Globus を用いた Grid 上での並列数値処理とその性能評価

武田恵史　西田晃[†]　小柳義夫[†]

近年, 広域に分散された世界中の情報資源を統一的に扱う Grid と呼ばれる技術が盛んに研究され, 開発が進められている. 本研究では, LAN 上の PC クラスタを Grid に見立て, グローバルコンピューティングのソフトウェアインフラストラクチャに必要とされる様々な要素技術を提供するツール群である Globus toolkit をクラスタ上に実装した. 実験では Globus を用いて実装された MPICH の拡張である MPICH-G2 を使用して Linpack ベンチマークテストを行い, Grid 上での並列計算における性能を, Globus 上に実装され資源管理機能を備えた MPICH-G2 と通常の MPICH の複数のネットワーク上での比較により評価した. 実験から, 十分な通信帯域幅のある環境では, 両者はほぼ同等な性能を示すことが分かった.

Parallel Numerical Computing on the Grid using the Globus Toolkit and its Performance Evaluation

Yasushi Takeda , Akira Nishida[†] and Yoshio Oyanagi[†]

In recent years, the Grid technology that uses collections of geographically distributed information resources in the world is actively studied and developed. In this study, we have regarded a PC cluster on a local area network as a Grid, and implemented the Globus Toolkit on the cluster. The Globus provides various elements of technology for software infrastructure on global computing. In a set of experiments, we have used MPICH-G2, an extended MPICH implementation of MPI on services provided by the Globus, and evaluated the performance of the Grid technology by comparing MPICH with MPICH-G2, which implements functions for resource management for parallel computing on the Globus, with the Linpack benchmark tests on two networks. The two implementations have shown almost the same performance on the higher speed network.

1. はじめに

近年, 広域に分散された世界中の情報資源を統一的に扱う Grid と呼ばれる技術が盛んに研究され, 開発が進められている. Grid 技術は計算機を使用するための新しい概念であり, 電力網を示す “Grid” という言葉に表わされるように, 増大する計算機資源に, 統一的に, また確実かつ安価にアクセスできるようにすることを目標としている. グローバルコンピューティングは, これらの世界中に分散した異機種計算機環境を仮想的な一つの高性能計算機と見立て, 分散並列計算を行うための技術である. インターネットなどのネットワーク上に接続されている計算機は, その資源を常に完全に利用しているわけではない. グローバルコンピューティングの概念は, 分散した遊休状態の計算機資源を利用することで, 計算に必要とされる時間, コストを削減することを目的として生まれたものである.

グローバルコンピューティングに関しては, 現在様々なプロジェクトが進められているが, このうち最も有名なものとして, Globus プロジェクトを挙げることができる. Globus プロジェクトでは, グローバルコンピューティングを実現するために必要とされる Globus Toolkit と呼ばれるソフトウェアツールを開発しており, Grid 上のサービスを提供するための標準的なミドルウェアとなりつつある. この他にも Ninf や Netsolve など, 多くのプロジェクトで研究が進められている. 本研究では, Globus 上に実装された MPICH の拡張である MPICH-G2 を使用して, Grid 上での並列計算における性能を, MPICH との比較により評価した.

2. Grid 技術

2.1 Globus Toolkit

Globus Toolkit は, セキュリティ, 資源配置, 資源管理, 通信などの基本的な Grid サービスを実装した一連のコンポーネントにより構成されている. ここでは

† 東京大学大学院情報理工学系研究科コンピュータ科学専攻
Department of Computer Science, the University of Tokyo

Globus Toolkit の概要について述べるとともに, 主なサービスについて説明する.

広範囲のアプリケーションやプログラミングモデルをサポートするため, Globus Toolkit では, 開発アプリケーションのために効果的なサービスを選択可能な “bag of services” モデルに基づいて作成されている. bag of services アプローチでは, Globus Toolkit と他の技術を組み合わせることにより, Grid の高度な能力を活用するアプリケーションや, 独自の Grid 基盤の作成を容易にする. Globus Toolkit が提供するこのような基本的なサービスを使うことにより, 次節で述べる MPICH-G のような, よりレベルの高いツールを開発することができる. 表 1 に Globus の主要なサービスを示す.

