クライアント・サーバ型のグローバルコンピューティングシステムの比較 — Ninf, NetSolve, CORBA, Ninf-on-Globus の性能評価 —

鈴村 豊太郎† 中川 貴之†
松岡 聡† 中田 秀基††

Ninf, NetSolve, Globus をはじめとするグローバルコンピューティングシステムの近年の発展により，広域ネットワークを利用して高性能計算を提供することが可能となってきている．しかし，それらの研究の多くは，アーキテクチャやアプリケーションの構築例に偏っており，その種のシステムが本質的に満たすべき性質については検証されていない．本稿では，Ninf, NetSolve, CORBA を用いて実アプリケーションを実装し，各システムの比較を行った．その結果，Ninf や NetSolve のようなグローバルコンピューティング専用のシステムは，CORBA と比較し性能面・管理面・プログラマビリティの点のいずれにおいても勝ることが明らかになった。また，Globus の通信サービスである Nexus を用いて，Ninf システムを構築した．その結果，前提とする通信モデルの相違により，繁雑な実装になり性能の低下を招いた．このことから，グローバルコンピューティングにおける理想的なソフトウエアアーキテクチャを選定する為に，多くのシステムにわたる研究が必要であると言える。

Comparison of Client-Server Global Computing Systems — Performance Evaluation of Ninf, NetSolve, CORBA, Ninf-on-Globus —

Toyotaro Suzumura,† Takayuki Nakagawa,†
Satoshi Matsuoka† and Hidemoto Nakada††

Recent developments of global computing systems such as Ninf, NetSolve and Globus have opened up the opportunites for providing high-performance computing services over wide-area networks. However, most research focused on the individual architectural aspects of the system, or application deployment examples, instead of the necessary charactersistics such systems should intrinsically satisfy, nor how such systems relate with each other. Our comparative study performs deployment of example publications of network-based libraries using Ninf, NetSolve, and CORBA. There, we discover that dedicated systems for global computing such as Ninf and NetSolve have clear management, progammability, as well as performance advantages over CORBA. Furthermore, deployment of Ninf on top of Nexus, the communication layer of Globus, has exhibited some loss of performance as well as somewhat kludgy glueing, due to the fundamental difference on the assumptions of the underlying communication models. Such results indicate that further basic research is necessary across multiple systems to identify the ideal software architectures for global computing.

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における大規模計算の要求に応えることが可能となってきている．これを実現するのが，高性能広域計算システム，いわゆるグローバルコンピューティングシステムであり，Ninf, NetSolve, Globus をはじめ，複数のシステムが開発されている．しかし，いずれも実験段階のものであり，実用にはまだいくつかの問題点を抱えているのが現状である．本稿では，グローバルコンピューティングの抱える問題のうち，以下の 2 つに着目した．

(1) グローバルコンピューティング専用のシステムの有効性の確認 — 現状のグローバルコンピューティングシステムでは CORBA を含む従来型の分散コンピューティングのソフトウエア基盤と比較して，優位性が明確には示されていない．ここでの優位性は，性能などの定量的側面，およびユーザビリティ，メンテナンス性などの定性的側面がある．

(2) 各グローバルコンピューティングシステム相互の関係の検証 — 例えば，低位のモジュールを提供する Legion[1)] や Globus[5)] のサービスを用いることにより Ninf[3)] や NetSolve[4)] のようなシステムを簡便に構築できるかどうかの検証は詳細にはなされていない．

本稿では，まず代表的なグローバルコンピューティン

† 東京工業大学 Tokyo Institute of Technology
†† 電子技術総合研究所 Electrotechnical Laboratory

グシステムの概要について述べるとともに, 上に掲げた各問題に対して以下の実験を行った.

(1) の問題点については, 実アプリケーションである半正定値計画問題アルゴリズム SDPA を Ninf, NetSolve, および CORBA の実装の一つである TAO[2) 上に実装し, 性能評価及び各システムの特性を検証・比較した.

