ハイパフォーマンス
コンピューティング　88−5
（２００１．１０．２６）

Globus Toolkit の NEC SX-4/5 への移植および評価

川戸正裕[†]　蒲池恒彦[†]　妹尾義樹[†]
田中良夫[††]　関口智嗣[††]

Grid コンピューティングのための事実上の標準ミドルウェアである Globus Toolkit を，NEC のスーパーコンピュータ SX シリーズに移植した。移植作業は NEC と産業技術総合研究所の共同で行ない，Globus を使った遠隔ジョブ実行や，Globus 上での MPI の実装である MPICH-G の実行に成功した。Globus を SX 上で動かすために，環境設定における SX の検出を行なうコードの追加や，I/O 関連のアーキテクチャ依存のコードを SX 向けに変更する作業を行なった。性能評価として，SX 上で MPICH-G と SX ネイティブの MPI ライブラリである MPI/SX を使用して，レイテンシとスループットの比較を行なった。この実験で，MPICH-G の方が 200 倍以上遅いという結果が得られた。この差の原因は TCP/IP や Globus I/O プロトコルのオーバーヘッドであり，MPICH-G の内部で MPI/SX を使うことで性能が改善すると考えられる。今後，MPICH-G と MPI/SX の組み合わせへの対応のほか，ジョブキューイングシステム NQS を使った Globus ジョブの実行への対応作業を行なう予定である。

Porting Globus Toolkit to NEC SX-4/5 and Evaluation

Masahiro Kawato [†], Tsunehiki Kamachi [†], Yoshiki Seo [†],
Yoshio Tanaka [††] and Satoshi Sekiguchi [††]

We have ported Globus Toolkit, the de facto standard Grid middleware, to NEC supercomputer SX series. We have succeeded in remote job execution via Globus and communication via MPICH-G, which is the MPI implementation on Globus. The porting work was carried out as collaboration between NEC and AIST (National Institute of Advanced Industrial Science and Technology). Some minor code modifications were required to run Globus on SX, such as codes for detecting SX and architecture dependedent I/O codes. As performance evaluation, we compared throughput between MPICH-G and MPI/SX (native MPI library on SX). The result was that MPICH-G was over 200 times slower than MPI/SX. The main reason of the reslut is high overhead of TCP/IP and Globus I/O protocol. Therefore using MPI/SX in MPICH-G will bring performance improvement. We are going to combine MPICH-G and MPI/SX and enable job execution with NQS batch queuing system in the future.

1. はじめに

1990 年代後半から，高性能計算機や実験装置を WAN (Wide Area Network) で接続して，単独の計算機や装置では不可能な処理を実現する **Grid コンピューティング**[1] が盛んになってきた。これまでの Grid に関する活動は米国の大学や研究機関が中心となっていたが，近年，日本国内でも計算機センター間をつなぐ Gbps クラスの高速ネットワークの構築など，Grid コンピューティングを行なう環境が整いつつある。

Grid コンピューティングを行なうためには，ハードウェアやネットワークだけでなく，遠隔の資源に効率良く，かつ安全にアクセスするためのソフトウェア・インフラストラクチャ (Grid ミドルウェア) が必要である。現在，事実上の標準 Grid ミドルウェアとして **Globus Toolkit**[2] (以下，Globus) が知られており，Globus を利用して Grid を構築するのが最も現実的である。しかしながら，Globus は，日本の計算機センターで多く使われている NEC の SX シリーズなどのスーパーコンピュータに対応していなかった。このため，これらのスーパーコンピュータへの Globus の移植は Grid 技術の発展に大きく寄与することになる。

このような背景から，Globus の SX への移植作業を NEC と産業技術総合研究所 (以下，産総研) の共同作業として行ない，現在 Globus を使った遠隔ジョブの実行に成功している。本稿では，この移植作業の概要と，Globus の SX への移植版を使った性能評価の結果

† 日本電気 (株) インターネットシステム研究所
Internet Systems Research Laboratories, NEC Corporation
†† 独立行政法人 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology

を述べる。

以下，本稿の構成は次の通りである。2 章で，Globus の構成や基本的なジョブ実行の機構について述べる。3 章で，今回の移植作業において SX 上で実現している Globus の機能を示す。Globus を SX 上で動かすために行なった変更の概要を 4 章で述べる。5 章では，現時点での Globus の移植版の性能評価の結果を示す。評価は，Globus を使ったプロセス間通信の性能について行なっている。最後に，6 章で結論と今後の予定を述べる。