Metacomputing Directory Service (MDS)

Grid では不均質な分散環境のもとで資源の状態が動的に変化するため, toolkit コンポーネント, プログラミングツール, アプリケーションのそれぞれが, システム構造と状態の変化に応じてその挙動を適応させる必要がある. Globus MDS は, Globus Grid 基盤の状態に関する統合された情報サービスを提供することにより, このような適応を可能にするよう設計されている. MDS により, アーキテクチャタイプ, オペレーティングシステム, メモリ, ネットワーク帯域幅と通信遅延, 利用可能な通信プロトコル, IP アドレスとネットワーク技術のマッピングなどの情報が提供される. MDS は, 具体的には Grid の構造と状態に関する情報を発見, 公開し, またこれらにアクセスするためのツール及び API 群である. MDS サービスは Lightweight Directory Access Protocol (LDAP) に基づいており, Grid コンポーネントに関する情報について, 統一的で拡張性のある表現を提供する.

本研究で使用した Globus 1.1.4 では, MDS として Grid Resource Information Service (GRIS), Grid Index Information Service (GIIS) の 2 種類のサービスが実装されている. GRIS は, 特定の資源に関する問い合わせを, グリッド資源上の Globus サービスの一部として実装されている情報プロバイダに転送することにより応答するための分散情報サービスである. GRIS サービスでは, ホストの一意性や CPU の利用状況などの情報を提供する. GRIS は, Globus Toolkit がインストールされるホストにはすべて実装される. 一方, GIIS は, Globus グリッドを構成するすべての資源に関する情報を提供する中央集中型の MDS サーバである. デフォルトではこのサービスは設定されないが, サイト内の GIIS 向けにホストとポートに関する情報を定義することができる.

Grid Security Infrastructure (GSI)

GSI は公開鍵暗号に基づいて認証を提供する. ユーザは, Globus Grid に認証された時点ですべての資源に対して管理者権限を付与され, これらの資源を利用できるようになる. この認証システムでは, Secure Sockets Layer (SSL) の実装に基づいた General Security Service API を使用している. 暗号化には RSA アルゴリズムを採用しており, 公開鍵, 秘密鍵の双方が利用される.

Generic Security Service API では, X.509 認証プロセスが利用される. ユーザは, X.509 証明書をホームディレクトリに置くことにより, システム上で認証を行うことができる. X.509 証明書には, 許可の有効期間, RSA 公開鍵, 及び認証機関 (Certificate Authority, CA) の署名に関する情報が含まれており, CA のみが証明書を発行することができる. Globus Grid の管理者は, Globus マップファイルと呼ばれるエントリマップを保持する. エントリマップは Globus の資源名をローカルなユーザ名にマップする.

```
“/O=Grid/O=Globus/OU=is.s.u-tokyo.ac.jp
/CN=TAKEDA Yasushi” ytakeda
“/O=Grid/O=Globus/OU=is.s.u-tokyo.ac.jp
/CN=familyname firstname” foo
⋮
```

表 2 Globus マップファイルの例

Globus Resource Allocation Maneger (GRAM)

GRAM は, 資源割当, プロセス生成, 状態監視, 及び管理サービスを提供する. GRAM に対するサービス要求は, ジョブ及びジョブ実行に必要な資源を記述するための言語である Resource Specification Language (RSL) によって行われ, ローカルなスケジューラと計算機に対するコマンドにマップされる. GRAM へのユーザインタフェースは, gatekeeper と呼ばれるデーモンプロセスである. GRAM に対するサービス要求に対して, gatekeeper は GSI 経由でクライアントとの相互認証を行い, jobmanager を起動して要求されたサービスを実行する.

