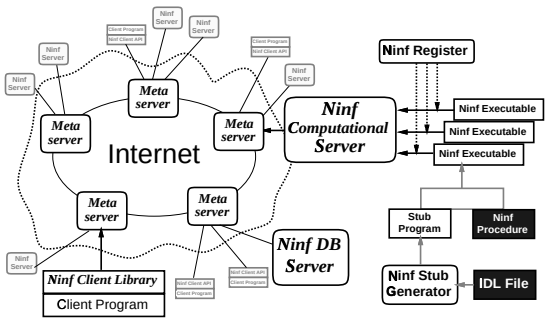


図 1　Ninf システムアーキテクチャ
Fig. 1　Overview of Ninf system.

2. Ninf システムの概要

図 1 に Ninf システムアーキテクチャを示す．Ninf システムは，Ninf のサービスを提供する **Ninf 計算サーバ**，Ninf クライアントの API を提供する **Ninf クライアントライブラリ**，広域ネットワークに分散した Ninf サービスのスケジューリングを支援する**メタサーバ**等から構成される．これらの構成要素間の通信は Ninf RPC（Remote Procedure Call）[9] と称する手続きにより実現される．Ninf Stub Generator，Ninf Register はサービスの提供を半自動化するツールであり，サーバの保守を容易にする．

Ninf 計算サーバ　Ninf サーバは計算ライブラリ等の資源を提供するホスト上のデーモンプロセスで，クライアントとの通信やサービスの開始・終了を管理する．提供されるサービスは Ninf Executable と称する実行ファイル形式をとり，サーバはこれらを適宜

fork&execする．Ninf Executableは提供するライブラリと通信ランタイムを含んだstubルーチンとリンクして，半自動的に生成される．

Ninfクライアントライブラリ Ninfのクライアントは，C，C++，Fortran，Java等の言語と各言語用の簡易なNinfクライアントライブラリを用いて作成される．Ninf_callはこのライブラリのAPIの一つでNinfサーバに計算要求を行う．行列積ルーチンを呼び出す例を次に示す．

```
double A[n][n], B[n][n], C[n][n];
Ninf_call("dmmul", n, A, B, C);
```

このようにローカルライブラリを呼び出す場合と類似した手続きでNinfリモートライブラリが利用できる．Ninf_callはNinf RPCを使って自動的に関数の引数の数・型を決定し，Ninf Executableと相互に通信して引数や計算結果をやりとりする．したがって，ユーザには自分のマシン上でリモートライブラリが実行されているように見える．他には非同期呼び出しNinf_call_async，並行に実行するトランザクション区間の指定Ninf_transaction_begin, Ninf_transaction_end等のAPIが用意されている．

メタサーバ メタサーバは複数のNinfサーバ情報をモニタし，クライアントからのリクエストのスケジューリング，動的負荷分散を行う．メタサーバを介することでユーザは必要なサービスを提供しているNinfサーバの所在を意識せず利用できる．また，メタサーバはトランザクションによる並行実行を支援する．トランザクション区間に記述されたNinf_callの列に対して引数のデータ依存グラフを生成し，必要な同期をとりながら，独立なNinf_callを並行に実行する．

3. 単一クライアントによるサーバの性能

まず，サーバに1つのクライアントがNinf_callするという理想的な利用条件でLinpackベンチマークを用いた評価実験を複数プラットフォーム上で行った．Linpackでは密行列の転送が必要とされるが，行列サイズが大きい場合には計算時間が主体となる．

3.1 Linpack ベンチマーク

Linpack ベンチマーク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．Ninfを用いたベンチマークでは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$8n^2 + 20n + O(1)$ bytesである．よってNinf_callの所要時間をT_{Ninf_call}，Linpackの問題サイズをnとすると，Ninf_callの実効性能は$P_{Ninf_call} = (2/3n^3 + 2n^2)/T_{Ninf_call}$のように表せる．

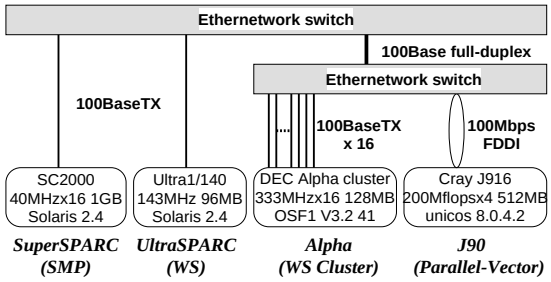


図2 クライアント及びサーバに用いた計算機
Fig. 2 Client and server machines in the LAN benchmark.

表1 評価に用いたサーバとクライアントの組合せ
Table 1 Combination of clients and servers in the LAN benchmark.

	Local	Remote(Ninf_call)		
Client		Ultra	Alpha	J90
SuperSPARC	○	○	○	○
UltraSPARC	○	–	○	○
Alpha	○	–	–	○

3.2 単一クライアントでの評価環境

クライアントおよびサーバに用いた計算機とそのOSは**図2**に示すとおりである．また，計測を行ったクライアントとサーバの組合せは**表1**に示す．Localは Ninfを用いずにクライアント上で実行したものである．

Linpackの求解ルーチンとして，J90ではlibSciライブラリのsgetrf，sgetrsを利用し，4PEを占有して実行するものとした．その他の計算機においてはブロック化等の高速化が施してあり，キャッシュに依存するWSでも高速に実行できるglub4，gslv4[10]ルーチンを用いた．これはNinfにとっては不利な条件となる．

問題サイズnは100~1600を対象とした．

3.3 単一クライアントによるLinpackの測定結果

図3にSuperSPARC，UltraSPARCをクライアントとした測定結果を示す．横軸はLinpackの問題サイズnを示し，縦軸はNinf_callとLocalの性能をMflopsで示す．