(2) の問題点に対しては, 実験的に Ninf の通信機構を Globus の通信サービス Nexus を用いて再実装をし, 評価を行った.

その結果, 前者の実験においては, 広域分散環境上のアプリケーションを構築する場合に, CORBA は Ninf や NetSolve に比べてプログラマビリティやメンテナンス性, パフォーマンスの面で欠如していることがわかり, グローバルコンピューティングシステムの有効性が確認された. また後者の実験により, 現状の Nexus では提供する通信モデル・プログラミングモデルが限定されており, Ninf のようなシステムを必ずしも簡便に構築することはできないという結論に達した.

本稿では 2 章で Ninf, NetSolve および CORBA の解説を行い, 3 章で Globus 上への Ninf の実装について述べる. 次に 4 章でそれらのシステムを使って SDPA および Linpack を実装し, ユーザビリティ, メンテナンス性などの定性的側面から各システムの比較を行う. さらに 5 章では, 4 章で実装したアプリケーションの性能評価を行い, 定量的側面から各システムの比較を行う.

2. グローバルコンピューティングシステムと CORBA

本章では, Ninf, NetSolve, および CORBA の簡単な説明を行う.

2.1 Ninf, NetSolve

Ninf および NetSolve は, ネットワーク上に存在する高性能な計算機を用いて, 大規模な科学技術計算を高速に実行することを目的としたシステムである. これらは基本的にクライアント・サーバ型モデルである. クライアントは, ネットワーク上のサーバに対して計算ライブラリを呼び出し, 実際の計算はサーバ上で行われる. 計算が終了すると, 結果がクライアントに返される.

Ninf の基本パッケージがサーバとクライアントであるのに対して, NetSolve はそれにエージェントを加えた形態でのみ動作する. エージェントは, 計算資源の状態や問題の特徴などの情報から計算時間を予測し, 負荷分散を行う.

ライブラリ提供者はその仕様を独自の IDL を用いて記述し, それを元に実行スタブを得る. 一方, クライアントは非常に簡潔な API の記述のみでライブラリを呼び出すことができる.

2.2 CORBA

CORBA は, 分散オブジェクトコンピューティングの標準仕様である. CORBA は, ネットワークプログラミングにおける多くの共通するタスクを自動化する機能を持つ.

CORBA システムにおいて, 計算を行うのは分散オブジェクトである. これらを管理するのが ORB (Object Request Broker) であり, クライアントとオブジェクトとの通信メカニズムを提供する機能を持つ. クライアントがオブジェクトを呼び出すと, ORB はオブジェクトを探しだし, 必要があれば活性化し, リクエストをオブジェクトに伝達して, 結果をクライアントに返す.

CORBA におけるインターフェイスは, CORBA IDL により記述する. CORBA IDL は言語非依存であり, C, C++, SmallTalk など多くの言語バインディングを持つ. またプラットフォーム独立で, CORBA IDL を用いて定めたインターフェイスは高い可搬性を持つ.

3. Globus 上への Ninf の実装

本章では, Globus の通信モジュール Nexus[6) を用いて, Ninf システムの再構築を行う.

3.1 Ninf の通信機構

Ninf は前述したようにクライアント・サーバ型システムでクライアントとサーバ間のプロトコルは TCP/IP を用いたストリーム通信で行われ, クライアント, サーバ双方のバッファを介してデータの授受を行う. また, Ninf はマスター・スレーブ式並列実行サーバで, マスターサーバは特定のポートからのコネクションを受け付けるとスレーブサーバプロセスを一つ起動する. 実際のコネクションに対する処理はスレーブサーバが受け持ち, マスターサーバは常にポートからの接続を受ける.