2.2 MPICH-G2

MPICH-G2 は, Globus サービス上に MPI を実装した MPICH の拡張である. これにより, MPI アプリケーションを異なったアーキテクチャ上で動作させることが可能となる. MPICH-G2 は, 異機種の計算機の間で交わされるメッセージ中のデータを変換し, マシン間のメッセージ交換には TCP を, またマシン内のメッセージ交換にはベンダ提供の MPI を自動的に選択することにより, マルチプロトコル通信をサポートする.

MPICH-G2 による実行プロセスは以下の通りである. まず, `mpirun` コマンドがアプリケーションを起動するために使用される. アプリケーションを起動するには, 1) RSL スクリプトを記述し, `mpirun` に直接それを渡す方法と, 2) MPICH 実行時と同様の引数を用いて `mpirun` を使用する方法の 2 種類がある. 後者の場合, RSL スクリプトは自動的に生成される. いずれの

サービス	名称	機能
資源管理	GRAM	資源割当とプロセス管理
通信	Nexus	unicast, multicast 通信サービス
情報	MDS	Grid の構成, 状態に関する情報への分散アクセス
セキュリティ	GSI	認証などのセキュリティサービス
状態管理	HBM	システムの状態管理
リモートデータアクセス	GASS	インターフェイス経由でのリモートアクセス
実行管理	GEM	実行ファイルの構築, キャッシング及び配置

表 1 Globus の主なサービス

場合も, RSL スクリプトを `globusrun` に渡すことにより MPICH-G2 ジョブが起動される. MPI_Init は, Dynamically-Updated Request Online Coallocator (DUROC) を用いたバリアにより, すべてのマシンでプロセスがロードされ, 実行を開始するまで待機する. 通信ライブラリには Nexus が利用される.

3. 性能評価

本研究では, Globus Toolkit 及び MPICH-G2 をローカルネットワーク上の PC クラスタに実装し, 並列版 Linpack ベンチマークライブラリである HPL を用いて MPICH, MPICH-G2 の性能比較を行った.

3.1 実行環境

実験で用いたクラスタシステムのハードウェアアーキテクチャとソフトウェア構成は次の通りである.

- ハードウェアアーキテクチャ
  - CPU : 4× dual Intel Xeon Processor 2GHz
  - 主記憶 : 4 × 1GB (256MB × 4) PC800 RDRAM (Rambus)
  - i860 chipset
  - 3Com 3C920 (10/100BASE-T)
  - Broadcom Gigabit Ethernet (1000BASE-T)
- ソフトウェア構成
  - OS : Red Hat Linux 7.1.2 (Kernel 2.4.7)
  - C コンパイラ : gcc version 2.96
  - MPI : MPICH 1.2.2.3 (MPICH-G2 を含む)
  - BLAS : ATLAS (Version 3.2.1)
  - Globus Toolkit Version 1.1.4

BLAS (Basic Linear Algebra Subprograms) は, ベクトル, 行列演算のための高性能な “building block” ルーチンである. Level 1 BLAS ではベクトル - ベクトル演算, Level 2 では行列 - ベクトル演算, また Level 3 では行列 - 行列演算が実装される. ATLAS (Automaticallly Tuned Linear Algebra Software) は, 複数のメモリ階層とパイプライン化された演算ユニットを持つプロセッサ向けに, 最適化された BLAS の C, Fortran 77 インタフェースを生成する. 実験では, gcc でコンパイルした ATLAS を HPL で用いた.

3.2 予備実験

まず, ネットワーク上でのハードウェアの性能を調べるため, 性能評価ツール mpptest により MPICH と MPICH-G2 のネットワーク性能を評価した.

図 1, 図 2 に, 100BASE-T 及び 10000BASE-T ネットワーク上でのノンブロッキング通信性能を示す. 100BASE-T 上では通信速度が遅いため, Nexus の利用による通信延遅の影響は小さいが, 1000BASE-T ネットワーク上では, MPICH-G2 の性能が MPICH と比較して低下していることが分かる.