SuperSPARCとUltraSPARCのLocalはnによらずほぼ一定の性能を得た．一方，Ninf_callではnが大きくなるにつれ性能が向上し，SuperSPARC，UltraSPARCとも$n = 200 \sim 400$ですでに各マシンのLocalよりも高い性能が得られた．各クライアントからJ90へのNinf_callでは，J90のLocalが$n = 1600$のとき600Mflopsに達しており，Ninf_callの性能は飽和していなかった．また，クライアントの性能が異なる場

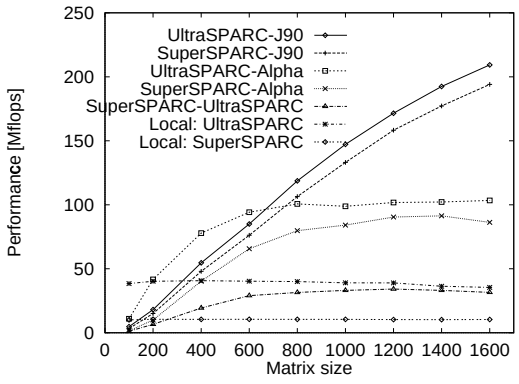


図 3 SPARC をクライアントとした Linpack の実行結果
Fig. 3 Ninf LAN Linpack results with single SPARC clients.

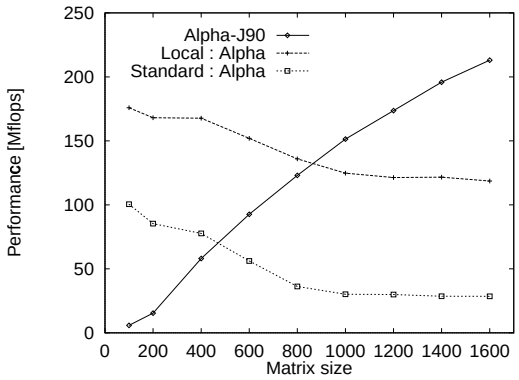


図 4 Alpha をクライアントとした Linpack の実行結果
Fig. 4 Ninf LAN Linpack results for single Alpha client.

合でも Ninf_call ではほぼ同程度の性能向上を示した．

図 4 に Alpha をクライアントとした測定結果を示す．ここで，ユーザが複雑な問題で最適化を容易に施せない場合を想定して，ブロック化されていない Linpack の標準ルーチン（dgefa，dgesl）を用いた性能も示す．

Alpha の Local と Ninf_call を比較すると，$n = 800 \sim 1000$ で Ninf_call の方が高性能であった．一方，クライアントで最適化していない標準ルーチンを用いた場合は，$n = 400 \sim 600$ の近辺で Ninf_call が Local の性能を上回った．

PC や WS の性能向上は著しいが，複雑で大規模な計算に対してユーザが短時間で性能を十分に引き出すプログラムを書くことは容易ではないことが多い．ネットワーク上の多数の Ninf サーバに数多くの最適化されたルーチンが登録されている環境が実現されれば，それを個々のユーザが用いることで，高性能な大規模計算に対する過度なプログラミングの負担が軽減される．

表 2 に Ninf_call 通信スループットの平均値とクライアント - サーバ間の FTP 通信スループットの実測値を示す．各クライアントと J90 間の Ninf_call のスループットは，ベースラインの転送速度が遅いため，Ninf_call では高い通信スループットが得られていない．また，UltraSPARC，Alpha をサーバとした場合も FTP と同程度のスループットが得られており，Ninf では異機種間においてもデータ変換等のオーバヘッドによる著しい性能低下が起こらないことを示している．これは，J90 との通信ではベンダで提供された非常に高速なデータ変換ルーチン（ieg2cray，cray2ieg）を使用しているためであり，Sun WS と Alpha 間ではデータ変換のコストが通信のコストに比べ十分に小さいことによるものと考えられる．

表 2 クライアント – サーバ間の Ninf_call/FTP 通信スループット
Table 2 Client-server Ninf_call/FTP communication throughput.

Client	*Server* (Throughput [MB/s]) Ninf_call/ FTP		
	UltraSPARC	Alpha	J90
SuperSPARC	3.3/4.0	4.0/4.0	2.2/2.8
UltraSPARC	–	6.6/7.4	2.4/2.7
Alpha	–	–	2.5/2.9

4. 複数クライアントによるサーバ性能

高性能広域計算においては，Ninf サーバは多数のクライアントから同時に Ninf_call されることが前提となる．高性能広域計算システムが有効に機能するには，このような場合にも平均処理時間の著しい増大を招かず，かつサーバマシンの稼働率を高く維持することが必要である．また，現状の広域ネットワークにおいて高性能広域計算の実用性や技術的問題となりうる点の確認は重要である．そこで，LAN/WAN 環境において複数クライアントを用いたサーバの性能評価を行った．評価には Linpack/EP ベンチマークを用いた．Linpack は前述したように通信のオーバヘッドが顕在化しやすいという性質を持つ．一方，EP は通信量一定で計算量を任意に設定できるため，通信の影響はほとんど現れない．

4.1 複数クライアントでの評価環境

Ninf クライアントプログラムのモデルとして，Linpack（sgetrf，sgetrs），および EP のルーチンを繰り返し呼び出すものを用意した．このモデルでは Ninf_call はステップ（s sec）ごとに一定の確率 p で発生するものとし，クライアント数は c，問題サイズ n は試行の間一定とする．測定は $s = 3$，$p = 1/2$，$c = 1, 2, 4, 8, 16$ という条件で実施した．

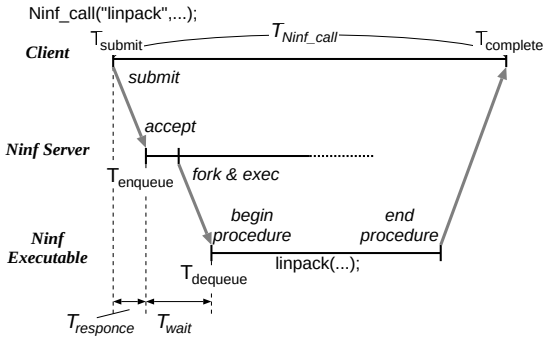


図 5 Ninf_call の測定のタイミング
Fig. 5 Multi-client Ninf_call measurements.