3.2 Nexus

Globus は, 資源管理, セキュリティ, 広域に分散した計算資源の情報収集, 通信, 耐故障性, 遠隔データアクセスなどのグローバルコンピューティングに必要なサービスを提供するツールキットである. 本稿では, Globus の通信サービス Nexus のみを用いる. Nexus は異機種並列分散環境用のポータブルなマルチスレッドライブラリでグローバルコンピューティング上の言語やライブラリ, アプリケーションの開発を支援する. また, ユニキャスト, マルチキャストのメッセージ転送ができ, 広範囲の通信プロトコル上で効率的な通信を可能としている.

Nexus の通信機構は次の 2 つの抽象化によって定義される.

- **Communication link**
  図 1のように通信の開始点と終了点を結びつけることによって形成される.
- **Remote Service Request(RSR)**
  RSR は一方向の非同期 RPC で, 通信開始点から Communication Link によって関連付けられた通信終了点までデータを転送すると, 通信終了点を含むプロセス内でスレッドが起動しそのデータを収集する.

Nexus の通信操作は, この RSR を通信の開始点に適用することで開始する. また, 図 1のように一つの通信終了点が複数の通信開始点に対して Communication Link をはることができ (逆も可), 複雑な通信構造を構築することが可能である.

3.3 実　装

Nexus の通信セマンティックスではリモートスレッドの invocation が核となっており, それによるデータ転送は副次的なものなので Ninf のようなストリーム通信とは相性が悪い. 従って, Ninf のストリーム通信に代わる

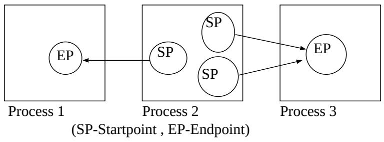

図 **1**　Nexus の通信機構

通信機構を Nexus の通信セマンティックスを用いて実現する必要がある．その方法として次の 2 つの方法が考えられる．

- **Ninf** の通信セマンティックスの変更
  送信側のプロセスがスレッドを起動させ，そのスレッドが直接 Ninf のバッファにデータを書き込むことでストリーム通信と同等の機構を実現する．
  この方法は，Nexus の通信セマンティックスに沿っているため，Nexus の性能を最大限に引き出せるという利点があるが，Ninf の通信セマンティックスを変更しなければならないという欠点を持つ．
- **Nexus** によるストリーム通信のシミュレート
  Ninf の構造を保つため，あらかじめクライアント，サーバ双方に Ninf のバッファとは別のバッファ (Globus Buffer) を設け，それを介してデータの授受を行うことにする．また，受信側のプロセスがスレッドを起動させ，送信側から受け取ったデータを Globus Buffer に書き込み，そのバッファから Ninf のバッファにデータ転送することでストリーム通信をシミュレートする．この方法は，Ninf の通信構造を変更せずに通信部分を置き換えることができるが，シミュレートのオーバヘッドにより性能の低下が予想される．

Globus はグローバルコンピューティングの上位レイヤためのツールキットという位置付けであり，第一の方法のように上位レイヤを変更するのは Globus の意に反する．よって，我々は第二の方法を採用した．

また，Ninf のマスタースレーブ式並列実行サーバの機構は複数のプロセスを作成する代わりに単一プロセス内に複数のスレッドを作ることで実現した．

実装の概要を図 2に示す．

3.4　考　　察

Nexus の通信セマンティックスは，前述のように Ninf のようなストリーム通信とは相性が悪い。よって，Ninf のようなストリーム通信に代わる通信機構を実現する必要がある．この実験では Nexus の通信セマンティックスを用いて，ストリーム通信をシミュレートするという方法をとった．しかし，余計なデータ転送が必要になり，結果的に実装が繁雑になってしまった。