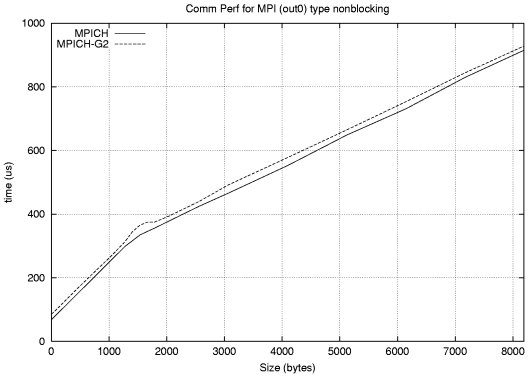

図 1 100BASE-T ネットワーク上でのノンブロッキング通信性能

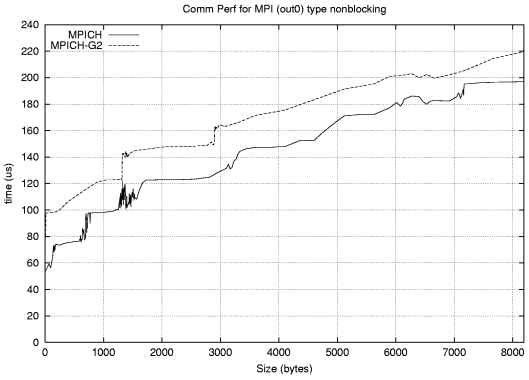

図 2 1000BASE-T ネットワーク上でのノンブロッキング通信性能

3.3 Linpack

Linpack は, BLAS を用いた直接法による連立一次方程式計算のライブラリであり, 性能評価のためのテストベンチマークとして最も広く用いられている. HPL は, MPI で書かれた分散メモリ型並列計算機向けのポータブルな Linpack ベンチマーク実装であり, 分散メモリ型計算機上で n 次元の密な倍精度乱数係数線形系

$$Ax = b \tag{1}$$

を解く.

Linpack では, サイズ $n \times (n+1)$ の係数行列 $[Ab] = [[L,U]y]$ に対して行部分枢軸選択を用いた LU 分解を行うことにより, 線形系を解く. 分解の過程で L が b に適用されるので, 解 x は上三角系

$$Ux = y \tag{2}$$

を後退法で解くことにより求められる.

並列計算機上でのデータのブロードキャストには多くの手法が考えられるが, HPL では 6 つのアルゴリズムが用意されている. ここでは, MPICH, MPICH-G2 上でそれぞれ最大性能が得られた increasing-ring アルゴリズム, long アルゴリズムについて説明する.

- increasing-ring
  最も古典的なアルゴリズムで, プロセスは次のプロセスに順次メッセージを送る.
- long
  このアルゴリズムでは, 実行に用いられる全てのプロセスの間で同期を取る. PxQ 個の部分行列として分割されたプロセスにおいて, メッセージは Q 個の小片に分割され, それぞれ Q 個のプロセスに送られる. 分割されたメッセージは, 隣接するプロセス同士の通信により, Q-1 ステップを経てすべてのプロセスに送られる. 小片を分割送信することで, 計算の待ち時間を減らす.

3.4 評価結果

ネットワークデバイスを明示的に指定する場合, MPICH では P4 procgroup ファイルを, また MPICH-G2 では Globus RSL ファイルを作成する必要がある.