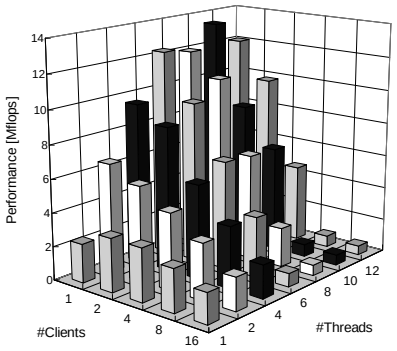


図 6 SMP の複数クライアントでのマルチスレッド版 Linpack の Ninf_call 平均実効性能
Fig. 6 Average performance of SMP multi-client LAN Ninf_call of multi-thread Linpack.

実験ではサーバの稼働率，負荷☆，および各 Ninf_call について，**図 5** のように測定し，通信スループット，性能を算出した．ここで $T_{responce}$， T_{wait} は Ninf_call の応答時間および待ち時間である．

4.2 Linpack ベンチマークによる評価

1で述べたとおり，複数クライアントの次々到着するサービス要求に対し，1)（並列性の低い）複数タスクで並行処理を行うか，2)（並列性の高い）タスクで逐次に処理を行うか，という選択がある．この場合，クライアント数が少なければ後者の方が高速であるが，クライアント数が増加し，並列化ライブラリのプロセス切替え等によるオーバヘッドが大きい場合は，前者の方が高速になると予想される．この評価を行うため，Linpack を 4PE を占有して高速実行するが同時に 1 タスクしか実行できない 4PE 版と，1PE で実行するが 4 タスク並行に処理可能な 1PE 版の両者を用い，その比較を行った．クライアントの実効性能は前節と同様にして算出し，問題サイズ n は 600，1000，1400 とした．

4.2.1 J90 を用いた LAN での測定結果

評価では J90 をサーバとし，Alpha WS クラスタをクライアントとした（図 2）．

表 3，**4** に 1PE 版，4PE 版を用いたときの性能，通信スループットと試行中のサーバの稼働率，負荷の平均値を示す．クライアント数 c が増加する場合，サーバの負荷が増加し，また通信スループットが低下するため，1PE 版，4PE 版ともに Ninf_call の性能が低下する．4PE 版において性能のばらつきが見られるのは，Linpack の実行の際に全 PE が占有されることによる．

問題サイズ n が増加する場合には，CPU 稼働率，負荷平均値が高くなるが，4PE 版で $n = 1400$，負荷平均値最大 30 に達しても稼働性は保たれていた．また，n の増加にかかわらず実効性能も維持されている．

1PE 版と比較すると，4PE 版は負荷，CPU 稼働率ともに高くなっている．一方で，各クライアントでの平均実効性能は c が少ない場合には 4PE 版が有利であるのに対し，c が多い場合でも 1PE 版がわずかに良い性能にとどまった．これは 4PE 版は計算コアの処理速度が高速なため，一時的に 4PE が占有されても Ninf_call の平均実行時間が低下しないこと，4PE が占有されている場合でも通信は同時に行われていることによるものと考えられる．また，Ninf_call の応答時間および待ち時間も Ninf_call のタイミングにより長くなることもあるが，平均値では n，c によらず 1PE 版で 0.40[sec]，4PE 版で 0.10[sec] 程度にとどまった．したがって，Cray J90 上で最高性能を実現する最適化されたライブラリを，Ninf サーバを用いて効率良く運用できることを確認した．

4.2.2 SMP を用いた LAN での測定結果

サーバとして SuperSPARC の 16 台構成 SMP（図 2）を用いた場合の結果を**表 5** に示す．

1PE 版では，c の増加にともなう性能の低下が J90 に比べて小さく，応答時間および待ち時間も増大しなかった．また，$c = 16$ でも稼働率に余裕を残していた．SPARC SMP は J90 と比較して計算性能が低いため，I/O ネックが顕在化しなかったためと考えられる．

一方，マルチスレッド版の Linpack ルーチン☆☆による評価結果（Solaris 2.5.1 + SuperSPARC SMP）を**図 6** に示す．XYZ 軸にはそれぞれクライアント数 c，Linpack ルーチンの使用するスレッド数 t，Ninf_call 性

☆ サーバで処理しているプロセスと同一サーバ上にある待ち状態の実行可能プロセスの総数

☆☆ 電子技術総合研究所の建部修見氏に提供していただいた．

表 3 1PE 版の LAN での複数クライアントによる Linpack 実行結果
Table 3 Performance results of 1-PE multi-client LAN Linpack.

n	c	性能 [Mflops] max/min/mean	スループット [MB/s] max/min/mean	サーバ稼働率	サーバ負荷
600	1	72.71/69.90/71.16	2.57/2.42/2.48	12.63	0.68
	2	72.01/64.33/69.63	2.52/2.10/2.41	20.86	1.20
	4	72.40/43.85/67.05	2.55/1.89/2.34	42.03	1.99
	8	72.04/17.44/49.02	2.55/0.60/1.87	82.20	4.90
	16	64.13/8.12/21.27	2.25/0.24/0.86	98.66	13.21
1000	1	95.06/93.13/93.40	2.58/2.49/2.53	21.40	1.06
	2	96.09/54.10/89.90	2.51/2.25/2.42	37.84	1.62
	4	93.64/59.92/81.39	2.58/1.47/2.11	76.02	3.58
	8	83.91/30.79/46.48	2.39/0.65/1.35	99.38	7.72
	16	33.24/15.26/21.14	1.12/0.31/0.57	100.00	16.01
1400	1	115.01/112.26/113.65	2.58/2.51/2.54	24.27	1.19
	2	113.58/85.15/110.48	2.54/2.36/2.44	46.95	2.19
	4	109.54/70.89/93.35	2.44/1.60/2.19	88.40	4.14
	8	64.74/42.17/50.11	1.72/0.81/1.21	99.97	8.53
	16	27.87/19.41/23.93	0.74/0.38/0.51	100.00	16.64