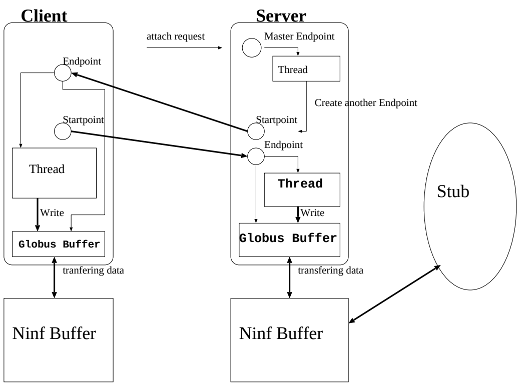

図 **2**　Nexus を用いて実装した Ninf の通信機構

この構築実験によりグローバルコンピューティングの下位レイヤをサポートするシステムは標準との親和性が重要であり，広範囲の通信モデル，プログラミングモデルが必要であるという知見が得られた．

4.　アプリケーションの実装および定性的評価

本章では，各システムに実装したベンチマークプログラム Linpack および実アプリケーション SDPA の概要の説明と，実装段階までにおける各システムの定性的評価を行う．

4.1　Linpack の概要

Linpack Benchmark は，ガウスの消去法を用いた密行列の連立一次方程式の求解に要する時間を測定する．演算数が一意に定まるため，浮動小数点演算の性能の標準としてよく引用される．

4.2　SDPA の概要

SDPA (Semidefinite Programming Algorithm) は，半正定値計画問題 (SDP)[8] の最適解を求めるソフトウェアである．SDP は LP を対称行列の空間に拡張した数理計画問題である．アルゴリズムに用いられている内点法は，従来の単体法では解けないような超大規模な問題を高速に解く計算手法として，近年めざましい発展を遂げている．

SDPA は，特定のフォーマットに従って記述されたファイルから行列データを読み込み，結果をファイルに保存する．このように，プリミティブでないデータ型を入出力として扱う方法は，もともと Ninf や NetSolve では想定されていなかった使い方である．この問題を解決するために，次のような方法を用いて実装を行った．クライアントはまず，ファイルに格納されている入力データをメモリ領域にコピーし，それをサーバに送信する．サーバ側では，メモリイメージとして受け取ったデータを一時ファイルに格納し，それを SDPA に渡す．出力データの場合はこの逆の操作を行う．

4.3　グローバルコンピューティングシステムへの要請と比較評価

グローバルコンピューティングシステムに求められるものとして，クライアントの小ささやプログラマビリティ，サーバの機能性やメンテナンス性，などがあげられる．ここでは，各システムがこれらの要件をどれほど満たしているかを比較検討する．

クライアント・サーバ型システムにおいて，クライアントの小ささは最も重要な事項の一つである．既存のプログラムを書き換える際，その作業はユーザに負担のかからない，かつ簡便なものである必要がある．Ninf および NetSolve では，この作業がそれぞれソースコード 1 行の書き換えですむ．具体的には，

```
func(arg1, arg2, ...);
```

というライブラリ呼び出しを，それぞれ

```
Ninf_call("func", arg1, arg2, ...);
```

```
netsl("func()", arg1, arg2, ...);
```

とするだけである．一方 CORBA のプログラミング手法は難解であり，これを習得することはユーザにとって負担が大きい．また，クライアントプログラムは IDL コンパイラが生成するスタブを必要とし，プログラミングの際，スタブが定義するプログラムの枠組みに沿ったコードを書く必要がある．以下にその一部を示す．

```
CORBA::Environment env;
CORBA::ORB_var orb = CORBA::ORB_init
    (argc, argv, 0, env);
CORBA::Object_var object =
    orb -> string_to_object (ior, env);
Linpack_var s = Linpack::_narrow
    (object.in (), env);
s -> linpack
    (a, lda, n, ipvt, b, info, env);
```

システムそのものの小ささも重要である．インストール作業を何度か要する場合も考えられることや，クライアントが必ずしも十分なストーレジを所有しているとは限らないことから，インストールに要する時間やディスク領域は少ない方が好ましい. 各システムのインストールに要する時間およびディスク領域は表 1 のようになった. この表が示す通り，TAO のインストールに要する時間およびディスク領域は，Ninf, NetSolve と比べて大変大きくなっており，環境によっては致命的な問題ともなり得る．