P4 procgroup ファイルでは, プロセッサ数, 実行ファイル, 及び使用するネットワークデバイスを指定することができる. 図 3 に p4procgroup ファイルの記述例を示す.

```
# example of 4 processes on 1000Base-T
192.168.0.1 0 /home/ytakeda/hpl/bin/xhpl
192.168.0.2 1 /home/ytakeda/hpl/bin/xhpl
192.168.0.3 1 /home/ytakeda/hpl/bin/xhpl
192.168.0.4 1 /home/ytakeda/hpl/bin/xhpl
```

図 3 MPICH-G2 上で 1000BASE-T ネットワークを使用する場合の P4 procgroup ファイル記述例

Globus RSL ファイルでは, TCP バッファサイズを変更することが可能である. 1000BASE-T ネットワークを利用してプロセスを起動する場合の RSL ファイルの記述例を図 4 に示す.

以降の実験では通信オーバーヘッドを抑えるため, TCP バッファサイズを 256KB に指定したが, 実際にはデフォルト値の 32KB と比較して大きな性能の改善は見られなかった. 表 3 に問題サイズ 10000, プロセス数 8 で HPL を実行した場合の TCP バッファサイズと性能との関係を示す.

HPL では, 入力データファイル HPL.dat によってパラメータを変更できるようになっている. 入力ファイルには, 問題の大きさ, マシン構成, アルゴリズムの詳細についての情報を記述することができ, 実行形式のプログラムにより使用される, 図 9, 10 に, 問題サイズ N=10000, プロセス数 np=8 の場合に MPICH, MPICH-G2 上で最大性能を記録した際の HPL.dat の設定値を示す. 表 4 は np 値に対応する P, Q の値である[☆].

buffer size	Performance
4KB	7.191Gflops
8KB	8.647Gflops
16KB	9.152Gflops
32KB	9.276Gflops
64KB	9.257Gflops
128KB	9.295Gflops
256KB	9.303Gflops

表 3 TCP バッファサイズと性能との関係 (N=10000, np=8 の場合)

np	1	2	3	4	5	6	7	8
PxQ	1x1	1x2	1x3	2x2	1x5	2x3	1x7	2x4

表 4 1 から 8 までの np 値に対応する P, Q の値

100BASE-T ネットワーク上での性能比較

図 5 に, 1 から 8 まで np 値を変えた場合の MPICH, MPICH-G2 上での性能比較結果を示す. この結果から, もっとも効率的なプロセッサ数はネットワーク性能によって決定されると推測される. 図 6 は, 2000 から 20000 まで問題サイズ N を変えた場合の MPICH と MPICH-G2 上での比較結果を示す. 100BASE-T 上では, MPICH-G2 は MPICH と比較して十分な性能が得られなかった.

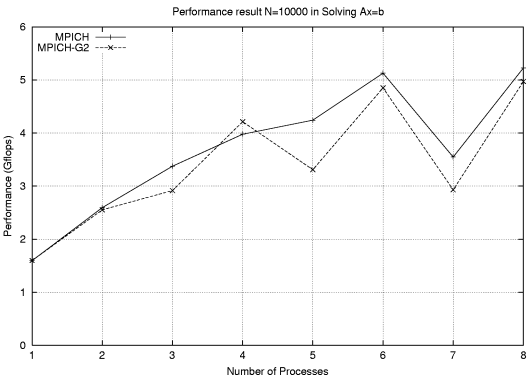

図 5 N=10000 の場合の 100BASE-T ネットワーク上での性能

1000BASE-T ネットワーク上での性能比較

ここで, 1CPUx2 ノード, 2CPUx1 ノードで実行した場合について, 1CPUx1 ノードを基準とした性能向上率

☆ パラメータと実行されるアルゴリズムの詳細については[6] を参照されたい.

```
+
( &(resourceManagerContact="out0.is.s.u-tokyo.ac.jp")
      (count=1)
      (label="subjob 0")
      (environment=(GLOBUS_DUROC_SUBJOB_INDEX 0)
                   (MPICH_GLOBUS2_TCP_BUFFER_SIZE 262144)
                   (MPICH_GLOBUS2_USE_NETWORK_INTERFACE 192.168.0.1))
      (directory="/home/ytakeda/hpl/bin")
      (executable="/home/ytakeda/hpl/bin/xhpl")
)
( &(resourceManagerContact="out1.is.s.u-tokyo.ac.jp")
      (count=1)
      (label="subjob 1")
      (environment=(GLOBUS_DUROC_SUBJOB_INDEX 1)
                   (MPICH_GLOBUS2_TCP_BUFFER_SIZE 262144)
                   (MPICH_GLOBUS2_USE_NETWORK_INTERFACE 192.168.0.2))
      (directory="/home/ytakeda/hpl/bin")
      (executable="/home/ytakeda/hpl/bin/xhpl")
)
```