表 4 4PE 版の複数クライアントによる Linpack 実行結果
Table 4 Performance results of 4-PE multi-client LAN Linpack.

n	c	性能 [Mflops] max/min/mean	スループット [MB/s] max/min/mean	サーバ稼働率	サーバ負荷
600	1	94.05/81.76/91.46	2.55/2.40/2.47	14.89	0.87
	2	91.05/21.62/83.17	2.46/2.04/2.36	27.56	1.42
	4	92.58/21.70/75.83	2.52/1.17/2.12	53.56	4.06
	8	89.35/24.31/51.51	2.50/0.55/1.36	90.00	9.98
	16	56.73/8.87/18.69	1.83/0.19/0.46	99.85	19.96
1000	1	150.96/70.39/141.43	2.56/2.46/2.51	28.64	1.45
	2	148.92/62.15/127.63	2.52/1.87/2.28	55.56	2.89
	4	134.26/61.01/92.98	2.53/0.98/1.56	87.90	6.02
	8	67.14/27.76/45.85	1.21/0.42/0.69	99.61	11.01
	16	32.22/13.93/20.33	0.59/0.20/0.30	99.99	24.81
1400	1	196.08/174.28/193.03	2.54/2.47/2.51	40.87	1.86
	2	184.17/139.53/157.98	2.48/1.80/2.13	66.79	3.17
	4	119.26/74.88/96.26	2.16/0.92/1.26	96.80	7.53
	8	69.26/34.23/48.27	0.91/0.43/0.59	99.86	15.11
	16	35.27/17.80/23.25	0.48/0.21/0.28	100.00	30.29

表 5 LAN での SMP の複数クライアントによる Linpack 実行結果（タスク並列）
Table 5 SMP multi-client LAN Linpack results in a task parallel manner.

n	c	性能 [Mflops] max/min/mean	スループット [MB/s] max/min/mean	サーバ稼働率 / 負荷
600	4	4.37/2.98/3.80	1.36/1.07/1.18	49.92/6.08
	8	4.25/2.73/3.51	1.05/0.14/0.37	62.91/8.84
	16	3.77/2.03/2.81	1.40/0.10/0.34	89.89/15.37

能を示す．問題サイズ n は 600 とした．

$c \times t$ がプロセッサ数 16 より小さい場合は良い性能を示しているのに対し，高スレッド化したライブラリを用いた場合，c が大きくなるにつれ著しい性能低下が見られた．これは SPARC SMP 上の Solaris では，複数クライアントからの実行要求に対するハンドリングを最適化していること，またマルチスレッドの co-scheduling を行っていないことから，キャッシュ /TLB ミスを含むスレッド切替えのオーバヘッドが顕在化したためと考えられる．

4.2.3 WAN での単一サイト，複数クライアントによる測定結果

WAN での単一サイト，複数クライアントによる評価では，サーバには同じく電子技術総合研究所の J90 を用

表 6 WANでの単一サイト，1PE版の複数クライアントによる実行結果
Table 6 Single-site, multi-client 1-PE Linpack benckmark results for WAN.

n	c	性能 [Mflops] max/min/mean	スループット [MB/s] max/min/mean	サーバ稼働率	サーバ負荷
600	1	7.81/3.25/5.90	0.166/0.067/0.128	4.22	0.37
	2	5.93/3.63/4.69	0.122/0.074/0.096	8.03	0.49
	4	3.63/1.22/2.41	0.075/0.025/0.050	8.34	0.47
	8	1.47/0.59/1.14	0.030/0.012/0.023	7.80	0.46
	16	0.69/0.37/0.54	0.014/0.007/0.011	8.10	0.46
1000	1	12.05/4.41/9.28	0.164/0.054/0.123	5.26	0.39
	2	8.47/5.81/7.18	0.104/0.071/0.088	9.00	0.52
	4	4.38/1.84/3.66	0.055/0.022/0.045	8.83	0.52
	8	2.27/0.99/1.83	0.028/0.012/0.022	8.39	0.47
	16	1.18/0.54/0.90	0.014/0.006/0.011	8.22	0.47
1400	1	17.32/9.93/13.89	0.164/0.090/0.130	6.57	0.41
	2	11.22/7.91/9.29	0.099/0.069/0.082	12.75	0.66
	4	6.47/2.15/5.38	0.058/0.019/0.048	9.46	0.52
	8	3.03/1.31/2.50	0.027/0.011/0.022	8.89	0.50
	16	1.58/0.76/1.25	0.014/0.007/0.011	10.60	0.57

表 7 WANでの単一サイト，4PE版の複数クライアントによる実行結果
Table 7 Single-site, multi-client 4-PE Linpack benckmark results for WAN.

n	c	性能 [Mflops] max/min/mean	スループット [MB/s] max/min/mean	サーバ稼働率	サーバ負荷
600	1	8.00/7.24/7.68	0.166/0.153/0.161	6.85	0.40
	2	6.05/1.51/3.77	0.124/0.030/0.077	6.90	0.42
	4	3.57/0.92/2.46	0.073/0.019/0.051	8.94	0.50
	8	1.52/0.49/1.08	0.031/0.011/0.022	7.99	0.49
	16	0.90/0.29/0.54	0.018/0.006/0.011	8.17	0.49
1000	1	12.75/7.26/10.50	0.160/0.091/0.131	6.43	0.39
	2	8.22/5.89/7.15	0.101/0.072/0.088	9.23	0.52
	4	4.80/1.46/3.97	0.061/0.018/0.049	9.39	0.53
	8	2.42/0.97/1.82	0.029/0.012/0.022	8.71	0.53
	16	1.36/0.47/0.88	0.016/0.006/0.011	8.75	0.49
1400	1	18.46/11.87/16.42	0.166/0.104/0.147	8.15	0.47
	2	10.44/8.03/9.34	0.092/0.070/0.082	13.58	0.73
	4	7.00/2.00/5.50	0.061/0.017/0.048	15.61	0.81
	8	2.91/1.42/2.46	0.025/0.012/0.021	9.20	0.56
	16	1.59/0.79/1.25	0.014/0.007/0.011	9.35	0.54