	時間	ディスク領域
Ninf	2 ~ 3 分	5 ~ 6 MB
NetSolve	3 ~ 5 分	約 25 MB
TAO	約 2 時間半	約 550 MB

表 1　インストールに要する時間およびディスク領域

次に，サーバに計算ライブラリを実装する場合の簡便さを考える．すでに存在する計算ライブラリを Ninf, NetSolve に実装するには，それぞれ独自の IDL を記述する必要がある．Ninf IDL は ANSI C を基本としていて書きやすく，記述量も少ないが，NetSolve の IDL は独自の記法を用いているため慣れを要し，また記述量も多い．Linpack の場合，IDL ファイルのサイズが Ninf では 247 バイト，NetSolve では 711 バイトとなった．図 3 に Ninf IDL の例を，図 4 に NetSolve IDL の例を示す．また CORBA に関しては，IDL はシンプルで記述性も非常に高いが，サーバプログラムを別に記述する必要があり，全体の記述量は多くなる．Linpack の場合，合計で 3044 バイトとなった．

```
Module linpack;

Library "linpack_flib.a";
FortranFormat "%s_";

Define linpack (IN double a[lda:n][n], IN int lda,
                IN int n, OUT int ipvt[n],
                INOUT double b[n], OUT int *info)
CalcOrder n^3
Calls "Fortran" linpack (a, lda, n, ipvt, b, info);
```

図 3　Ninf IDL

また，新たに構築した計算ライブラリをサーバに登録することが，サーバを一時的に止めることなしに可能かどうかも重要である．Ninf は稼働中のサーバに計算ライブラリを登録するユーティリティを持つが，NetSolve ではサーバの再起動を要する．オブジェクトごとにサーバを起動する CORBA も同様である．

```
@PROBLEM linpack
@FUNCTION linpack
@LIB $(HOME)/work/linpack/linpack_flib.a
@LANGUAGE FORTRAN
@MAJOR COL
@PATH /Linpack/
@DESCRIPTION
Linpack

@INPUT 3
@OBJECT MATRIX D A
An input matrix
@OBJECT VECTOR D B
An input vector
@OBJECT VECTOR I IPVT
An in&out vector

@OUTPUT 4
@OBJECT SCALAR I INFO
A status information
@OBJECT MATRIX D OUTA
An output matrix
@OBJECT VECTOR D OUTB
An output vector
@OBJECT VECTOR I IPVT
An in&out vector

@COMPLEXITY 2,3

@CALLINGSEQUENCE

@ARG I0,O1
@ARG mI0
@ARG nI0,lI0,mI1,mI2
@ARG I2,O3
@ARG I1,O2
@ARG O0

@CODE

int *lda, *n;

lda = @mI0@;
n = @lI0@;
@O0@ = (int *)malloc(sizeof(int));
@O1@ = @I0@;
@O2@ = @I1@;
@O3@ = @I2@;
linpack(@O1@, lda, n, @O3@, @O2@, @O0@);

@END_CODE
```

図 4　NetSolve IDL

以上の比較により，次のような知見が得られる．

・ Ninf, NetSolve は扱いやすいシステムであるといえる
  – クライアント API が簡便
  – インストールが容易
  – IDL 文法の知識だけでサーバ構築が可能
  * NetSolve はサーバ運用の手間が若干煩雑
・ CORBA は誰にでも簡単には利用できない
  – クライアントにかかる負担が大きい
  – インストールに多大なディスク領域, 時間を要する
  – 複雑な CORBA プログラミングの知識が必要

これによって, 専用のグローバルコンピューティングシステムの有用性が一つ示されたといえる.