2. Globus の概要

ここでは，以降の章で前提としている Globus の機能および構成の概要について述べる。

Globus が持つ基本的な役割は，多様な計算資源に対して単一のインタフェース (API) を提供し，計算資源の利用者と個々の計算資源の間の相互作用を仲介することである。Globus の API は，アプリケーションによって直接使われるよりも，むしろ上位レベルのミドルウェアに使われることを意識して設計されている。そのような上位レベルのミドルウェアの 1 つに，Globus 上での MPI (Message Passing Interface)[4] の実装である MPICH-G が知られている。

2.1 Globus のコンポーネント

Globus は，機能ごとに分かれた複数のコンポーネントから構成される。主な Globus のコンポーネントには以下のものがある。

- GSI (Grid Security Infrastructure): ユーザとホストの認証 (authentication) の機能を提供する。GSI は，遠隔プロセスの起動やプロセス間通信の際に使われる。
- GRAM (Globus Resource Allocation Manager): ネットワークを介したジョブ管理 (遠隔ジョブ管理) の機能を提供する。遠隔ジョブ管理には，プロセスの起動，ポーリング，強制終了が含まれる。
- DUROC: 複数のプロセスを同時に起動する (co-allocation) 機能を提供する。co-allocation は，Grid 上での並列計算の実現に必要な機能である。
- MDS (Metacomputing Directory Service): サーバの名前，機種，負荷の状態などの情報を管理する情報サービスの機能を提供する。
- Globus I/O: プロセス間通信を中心とする I/O 関連の機能を提供する。接続時に GSI を使った認証を行なうことができる。
- GASS (Global Access to Secondary Storage): 遠隔のファイルサーバへの読み書きの機能を提供する。

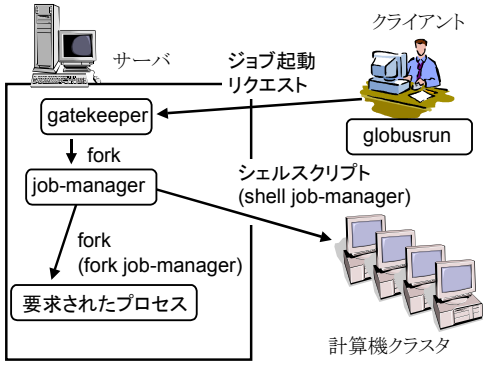

図 1 遠隔ジョブの実行手順

これらのコンポーネントはライブラリの形で提供され，アプリケーションや Globus の上で動くミドルウェアとリンクすることによって利用することができる。

2.2 遠隔ジョブの実行

Globus を使って実現される最も基本的なサービスは，遠隔のホスト上でプログラムを実行する遠隔ジョブ実行である。ユーザは **globusrun** コマンドを呼び出すことによって遠隔ジョブを実行することができる。遠隔ジョブ実行は GRAM の提供する機能であるが，GSI や Globus I/O などの他のコンポーネントも GRAM から利用される。

遠隔ジョブの実行には，以下の 3 つのプロセスが関わる。これらは前節で示した GSI や GRAM などの複数のサービスを組み合わせて実現されている。

- gatekeeper: クライアントからの接続を監視して，job-manager などの他の Globus 関連のサービスを起動する。
- job-manager: サーバホスト上で起動され，クライアントからの要求に従ってジョブを起動／制御する。
- globusrun: ユーザによりクライアントホスト上で起動され，サーバホストに遠隔ジョブ起動／制御の要求を送ったり，結果を受け取る役割を持つ。

図 1 に，これらのサービスを使って遠隔ジョブを実行する手順を示す。

(1) ユーザは，クライアントホスト上で，globusrun コマンドを実行する。引数として，サーバのアドレス・ポート番号と，起動する実行ファイル名を指定する。

(2) globusrun は，サーバホスト上の gatekeeper に

接続し，遠隔プロセス起動リクエストを送る。このとき，ユーザとサーバホストの間での認証が行なわれる。

(3) gatekeeper は，job-manager を起動し，クライアントからの要求を job-manager に引き渡す。

(4) job-manager は，クライアントに指定されたプロセスを起動する。job-manager には，fork システムコールを使ってプロセスを起動する **fork job-manager** と，シェルスクリプトを使ってプロセスを起動する **shell job-manager** の 2 種類がある。