い，クライアントにはお茶の水女子大学のSuperSPARC（50 MHz×2, 112 MB）8台構成のWSクラスタを利用した．**表 6**，**7**に1PE版，4PE版の実行結果を示す．

1PE版，4PE版の比較では，LANでの評価と同様の傾向が見られ，WANで運用する際も4PEで実行するライブラリの選択が望ましいと判断できる．

nが大きくなると，**3**で述べたように通信量に対して計算規模が比較的小さいため，WANにおいてもNinf_callの性能は向上する．一方，cが増加する場合はクライアント - サーバ間に多量のデータ転送が起こり，通信スループットの低下が著しい．このため単位時間あたりに到着するNinf_callの数が減少し，$c = 16$となってもサーバの稼働率，負荷ともに余裕を残した．

このように現状の広域ネットワークスループットでは，同じサイトから通信主体の計算を複数投入された場合，十分な性能で処理することが難しい．したがって，サーバの稼働率や負荷だけでなく，ネットワークの混雑具合やタスクの広域配置を考慮した適切な負荷分散アルゴリズムが必要である．

4.2.4 WANでの複数サイト，複数クライアントによる測定結果

WANでの複数サイト，複数クライアントによる評価では，クライアントとしてお茶の水女子大学（Ocha-U），東京大学（U-Tokyo），名古屋工業大学（NITech），東京工業大学（TITech）の4つのサイトのマシンを利用した．サーバには電子技術総合研究所のJ90を用いた．各サイトで用いたマシン，クライアント - サーバ間

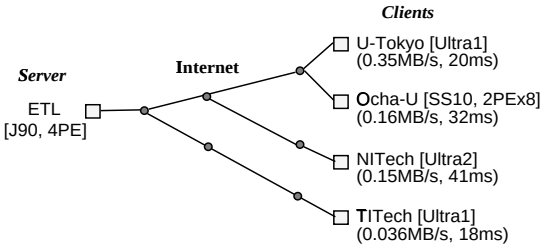


図 7 WAN の計測環境
Fig. 7 WAN multi-client benchmarking environment.

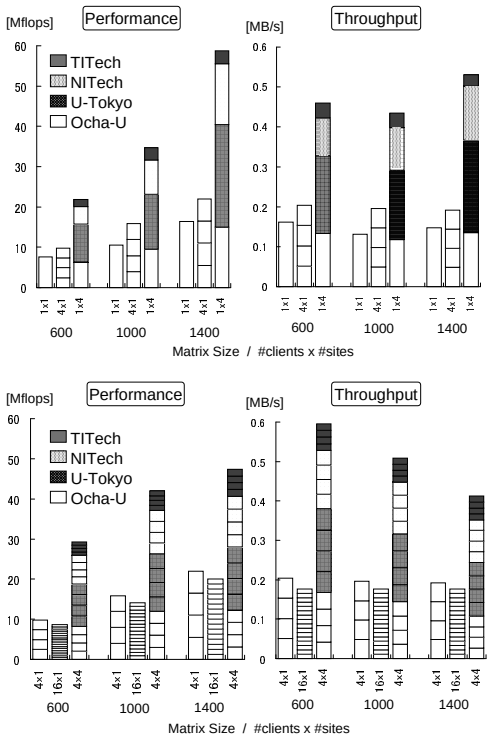


図 8 WAN での複数サイト，複数クライアントによる実行結果
Fig. 8 Multi-client, multi-site WAN Linpack benchmark results.

の ping のレイテンシ， FTP 通信スループット，トポロジを**図 7** に示す． Linpack のルーチンは 4PE 版を用い，各サイトでのクライアント数 c は 1， 4，問題サイズ n は 600 とした．

図 8 に 4 つのサイトを利用し， $c = 1$ （上）， $c = 4$ （下）としたときの実行結果を示す．比較のため，単一サイト（ETL-Ocha-U 間）での実行結果も示した．横軸には問題サイズとクライアント数 $c\times$ サイト数を示し，縦軸にはそれぞれ， Ninf_call の実効性能と通信スループットの総和を示している．

複数サイトを用いた場合の ETL-Ocha-U 間の通信スループットは単一サイトのみの場合と比較すると， $c = 1$ のときで 9〜18%， $c = 4$ のときで 18〜44% しか低下していなかった．したがって，通信スループットの総和は複数サイトからジョブが投入される場合，各クライアント - サーバ間で 1 対 1 のときに観測される通信スループットより大幅に上回ることが分かる．サーバの CPU の稼働率，負荷平均値についても複数サイトの方が高い値となっていた．

複数サイトで $c = 1$ と $c = 4$ の場合を比較すると，通信スループットのばらつきによる影響があるものの，各クライアントでの総合性能の差は少なく， WAN においては 1 対 1 の通信性能により性能が制限されることが分かった．

また，一般にクライアント数と問題サイズの増加によりサーバの計算性能が飽和してしまうと予測されるが， CPU の稼働率は $c = 1$ （計 4）のときで 20.2〜30.3%， $c = 4$ （計 16）のときで 27.1〜34.2% にとどまっており，負荷平均値についても同様であった．したがって，複数サイトを用いた評価においても，通信量の多い計算ではネットワークの負荷により性能が決まるため，サーバの稼働率，負荷平均値のみを考慮しても適切な負荷分散が実現できないことが分かる．