5. グローバルコンピューティングシステムの性能比較

本章では，SDPA と Linpack を用いた，各グローバルコンピューティングシステムおよび CORBA の性能評価について説明する．

5.1 評 価 環 境

サーバには, 東工大の Ultra60 (UltraSparc2 300MHz x2, 256MB) を用いた. クライアントは LAN の場合, サーバと 100BaseTX Ethernet で接続されている Ultra2 (UltraSparc2 200Mhz x2,256MB) を用いた. また, WAN の場合, 電総研の SparcStation5 (MicroSparc2 85Mhz, 32MB) を用いた. この計算機は, 東工大のサーバと平均約 3Mbps でつながれている.

5.2 評 価 手 法

評価には前章で解説した Linpack および SDPA を用いた. 評価手法は Linpack の場合, 問題サイズを n とすると演算量は $2/3n^3 + 2n^2$, 通信量は $8n^2 + 20n$ と表され, Linpack の全経過時間を T_{all}, 通信時間を T_{comm} とすると, 実行性能 P と通信スループット T は $P = (2/3n^3+2n^2)/(T_{all}), T = (8n^2+20n)/(T_{comm})$ と表される. すなわち, 問題サイズが増大するにつれ，全体の実行時間に対して通信時間が隠蔽される．

また SDPA においては, サーバ上でスタンドアロン実行した際の性能値を 1 とした性能比で実行性能を表し, 総転送データ量を転送時間で割ることで通信スループットを求めた.

評価に用いたシステムは，Ninf, NetSolve, CORBA の実装の一つである TAO, 3 章で実装した Ninf-on-Globus の計 4 つである．ただし TAO に関しては，今回クライアントとして使用した計算機のネットワーク設定の都合上，WAN での実験ができなかったため，LAN による実験のみを行った．

5.3 評価結果

5.3.1 Linpack

Linpack の LAN 上での実行性能の実測値を図 5 に，通信スループットの実測値を図 6 に示す．また，WAN での実行性能の実測値を図 7 に，通信スループットの実測値を図 8 に示す．

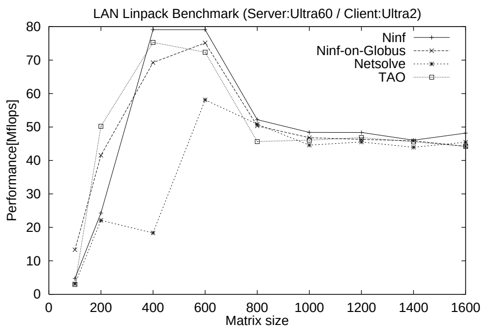

図 5　LAN 上における Linpack 実行性能

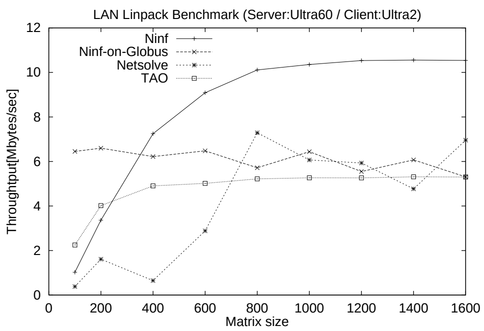

図 6　LAN 上における Linpack 通信スループット

実行性能においては, Ninf-on-Globus 以外の三者にはあまり違いが見られないが, 通信スループットは Ninf が最も高く, NetSolve, TAO はやや低い値となっている. NetSolve では, 総じて Ninf よりも 1 ~ 2 秒多く通信時間を費やしており, エージェントによって行われるクライアントとサーバの仲介や, 計算時間の予測などによるものと推測される．また TAO では，ORB によるオブジェクトリファレンスの取得，およびオブジェクトの活性化にかかるコストのためと思われる．

Ninf-on-Globus は問題サイズ 400 から 600 の時, オリジナルの Ninf と比べて実行性能の低下が見られる. これは 3 章で解説したように Ninf-on-Globus の余分なバッファコピーが影響している為で, 図 6と図 8のスループットの結果に顕著に表れている．しかし, 問題サイズが大きい時には通信量がほとんど無視できるようになるので, 性能差がなくなる.