shell job-manager は，並列計算機や計算機クラスタで，NQS (Network Queuing System) や LSF (Load Sharing Facility) などのジョブキューイングシステムを通してジョブを実行するのに用いられる。ジョブの実行は，ジョブキューイングシステムを操作するコマンド (NQS の場合は qsub, qstat など) を記述したシェルスクリプトを呼び出すことによって行なわれる。シェルスクリプトはジョブキューイングシステムの種類ごとに別のものが必要であり，いくつかのスクリプトが Globus の配布に含まれている。

gatekeeper の起動方法には，クライアントからの接続時に inetd から起動する方法と，スタンドアロンのデーモンとして常駐する方法の 2 種類がある。前者の場合には gatekeeper はスーパーユーザ権限で実行されるのに対し，後者の場合には一般ユーザ権限で実行され，job-manager と新しいプロセスも gatekeeper と同じユーザの権限で実行される。

3. SX 上で実現している Globus の機能

この章では，Globus の SX への移植作業の状況として，実験環境と，現時点で実現している Globus の機能を述べる。

3.1 実 験 環 境

SX 上の Globus の実験は，つくばギガビットラボ (つくば情報通信研究開発センター，http://www.tsukuba.tao.go.jp/) に設置されている SX-4 を使って行なった。ギガビットラボは，通信・放送機構が運営する研究開発用高速ネットワーク「Japan Giganet Network (JGN)」の一部で，高速ネットワークの基盤・応用技術を研究するための共同設備として提供されている。

基本的な実験環境を**図 2**(A) に示す。つくばギガビットラボの SX-4 はプライベートアドレスしか持たないため，産総研のワークステーションから SX-4 を使うためには，基本的にゲートウェイを経由してログインする必要がある。この場合，SX と外のマシンの間では直接通信できないため，クライアントとサーバの両方を SX 上で動かすことになる。しかし，図 2(B) のように，ギガビットラボ側の NAT (Network Address Translation) ルータを利用すれば，外部のマシンと直接通信することが可能になり，産総研のマシンをクライアント，SX をサーバとすることもできる。

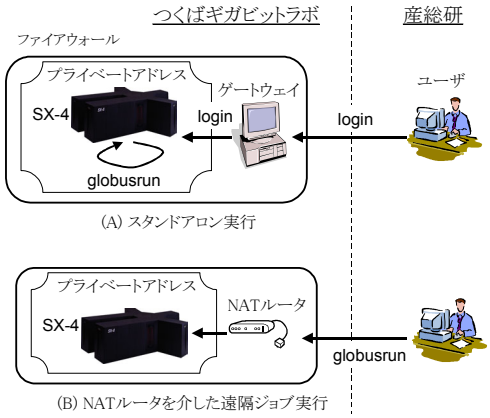

図 2 実験環境

3.2 実現している Globus の機能

現在，Globus の資源管理機能を使って，以下の 3 種類の動作テストに成功している。

- globusrun を使った，SX 上での単体実行 (図 2(A))。
- globusrun を使った，産総研のクライアントからの遠隔プロセス実行 (図 2(B))。
- mpirun (MPICH-G) を使った，SX 上での単体実行。

これらの動作テストは，以下のような条件の下で行なっている。

- job-manager には fork job-manager を用いている。(2.2 節参照)。
- gatekeeper は一般ユーザ権限で (i.e. スタンドアロンのデーモンとして) 起動する (2.2 節参照)。これは，SX 上での動作はまだ実験段階のため，スーパーユーザ権限での動作を避けるためである。

したがって，shell job-manager を使う場合や，gatekeeper を inetd 経由で実行する場合については，今後動作確認を行なう必要がある。特に，shell job-manager については，SX の標準キューイングシステムである NQS に対応するスクリプトを作成する必要がある。

この動作テストにより，以下の Globus のコンポーネントが SX-4 上で正常に動作することを確認した。

- GSI (ジョブ実行の際のユーザ／ホスト認証)

- GRAM (遠隔ジョブ実行)
- DUROC (MPICH-G における複数プロセスの起動)
- Globus I/O (クライアント／サーバ間の通信)
- GASS (クライアントと遠隔プロセスの間での標準入出力の転送)