4.3 EP による評価

EP は NAS Parallel Benchmarks[11] のカーネルプログラムの 1 つであり，乗算合同法により一様乱数および正規乱数を生成する典型的なモンテカルロシミュレーションである． Ninf を用いたベンチマークでは，乱数生成の試行をサーバ上で行う．通信量は問題サイズによらず一定で $O(1)$bytes，演算量は乱数を生成した回数とし，試行回数が 2^n のとき 2^{n+1} となる．したがって Ninf_call の所要時間を T_{Ninf_call} とすると Ninf_call の性能は $P_{Ninf\_call} = 2^{n+1}/T_{Ninf\_call}$ のように表せる． Linpack は通信のオーバヘッドが顕在化しやすいのに対し， EP は通信の影響がほとんど現れないという性質を持つ．

評価は LAN/WAN において **4.2.1**， **4.2.3** と同じ環境で行った． EP ルーチンは 2^{24} 回（Sample）の試行を行い， 1PE で実行し 4 タスク並行に処理可能なライブラリを用いた．

4.3.1 LAN/WAN での複数クライアントによる EP の測定結果

表 8 に EP の LAN/WAN での複数クライアントによる実行結果を示す． 4PE 構成の J90 に対し，各 Ninf_call は 1PE で処理されるため，クライアント数 c が 4 までは Ninf_call の性能が維持され， $c = 4$ のときにサーバの稼働率がほぼ 100% に達した．一方， $c = 8$，

表8 EPのLAN/WAN（単一サイト）での複数クライアントによる実行結果
Table 8 Multi-client EP benchmark results for LAN and single-site WAN.

		性能 [Mflops]	通信時間 [sec]	サーバ
n	c	max/min/mean	max/min/mean	稼働率 / 負荷
LAN	1	0.17/0.17/0.17	0.01/0.01/0.01	30.51/1.44
	2	0.17/0.17/0.17	0.01/0.01/0.01	53.86/2.39
	4	0.17/0.16/0.17	0.01/0.01/0.01	98.18/4.18
	8	0.09/0.08/0.08	0.01/0.01/0.01	100.00/8.90
	16	0.04/0.04/0.04	0.02/0.01/0.01	100.00/16.88
WAN	1	0.17/0.17/0.17	0.06/0.04/0.04	25.02/1.21
	2	0.17/0.16/0.17	0.06/0.04/0.04	49.16/2.19
	4	0.17/0.16/0.17	0.24/0.04/0.05	98.14/4.16
	8	0.09/0.08/0.08	0.05/0.04/0.04	100.00/8.91
	16	0.04/0.04/0.04	0.09/0.04/0.04	99.94/16.88

16ではNinf_callの性能が1/2，1/4となっていた．これはEPでは通信量が少ないため，LAN/WANともにNinf_callの性能はほとんどサーバの稼働状態のみに依存するためである．また，複数サイトを用いた場合でも同様の結果が得られると考えられる．

5. 議　　論

5.1 性能保証

利用できるサーバマシンがいくら高性能であっても，1つのサーバに対して多数のアクセスがあると1ジョブあたりの計算性能は保証されない．これに対し，接続数の制限も可能だが，WWWのようにアクセスパターンが断続的である場合とは異なり，Ninfでは計算主体でサーバマシンとの接続が計算終了まで保たれる．よって接続数を単純に限定することは広域ネットワーク上でのNinfサービスの可用性を損なうことを意味する．

ユーザあたりの計算性能を保証するには適切な接続制限を設けるとともに，Ninf_callの手続きを2フェーズ化した枠組みが必要になる．つまり，リモートライブラリの実行に必要な引数の転送が終了した時点で接続をいったん切り，計算終了後，クライアントが計算結果を受け取るために再度接続する．すでに我々はNinfで利用するための数値情報データベースシステム[12]において同種の手法を採用している．Javelin[5]では類似の手法で中間のブローカサーバにジョブを付託する．

また，現状の広域ネットワークの整備状況や帯域予約機能のないプロトコルベースでは，通信量がボトルネックとなりサーバマシンの計算能力を十分に活用できない恐れがある．このような状況を避けるため，現在ではメタサーバとIDLを有効活用する予定である．Ninfでは，IDLと実行時の引数情報により計算要求がサーバに到着時に計算に必要な通信量と計算時間を予測可能である．したがって，計算量主体のジョブと通信量主体のジョブを適切なサーバにメタサーバを介して振り分けることにより，計算能力の有効利用とクライアント実効性能が維持できると予測される．

5.2 サーバのジョブハンドリングメソッド

Ninfシステムが複数クライアントに共有される状況下でも，平均応答時間を短くし，サーバの稼働率を向上させる必要がある．現状のNinfサーバは単純に要求が来ると即座にfork & execするのみでFirst Come First Served（FCFS）で処理されるため，平均の応答時間が長くなり，CPUに余力のある場合にそれ以上稼働率を向上させることができない．一方，NinfではNinf IDLにアルゴリズムの複雑さを記述しておくなどすることで，ジョブの通信／計算時間を比較的正確に予想できるため，Shortest Job First（SJF）で実行する機構を導入することにより，応答時間・稼働率の改善が期待できる．また，SMPではジョブのco-schedulingを適用することにより，システム応答時間を多少犠牲にしても，総合的なシステム性能の向上が望める．

5.3 並列機のためのマルチジョブスケジューリング

本稿ではサーバとして4PE構成のJ90と16PE構成のSPARC SMPを用いた．PE数の増加にともない1ジョブに費すPE数の選択と複数ジョブ間のスケジューリングが問題になる．NinfではIDLに様々な情報を付加しておけるので，問題サイズに応じて利用するPE数の異なるルーチンを呼び出すことも可能である．したがって，スケジューリングのオーバヘッドをおさえたり，大きな問題も高速処理することで平均処理時間を小さくできる．