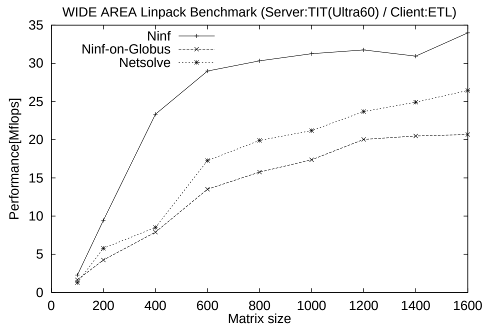

図 7　WAN 上における Linpack 実行性能

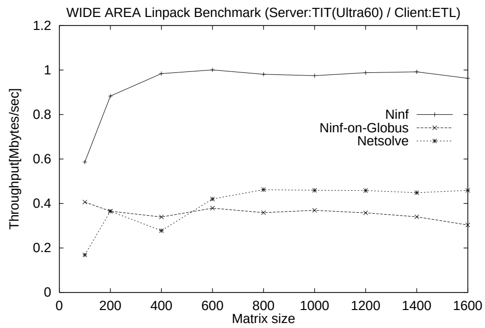

図 8　WAN 上における Linpack 通信スループット

WAN では Ninf の性能の高さが目立つ．NetSolve および Ninf-on-Globus では，問題サイズが大きい場合でも Ninf よりも実行性能が劣る結果となっている．これは WAN の場合，通信量が全経過時間に対してドミナントになってしまうためである．

5.3.2 SDPA

次に SDPA の LAN 上での実行性能の実測値を図 9 に, 通信スループットの実測値を図 10 に示す. また, WAN での測定結果を図 11 に, 通信スループットの実測値を図 12 に示す.

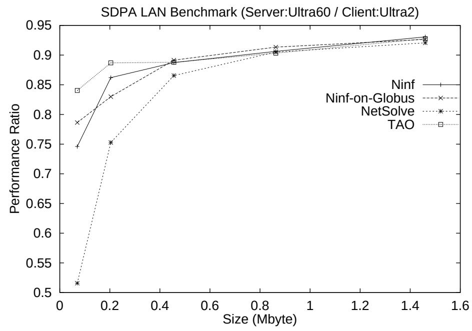

図 9　LAN 上における SDPA 実行性能

各システムとも，ほぼ同じ実行性能を示した. 今回計測したデータでは，データ量が大きい場合，通信コストが実行時間に比べて非常に小さく，通信時間がほぼ隠蔽されていると考えられる．特に NetSolve では，

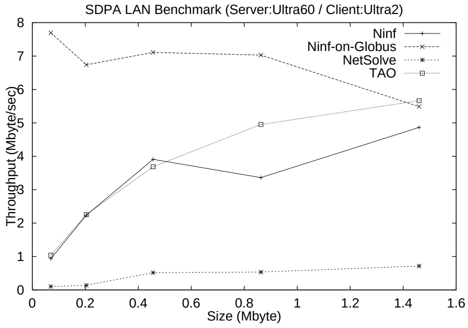

図 10 LAN 上における SDPA 通信スループット

Linpack と同様スループット値は低いが，データ量が大きくなると実行性能の差はほとんどなくなる．ここでは Ninf-on-Globus が最も高い通信スループットを示しているが，これは UDP を用いて通信をしているためと考えられる．