ただし，これらのコンポーネントのすべての機能が使われているわけではないため，実用的に使うためには，より詳細な実行テストを行なう必要がある。また，2 章で示した Globus のコンポーネントのうち，MDS については今回の実行テストでは使われていない。

4. SX への移植作業の概要

この章では，作業・実験環境と，SX への対応のために Globus に加えた修正の内容を述べる。

移植の元になるソースには，Globus バージョン 1.1.3 を使用した。2001 年 9 月時点で公開されている最新版である 1.1.4 とは，I/O 関連にわずかな違いがあるだけで，ほぼ同じものである。

SX シリーズはすでに Globus の動作が確認されているシステムとは異なったアーキテクチャを持っていることから，Globus のソースコードへの変更が必要になることが予想された。具体的には，主に以下の点に注意する必要があった。

- configure スクリプト: Globus のソースには，GNU autoconf によって作成された自動環境設定 (configure) スクリプトが含まれる。configure スクリプトが SX を想定していない場合，SX のための記述をスクリプトを追加する必要がある。
- SX のアーキテクチャへの対応: SX の主な特徴に，64 ビットアーキテクチャとベクトルアーキテクチャがある。特に 64 ビットアーキテクチャに Globus が対応していないと，大規模なソースコードの変更が必要になる。
- ライブラリ・システムコール: Globus のソース中で，特定のシステムに依存したライブラリ関数やシステムコールの呼び出しを行なっている場合，SX に合わせてソースを書き換える必要が生じる。

これらの点について調査したところ，大規模な変更を要する問題点は発見されなかったが，いくつか小さな変更を要することが分かった。

また，実際にコンパイルと実行を試みる段階で，事前には予想されなかった問題点も発見された。これは，Globus のソース中のプロタイプ宣言の不足から来るものである。

以降の節では，これらの対応作業の詳細を述べる。

4.1 configure スクリプト

Globus のソースファイルに含まれる configure スクリプトは，アーキテクチャや OS，個々のシステムの構成から，適切なコンパイル／リンクオプションを設定するものである。configure スクリプトで想定されていないシステムでは，正しく環境設定を行なうことはできない。

調査の結果，Globus 1.1.3 のソースに含まれている configure スクリプトは SX に対応していなかった。アーキテクチャの判別は configure 本体から呼び出される config.guess, config.sub スクリプトで行なっており，この部分に SX を判別するコードを追加することで，configure スクリプトを SX に対応させることができた。この判別のコードはアーキテクチャ／ OS 名を取得する uname コマンドの出力結果を元に条件分岐しているだけなので，数行のコードを追加するだけで済んでいる。

4.2 SX のアーキテクチャへの対応

SX の特徴の 1 つであるベクトルアーキテクチャについては，C/Fortran コンパイラによる自動ベクトル化機能があるため，特に対応の必要はない。また，64 ビットアーキテクチャについても，Globus は Alpha や MIPS10000 などの他の 64 ビットアーキテクチャにすでに対応しているため，本質的な問題はなかった。

ただし，整数演算の精度については注意する必要があった。SX における整数の除算には，浮動小数点演算命令を用いて実行する方法と，ソフトウェアで計算する方法がある。デフォルトでは前者の方法が使われるが，この方法では，除算の精度は 53 ビットしか保証されない。Globus のソースには，ハッシュ値の計算などで 64 ビットの精度を要求する箇所があり，除算をソフトウェアで実行するようにコンパイルオプションを指定する必要がある。

4.3 ライブラリ／システムコール

SUPER-UX の標準の C コンパイラは ANSI C 言語標準規格を満たしているため，Globus で使われている関数のほとんどについては変更を必要としなかった。ただし，通信を含む I/O 関連のコードについては，一部に特定のシステムを前提とした記述があり，そのままでは SX 上で動かすことはできなかった。

Globus に含まれていた主なシステム依存の記述は，ブロッキング／ノンブロッキング I/O の切り替えである。Globus では，プロセス間の通信にノンブロッキング I/O を用いているため，ブロッキング I/O からノンブロッキング I/O への切り替えを行なう手段が必要である。元の Globus のソースでは，ノンブロッキング