一方，複数ジョブのプロセッサグループへの割当てを考えると，到着順（FCFS）では先頭のジョブが実行可能になるまでブロックするため，Idle PEが増加し，PE利用率を低下させる．ジョブキュー内で実行可能な最初のものを実行するFit Processors First Served（FPFS）やジョブキュー内で実行可能なものの中で最多のPEを利用するものを実行するFit Processors

Most Processors First Served（FPMPFS）などの改良があるが，Ninfのような高性能広域計算システムに適した方針を選択するには検討が必要である．

6. 関連研究

テネシー大学のNetSolve[3]はNinf_callとほぼ同等のAPIを提供するシステムであり，Agentと称するプロセスを通して負荷分散を実現する．Ninfとの技術的違いとして，NetSolveではクライアントからコードフラグメントをサーバに転送して動的にコンパイル／リンクして実行することが可能である．一方，データフローを解析してスケジュールする機能やDBサーバへのAPIはNinf独自のものである．アダプタを介することにより，Ninfクライアントから NetSolve サーバに登録されているライブラリを，NetSolveクライアントからNinfのサーバに登録されているライブラリを呼び出すことが可能である．

ETHのRemote Computation System（RCS）[6]は複数のスーパコンピュータを統一したインタフェースで利用するためのRPC機能を提供する．通信レイヤにPVMを用いているため広域計算に適しておらず，クライアントAPIがNinfとは異なり拡張性に欠ける．

Ninf，NetSolve，RCSはRPCベースのシステムであるため，各サーバ計算機のアーキテクチャにチューニングされた高速・高精度なライブラリを提供することが可能である．また，ユーザの記述したコードをサーバに転送して実行するシステムよりも安全性の確保が比較的容易であるという利点がある．

Legionはオブジェクトベースの分散実行環境であり，多くの計算資源を統合した仮想計算機を実現する．Legionではシステムレベルのスケジューラを持たず，各アプリケーションに適したスケジューリングポリシーにより，ユーザレベルでスケジューリングを行う（分散プログラミングを支援するオブジェクト指向言語Mentat[13]により書かれたクライアントのコードの実行に対しては，Prophet[14]によってスケジューリング可能である）．

Globus/Nexus[7]のようにシステムアーキテクチャの異なるレベルに着目したものや，Javelin[5]のように移植性や他の問題に着目した高性能広域計算プロジェクトが複数ある．

7. まとめと今後の課題

本稿では，高性能広域計算を支援する基盤システムNinfの性能を解析した．評価は通信／計算の特性の異なるベンチマークにより性能，アーキテクチャの異なるクライアント-サーバマシンを用いてLAN/WAN（単一／複数サイト）環境で行い，以下の結論が得られた．

- 現状のNinfシステムやNetSolveなど類似のシステムはLAN/WAN環境でロバストに機能する．また，SMPではサーバの稼働状態に対し適切なスレッド数の選択が必要だが，最適化された並列ライブラリが広域計算サーバにおいて有効利用できる．
- LAN環境では計算サーバの性能がNinfを用いた性能を支配していたのに対し，現状のWAN環境では性能は通信スループットの制限に大きく依存する．特に単一サイトに複数のクライアントが集中していた場合，通信主体の計算ではすぐにネットワークが飽和した．一方，クライアントが複数サイトに分散している場合では，性能は広域ネットワークのトポロジに依存する．
- EPのように計算量が通信量に比べて圧倒的な計算に対してはLAN/WAN環境の違いによる性能への影響はほとんどなく，現状の広域システムでも実行可能である．
- 一方，クライアント-サーバの通信を2フェーズ化した枠組みの実現など通信プロトコルの改良や，計算サーバでのタスクスケジューリングや高性能広域計算における負荷分散について様々な手法の検討が必要である．

本稿では2つのアプリケーションコアにより広範囲のベンチマークを実施したが，他の特性を持つ実アプリケーションやWANでの異なるトポロジに配置された計算機環境での評価が必要である．しかしながら広域ネットワークで再現性のある大規模ベンチマークを行うことは難しい．そのためNinfの高性能広域計算シミュレータを作成し，容易に様々なパラメータによる評価を実施可能とすることを考えている[15]．さらにNinfシミュレータでは基本的な高性能広域計算モデルでのパラメータを決定するため，より詳細な解析が必要である．

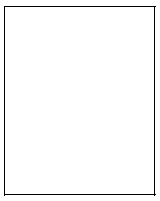
また，Ninfシステムをよりロバストなものとし，広域で様々なアプリケーションからの利用を実現するために，容易に配布できるものとしなければならない．これは一般に高性能広域計算システムの振舞いに関するより詳細な知見を得ることにほかならない．さらに，他の高性能広域計算を目的とする研究と協力し，技術的結果や基盤についてできる限り共有することを考えている．

謝辞 本システムに関して日頃より議論ならびに実現にご協力をいただいた，日本シリコングラフィックス・クレイ（株）つくば・東北事業所，富士総合研究所西川宜孝研究員およびお茶の水女子大学細矢治夫教授に感謝いたします．