図 4　MPICH-G2 上で 1000Base-T ネットワークを利用する場合の Globus RSL ファイル記述例

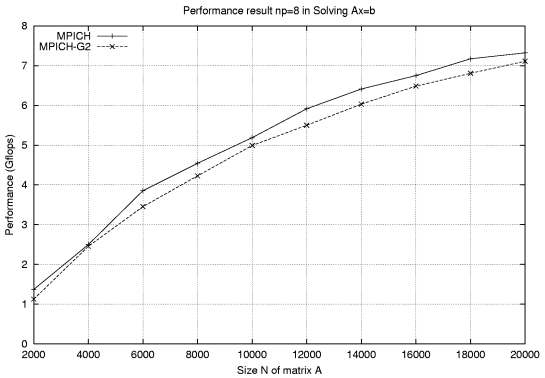

図 6　np=8 の場合の 100BASE-T ネットワーク上での性能

を表 5 に示す．この結果から, 同一のノード上でバスを共有することにより若干通信性能が低下するものの, 2 つのプロセッサが効率的に使われていることが分かる．

	MPICH	MPICH-G2
1CPU×1node	1.598Gflops(100)	1.601Gflops(100)
1CPU×2nodes	2.945Gflops(184)	2.934Gflops(183)
2CPUs×1node	2.843Gflops(178)	2.821Gflops(176)

表 5　1 ノード上で 2 プロセッサを用いた場合の性能向上

図 7 に, 1 から 8 まで np 値を変えた場合の MPICH と MPICH-G2 上での性能比較結果を示す．図 8 は, 2000 から 20000 まで問題サイズ N を変えた場合の比較結果である．これらの結果から, N が大きい場合には, アルゴリズムを調整することによって MPICH と MPICH-G2 でほぼ同等な性能が得られることが分かる．予備実験では通信遅延によって 1000BASE-T ネットワーク上での MPICH-G2 の性能が低下していることから, N のサイズが小さい場合に MPICH-G2 の性能が MPICH より低下している原因は, Nexus による通信遅延にあると考えられる．すなわち, このようにネットワークの帯域幅が十分に大きい場合には, 通信回数を抑えたアルゴリズムを採用することにより, MPICH-G2 上でも MPICH の場合と同等の性能を実現できるものと考えられる．