I/O への切り替えに fcntl システムコールを使っていたが，SX で動かすには，この部分を ioctl に変更する必要があった。

4.4 プロトタイプの不足

Globus 1.1.3 のソースには，strcat などの標準ライブラリ関数を適切なヘッダファイルをインクルードせずに使っている箇所がいくつか含まれていた。このような関数はプロトタイプがないので，返り値の型は int となる。SX のような 64 ビットアーキテクチャでは，long 型やポインタは int よりもビット数が多いため，返り値が本来の値と異なる可能性がある。

調査の結果，SX 上で Globus を実行する場合，プロタイプのない関数から異常な返り値を受け取ることによる障害が発生することが判明した。このため，不足していた#include 文をソースファイルに追加する作業を行なった。

5. 性 能 評 価

現在の SX 上での Globus の性能評価として，MPICH-G (Globus 上での MPI の実装) を使ったノード内通信のベンチマークテストを行なった。比較対象として，SX 上でのネイティブな MPI ライブラリである MPI/SX を用いた。

MPI/SX と MPICH-G の実装上の違いとして，前者はプロセス間通信に共有メモリを使っているのに対して，後者はループバックインタフェースによる TCP/IP 通信を使っていることが挙げられる。さらに，MPICH-G では，GSI (2.1 節参照) による認証などのオーバーヘッドが加わる。両者でどの程度の差が出るのかを測定することにより，SX 上での Globus の有効な使い方を判断したり，今後の最適化を行なう際の基準となることが期待できる。

5.1 ベンチマークテストの内容

性能評価に使った実験環境は，図 2(A) に示したものである。具体的には，以下の条件で実験を行なった。

- SX-4 の単一ノードのみを使用する。参考のために，MPI/SX については SX-5(1 ノード) を使用した測定も行なった。
- SX 上のジョブキューイングシステム (NQS) は使わず，対話的シェルのみを使用する。Globus の job-manager には fork job-manager を使用する。

ベンチマークテストとして，以下の 2 種類のプログラムを使用した。

- ping-pong: レイテンシを測定するためのプログラムで，2 つのプロセス間で int 型のデータを交互に送り，平均のメッセージ到達時間を計算する。

実行方法	平均レイテンシ [μsec]
MPICH-G(SX-4)	4189.05
MPI/SX(SX-4)	19.35
MPI/SX(SX-5)	5.35

表 1 ping-pong の測定結果

実行方法	スループット [KB/sec]	
	size=1KB	size=1MB
MPICH-G(SX-4)	2.304×10^2	1.155×10^3
MPI/SX(SX-4)	6.437×10^4	2.254×10^6
MPI/SX(SX-5)	1.933×10^5	6.475×10^6

表 2 throughput の実行結果

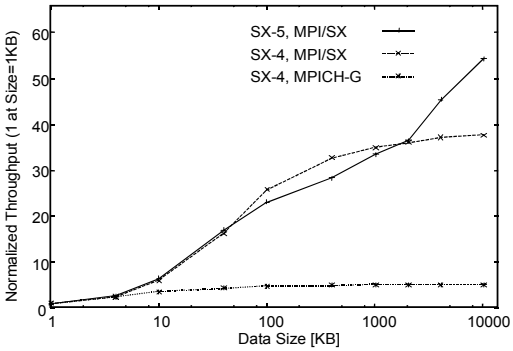

図 3 データサイズとスループットの関係

- throughput: 2 つのプロセス間で MPI_Send を使ってデータを交互に送り，所要時間を測定する。一度に送るデータサイズを 1KB から 10MB の範囲で変えて測定を行なう。

5.2 測 定 結 果

ベンチマークの実行結果を**表 1**，**表 2**，および**図 3** に示す。

表 1 は，ping-pong を実行した時の平均の応答時間を示している。この結果は，MPICH-G を使って同一ノード内でのメッセージ送信を行なう場合のレイテンシは MPI/SX を使う場合の 200 倍程度であることを意味している。今後，TCP/IP と Globus のプロトコルの処理がそれぞれどの程度の割合で影響を与えているかについて調査する予定である。

throughput については，一度に送るデータサイズが 1KB および 1MB の場合の実行結果を**表 2** に示す。データサイズが 1KB の場合，MPICH-G では MPI/SX よりも 280 倍程度スループットが小さいという結果で，ping-pong と近い性能比になっている。一方，1MB の場合には 2000 倍程度と，データサイズが大きい方が性能比が大きくなる傾向があった。