参 考 文 献

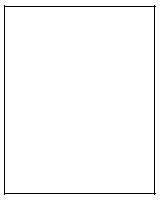
1) Ninf: A Network based Information Library for Global World-Wide Computing Infrastructure. http://ninf.etl.go.jp/.
2) Sato, M., Nakada, H., Sekiguchi, S., Matsuoka, S., Nagashima, U. and Takagi, H.: Ninf: A Network based Information Library for a Global World-Wide Computing Infrastracture, Proc. HPCN'97 (LNCS-1225), pp. 491–502 (1997).
3) Casanova, H. and Dongara, J.: NetSolve: A Network Server for Solving Computational Science Problems, *Proc. Super Computing '96* (1996).
4) 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).
5) 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).
6) Arbenz, P., Gander, W. and Oettli, M.: The Remote Computation System, Technical Report 245, ETH (1996).
7) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *International Journal of Supercomputer Applications* (1997).
8) Nakada, H., Takagi, H., Matsuoka, S., Nagashima, U., Sato, M. and Sekiguchi, S.: Utilizing the Metaserver Architecture in the Ninf Global Computing System, *Proc. HPCN'98* (1998).
9) 中田秀基, 佐藤三久, 関口智嗣: ネットワーク数値情報ライブラリ Ninf のための RPC システムの概要, TR 95-28, 電子技術総合研究所 (1995).
10) 小国, 村田, 三好, ドンガラ, 長谷川: 行列計算ソフトウェア, 丸善 (1991).
11) NPB: NAS Parallel Benchmarks. http://science.nas.nasa.gov/Software/NPB/.
12) 稲木貴光, 松岡聡, 小川宏高: 超広域分散数値情報データベース NinfDB の構築について, 第 56 回情報処理学会全国大会論文集 (1998).
13) Grimshaw, A.S.: Easy to Use Object-Oriented Parallel Programming with Mentat, *IEEE Computer*, pp. 39–51 (1993).
14) Weissman, J. B. and Zhao, X.: Scheduling Parallel Applications in Distributed Networks, *Journal of Cluster Computing (accepted)*.
15) 竹房あつ子, 合田憲人, 小川宏高, 中田 秀基, 松岡聡, 佐藤三久, 関口智嗣, 長嶋雲兵: 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 (1998).

(平成 9 年10月31日受付)
(平成10年 4 月 3 日採録)



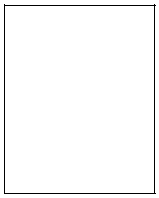
竹房あつ子

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. 現在, 同大学院人間文化研究科複合領域科学専攻博士後期課程在学中. 広域分散システムの研究に興味を持つ.



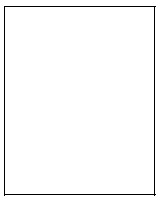
小川 宏高

昭和 46 年生. 平成 6 年東京大学工学部計数工学科卒業. 平成 8 年同大学大学院工学系研究科情報工学専攻修了. 平成 10 年同博士課程中退. 現在, 東京工業大学大学院情報理工学研究科助手. プログラミング言語, 言語処理系, オブジェクト指向技術, 並列計算機アーキテクチャ, 広域分散システムに興味を持つ. ACM 会員.



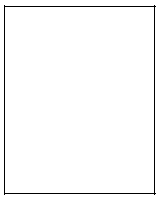
松岡　聡（正会員）

昭和38年生．昭和61年東京大学理学部情報科学科卒業，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科数理・計算科学専攻助教授．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等の研究に従事．現在進行中の代表的プロジェクトは，世界規模の高性能計算環境を構築するNinfプロジェクト，計算環境に適合・最適化を目指すJava言語の開放型Just-In-Timeコンパイラ OpenJIT，制約ベースのTRIPユーザインタフェース．並列自己反映型オブジェクト指向言語ABCL/R2の研究で1996年度情報処理学会論文賞受賞．1997年はオブジェクト指向の国際学会ECOOP'97のプログラム委員長を務める．ソフトウェア科学会，ACM，IEEE-CS各会員．



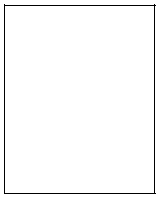
中田　秀基（正会員）

昭和42年生．平成2年東京大学工学部精密機械工学科卒業．平成7年同大学大学院工学系研究科情報工学専攻博士課程修了．博士（工学）．同年電子技術総合研究所研究官．並列プログラミング言語，オブジェクト指向言語，分散計算システムに関する研究に従事．



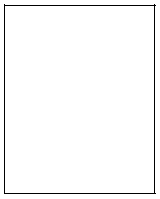
高木　浩光（正会員）

昭和42年生．平成元年名古屋工業大学工学部電気情報工学科卒業．平成6年同大学大学院工学研究科博士課程修了．博士（工学）．同年名古屋工業大学工学部付属情報処理教育センター助手．この間，VLIW型プロセッサのための同期機構とスケジューリング方式に関する研究等に従事．平成10年電子技術総合研究所入所．グローバルコンピューティング，オブジェクト指向言語の最適化コンパイル方式等に興味を持つ．電子情報通信学会，ソフトウェア科学会各会員．



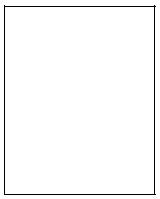
佐藤　三久（正会員）

昭和34年生．昭和57年東京大学理学部情報科学科卒業．昭和61年同大学大学院理学系研究科博士課程中退．同年新技術事業団後藤磁束量子情報プロジェクトに参加．平成3年通産省電子技術総合研究所入所．平成8年より，新情報処理開発機構つくば研究センターに出向．現在，同機構並列分散システムパフォーマンス研究室室長．理学博士．並列処理アーキテクチャ，言語およびコンパイラ，計算機性能評価技術等の研究に従事．日本応用数理学会会員．



関口　智嗣（正会員）

昭和34年生．昭和57年東京大学理学部情報科学科卒業．昭和59年筑波大学大学院理工学研究科修了．同年電子技術総合研究所入所．情報アーキテクチャ部主任研究官．以来，データ駆動型スーパコンピュータSIGMA-1の開発等の研究に従事．並列数値アルゴリズム，計算機性能評価技術，ネットワークコンピューティングに興味を持つ．市村賞受賞．日本応用数理学会，ソフトウェア科学会，SIAM，IEEE各会員．



長嶋　雲兵（正会員）

昭和30年生．昭和58年北海道大学大学院博士後期課程修了．理学博士．同年岡崎国立共同研究機構分子科学研究所助手．平成4年お茶の水女子大学理学部情報科学科助教授．平成8年同教授．平成10年工業技術院物質工学工業技術研究所理論化学研究室長．理論化学，並列分散処理，性能評価の研究に従事．日本化学会，日本応用数理学会，IEEE，ACM各会員．