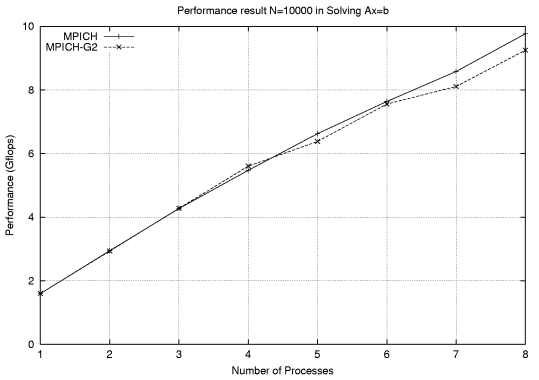

図 7　N=10000 の場合の 1000BASE-T ネットワーク上での性能

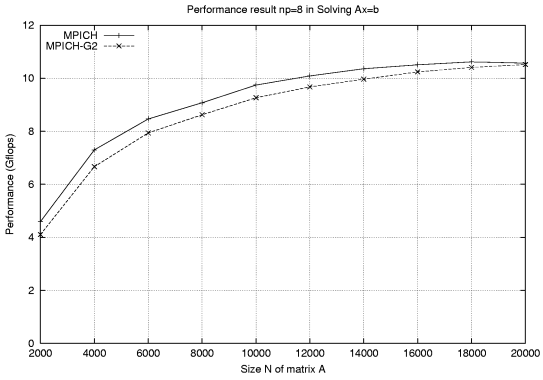

図 8　np=8 の場合の 1000BASE-T ネットワーク上での性能

4. 結　　論

本稿では, Grid サービスを提供するツールキットで

ある Globus を用いて, クラスタ上に Globus Tookit 及び Globus 向け MPICH 拡張である MPICH-G2 を実装することにより, Grid 上での並列数値処理の可能性について検証した.

性能性能ツール mpptest により, 通信性能をテストした結果, 100BASE-T ネットワーク上での性能と比較して, 1000BASE-T ネットワーク上では, メッセージサイズが小さい場合に MPICH に対する MPICH-G2 の性能低下がより顕著であることが分かった. また, Linpack ベンチマークの並列実装である HPL を用いて MPICH と MPICH-G2 の性能を比較した結果, 1000BASE-T ネットワーク上では最適なパラメータを選ぶことにより, MPICH と MPICH-G2 の性能がほぼ同等となることが分かった. これは, MPICH-G2 上で最適な long アルゴリズムでは MPICH の場合より通信回数が少なく, ネットワーク遅延の影響が小さいことによるものであるが, このことは, 十分な通信帯域幅のある環境では, 通信回数を抑えたアルゴリズムを採用することにより, ネットワーク遅延の影響を効果的に軽減できることを示している. 今回の実験は LAN 上で行ったものであり, 実際には WAN 上に分散した非均質な環境でのより現実的な評価が必要であるが, 広域環境上での科学技術計算の可能性を示唆しているものと思われる.

参 考 文 献

1) I. Foster and Carl Kesselman. *The GRID:Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers.,1998.
2) *The Globus Project*. http://www.globus.org/
3) *MPICH-G2*. http://www3.niu.edu/mpi/
4) William Gropp and Ewing Lusk. *User's Guide for mpich, a Portable Implementation of MPI Version 1.2.2*. Mathematics and Computer Science Division, Argonne National L aboratory.,1996
5) *Automatically Tuned Linear Algebra Software (ATLAS)*. http://math-atlas.sourceforge.net/
6) *HPL - A Portable Implementation of the High-Performance Linpa ck Benchmark for Distributed-Memory Computers*. http://www.netlib.org/benchmark/hpl/
7) *Globus Toolkit 1.1.3 System Administration Guide*. The Globus Project.,2000


```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name(if any)
6            device out(6=stdout,7=stderr,file)
1            # of problems sizes (N)
10000        Ns
1            # of NBs
72           NBs
1            # of process grids (P x Q)
2            Ps
4            Qs
16.0         threshold
1            # of panel fact
1            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
2            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact
2            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
0            BCASTs (0=1rg, 1=1rM, 2=2rg, 3=2rM, 4=Lng,
 5=LnM)
1            # of lookahead depth
1            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

図 9 MPICH 上での実行に用いた入力ファイル

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name(if any)
6            device out(6=stdout,7=stderr,file)
1            # of problems sizes (N)
10000        Ns
1            # of NBs
72           NBs
1            # of process grids (P x Q)
2            Ps
4            Qs
16.0         threshold
1            # of panel fact
1            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
2            NBMINS (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact
0            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
5            BCASTs (0=1rg, 1=1rM, 2=2rg, 3=2rM, 4=Lng,
 5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
2            SWAP (0=bin-exch,1=long,2=mix)
64           swapping threshold
0            L1 in (0=transposed,1=no-transposed) form
0            U in (0=transposed,1=no-transposed) form
1            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

図 10 MPICH-G2 上での実行に用いた入力ファイル