そこで，データサイズが 1KB の時のスループットを 1 とした場合のデータサイズとスループットの関係を**図 3** に示す。MPI/SX と MPICH-G のどちらも，一度に送るデータサイズが大きいほどスループットが大きくなっているものの，MPICH-G の方が上昇の幅が小さい。

この結果から，TCP/IP の処理または Globus I/O プロトコル内にボトルネックとなっている部分があることが予想される。今後，詳細な調査により原因を究明する予定である。

5.3 考　　察

前節までの実験結果から，現在の移植版では，ノード内の通信に Globus を使う場合には実用的な性能が得られないことが分かった。そこで，ノード内の通信には MPI/SX，他のノードとの通信には TCP/IP を使った通信を行なうのが現実的である。

現在，新しい Globus 上の MPI の実装である MPICH-G2 が公開されている。MPICH-G2 は，MPICH-G に対して性能の改善が図られているとともに，MPI/SX のようなベンダー提供の MPI ライブラリと Globus を組み合わせることが可能になっている。しかし，MPICH-G2 と MPI/SX で，内部で使われているシンボル名が衝突するという問題があるため，現在の MPICH-G2 では MPI/SX と組み合わせて使用することはできない。これは MPI/SX が MPICH をベースにしていることに起因する問題であり，他の MPICH ベースのライブラリでも同様の問題が発生する。今後のバージョンの MPICH-G2 でこの問題が解決されることになっており，それに伴って MPI/SX との組み合わせを試みる予定である。

6. まとめと今後の予定

本稿では，NEC と産総研の共同で行なった，Grid ミドルウェア Globus の SX シリーズへの移植作業と，その性能評価の結果を報告した。

移植作業として，システム依存の関数の呼び出しの置き換えや SX 上で発現するバグの修正を行なった。その結果，fork job-manager を使った単一プロセッサおよび複数プロセッサを使ったジョブの実行と，MPICH-G を使った並列プログラムの実行に成功した。SX-4 の単一ノード上で行なった MPICH-G の性能評価では，ネイティブな MPI の実装である MPI/SX に比べて 2 桁以上遅いという結果になり，ノード内での通信には MPI/SX を使う方法が有望であることが明らかになった。

今後，SX 上の Globus を実用に耐えうるものにするため，Globus の各コンポーネントと関連ソフトウェアの移植作業を進める予定である。

Globus のコンポーネントとしては，SX 標準の NQS バッチスケジューラを使った遠隔ジョブ実行への対応を行なう。このためには，今までの動作確認では使っていない shell job-manager (2.2 節参照) を使用し，NQS を通してジョブの投入や制御を行なうシェルスクリプトを作成する必要がある。

Globus に関連したソフトウェアとして，MPICH-G2 と Nexus Proxy[3] を SX に対応させる予定である。5.3 で触れたように，MPICH-G2 はベンダー提供の MPI ライブラリと Globus I/O との組み合わせが可能な設計になっており，ノード内での通信に MPI/SX を使うのに適している。現在の MPICH-G2 のバージョンではまだ MPI/SX との組み合わせには問題があるため，MPICH-G2 のバージョンアップを待ってから動作確認を行なう予定である。

Nexus Proxy は，ファイアウォールの内部と外部のホスト間で Globus I/O による通信を中継するソフトウェアである。日本国内の多くの計算機センターはファイアウォールの中にあり，その中の計算機が外部のホストと通信するためには，通信を中継するプロキシの機構が必要である。今後，SX 上での Nexus Proxy の動作検証と，必要に応じて SX への対応作業を行なう予定である。

謝辞　Globus の SX への移植に当たり，中村貴之氏をはじめ作業・実験環境をご提供頂いたつくばギガビットラボの皆様，ご助言を頂いた SRA の平野基孝氏に感謝致します。

参　考　文　献

1) I. Foster and C. Kesselman (editors). The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, 1998.
2) I. Foster, C. Kesselman. The Globus Project: A Status Report. *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pp. 4-18, 1998. Available at http://www.globus.org/research/papers.html
3) 田中 良夫, 平野 基孝, 佐藤 三久, 中田 秀基, 関口 智嗣. Firewall に対応した Globus による広域クラスタシステムの構築と性能評価, 情報処理学会ハイパフォーマンスコンピューティング研究会, Vol.2000, No.57, pp.63-68, 2000.
4) Message Passing Interface (MPI) Standard. http://www.mcs.anl.gov/mpi/