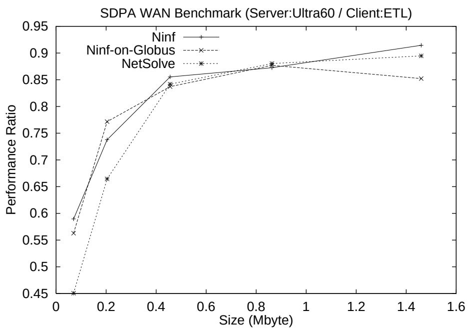

図 11 WAN 上における SDPA 実行性能

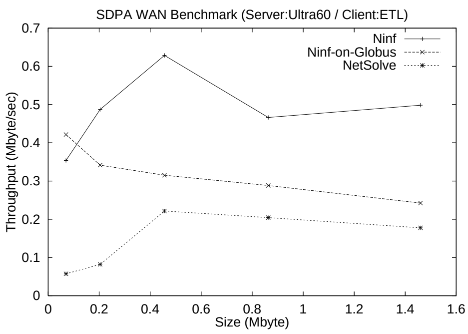

図 12 WAN 上における SDPA 通信スループット

WAN での測定においては，Linpack と同様 Ninf が最も高い通信スループットを示している．Ninf は LAN, WAN ともデータ量が大きい場合にスループット値が落ち込んでいるが，この原因は現在調査中である．

6. まとめ

本稿では, まず第一に Ninf の通信機構を Globus の通信モジュール Nexus を用いて実装した Ninf-on-Globus を構築し, Nexus がグローバルコンピューティングシステムに有用なモジュールであるかを検証した. また, 次に Ninf, NetSolve, Ninf-on-Globus, CORBA を用いてベンチマークプログラム Linpack, 実アプリケーションである半正定値計画問題 SDPA を実装し, 各システムの特性を明らかにするとともに, LAN 並びに WAN 上でその性能評価を行った.

その結果, 前者の実験で構築した Ninf-on-Globus はオリジナルの Ninf と比較すると, Linpack の WAN 上の計測では最大 10% 低い実行性能を示した. これは, Nexus が限定したプログラミングモデルしか提供していない為, Ninf のストリーム通信を実現するには繁雑な実装になり, そのオーバヘッドが影響したためと思われる. この実験によりグローバルコンピューティングの下位レイヤをサポートするシステムは広範囲の通信モデル, プログラミングモデルが必要であるという知見が得られた.

後者の実験では, 実装段階で CORBA が Ninf や NetSolve に比べてメンテナンス性, プログラマビリティ, パフォーマンスの点で劣ることがわかった. そして, Linpack の性能評価では, WAN 上では通信がドミナントになり全体の実行性能に影響を及ぼすことがわかった. また, Linpack の LAN 上におけるスループットの結果では, Ninf が CORBA よりも上回る結果を示し, グローバルコンピューティングシステムの有効性が示されたと言える.

これらの評価によって現状のグローバルコンピューティングシステムに欠けている事項が明らかになり, 次バージョンの Ninf の構築に対する技術的知見を得ることができた.

今後の課題として以下の事項が挙げられる。

- TAO の WAN 上における性能評価を行う.
- これらのシステムを用いて, 他の実アプリケーションも実装し, 性能評価を行う.
- Nexus 以外の Globus のモジュールや Legion についても検証する.

謝　辞

本研究を行うにあたり, 御指導並びにご討論いただいた東京工業大学小島研究室ならびに京都大学の藤沢助手, Ninf グループに深く感謝致します.

参考文献

1) http://www.cs.virginia.edu/~legion/.
2) http://www.cs.wustl.edu/~schmidt/TAO.html.
3) http://ninf.etl.go.jp/.
4) http://www.cs.utk.edu/netsolve/.
5) http://www.globus.org/.
6) I. Foster, C. Kesselman, and S. Tuecke. The nexus approach to integrating multithreading and communication. In *J. Journal of Parallel and Distributed Computing*, pp. 70–82, 1996.
7) Steve Vinoski. CORBA:: Integrating Diverse Applications Within Distributed Heterogeneous Environments. *IEEE Communications Magazine*, Vol. 14, No. 2, February 1997.
8) 小島政和. 半正定値計画問題と内点法. Technical report, 応用数理 Vol.6, April 1996.