2006－HPC－107 (9)
2006／7／31

グリッドチャレンジテストベッドの構築と運用 ～グリチャレテストベッドの作り方～

合田憲人[†1] 大澤清[†1] 大角知孝[†1]
笠井武史[†1] 小野功[†1] 實本英之[†1]
松岡聡[†1] 斎藤秀雄[†2] 遠藤敏夫[†2,☆]
横山大作[†2] 田浦健次朗[†2] 近山隆[†2]
田中良夫[†3] 下坂久司[†4] 梶原広輝[†4]
廣安知之[†4] 藤澤克樹[†5,†3]

本稿では，2005 年 12 月から 2006 年 5 月にかけて実施された Grid Challenge in SACSIS2006 において使用されたグリッド実験環境の構築・運用事例を報告する．Grid Challenge は，大学，研究所が提供する複数の計算資源からなるグリッド実験環境上で，参加者がプログラミング技術を競う大会であり，今大会では 1200CPU 超の計算資源からなるグリッド実験環境が運用された．本稿では，実験環境ハードウェアおよびソフトウェアの仕様を紹介するとともに，ユーザ管理，ジョブ管理，障害対応といった運用事例についても報告する．

Construction and Operation of the Grid Challenge Testbed

Kento Aida,[†1] Kiyoshi Osawa,[†1] Tomotaka Osumi,[†1]
Takefumi Kasai,[†1] Isao Ono,[†1] Hideyuki Jitsumoto ,[†1]
Satoshi Matsuoka,[†1] Hideo Saito,[†2] Toshio Endo,[†2,☆]
Daisaku Yokoyama,[†2] Kenjiro Taura,[†2] Takashi Chikayama,[†2]
Yoshio Tanaka,[†3] Hisashi SHIMOSAKA,[†4]
Hiroki KAJIWARA,[†4] Tomoyuki Hiroyasu [†4]
and Katsuki Fujisawa[†5,†3]

This paper presents a case study to operate the Grid testbed for the Grid Challenge in SACSIS2006. The Grid Challenge is a programming competition on a Grid testbed, which is organized by multiple computing resources installed in universities and laboratories. In the last competition, the Grid testbed with more than 1200 CPUs was operated. The paper shows hardware/software specifications of the Grid testbed, and reports experience of the operation, which includes accounting, job management, and troubleshooting.

†1 東京工業大学
Tokyo Institute of Technology
†2 東京大学
The University of Tokyo
☆ 現在，東京工業大学
Presently with Tokyo Institute of Technology
†3 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
†4 同志社大学
Doshisha University
†5 東京電機大学
Tokyo Denki University

1. はじめに

グリッド研究の発展に伴い，実際にグリッドの実験環境（テストベッド）を構築し，ミドルウェアやアプリケーション技術の実証実験を行った結果が数多く報告されている．これらの実験環境には，グリッドアプリケーションの性能評価を行うことを目的として実験環境を構築する例[1)~4)]や，実験環境の構築と運用自体を目的とする例[5)]があげられるほか，プロダクショングリッドと呼ばれる実用的なグリッド環境の構築と運用を目的としたプロジェクト[6)]も進められている．しかし，このようなグリッド環境の構築および運用技

術はまだ確立されていないため，その構築および運用には高度な専門的知識と経験が必要であり，未だ非常に困難であるという問題がある．このような状況の中で，実際のグリッド実験環境の構築および運用事例に関する情報を蓄積することは，グリッド環境の構築・運用技術を確立させるために重要である．

本稿では，2005 年 12 月から 2006 年 5 月にかけて実施された Grid Challenge in SACSIS2006[7] において使用されたグリッド実験環境の構築・運用事例を報告する．Grid Challenge は，国内の大学，研究所が提供する複数の計算資源からなるグリッド実験環境上で，参加者がプログラミング技術を競う大会であり，今大会では 1200CPU 超の計算資源からなるグリッド実験環境が運用された．本稿では，実験環境ハードウェアおよびソフトウェアの仕様を紹介するとともに，ユーザ管理，ジョブ管理，障害対応といった運用事例についても報告する．

2. グリッドチャレンジの概要

Grid Challenge（通称「グリチャレ」）は，グリッド上のプログラミング技術を競う大会である．2005 年に第 1 回目が開催され，本稿で実験環境を紹介する Grid Challenge in SACSIS2006（以後「グリチャレ 2006」と表記する）は第 2 回目に相当する．グリチャレでは，国内の大学および研究所から提供される PC クラスタ群からグリッド実験環境（Grid Challenge Federated Clusters，以後「GCF」と表記する）が構築され，GCF 上で参加者がプログラムを開発し，その性能を競う．GCF の構築および運用は全て参加組織のボランティアにより進められ，グリチャレ 2006 では，産業技術総合研究所，東京工業大学，東京大学，同志社大学が GCF の構築および運用に参加した．

参加者は，規定課題部門と自由課題部門に参加できる．規定課題部門では，参加者が主催者の用意した問題を解くプログラムを開発し，その性能を競うことを目的としている．グリチャレ 2006 では，最適化問題の一つであるグラフ分割問題[8] が課題として与えられた．自由課題部門は，GCF 上で参加者が自由に実験を行うことを目的としており，参加者は予め提出した実験概要に基づいてプログラムを開発し，実験を進める．また規定課題部門では，GCF 上の計算資源を全参加者が共有する状況でプログラムを実行する予選期間と，予選を通過した参加者が計算資源を専有利用できる決勝期間に分かれて競技が行われる．

3. 実験環境

本節では，GCF を構成する計算資源に求められる要求要件，計算資源のハードウェアおよびソフトウェア構成について述べる．

3.1 要求要件

ユーザ（グリチャレ参加者）が GCF を利用できる期間は約 1ヶ月〜1ヶ月半である．短期間に GCF 上でユーザがプログラムの開発および実行を円滑に行うことを可能とするため，GCF を構成する PC クラスタには以下の機能が要求される．

(1) SSH やグリッドミドルウェア経由でジョブを遠隔起動できること．
グリッド実験環境上で実行されるプログラムの起動方法は，SSH 等のツールを使ってプロセスを遠隔起動する方法，またはグリッドミドルウェアを経由してプロセスを起動する方法が想定される．グリチャレでは，ユーザが様々な方法を用いてプログラム開発を行うことを可能とすることが望ましいため，GCF ではどちらの方法も利用可能とする．

(2) 計算機の状況に関するモニタリングツールがあること．
大規模グリッド実験環境では，運用中に一部の計算機等に障害が生じる可能性が高い．実験環境上の計算機の状況をモニタリングツールにより監視できることは，管理者が迅速に障害対応を行う上で必須である．またユーザは，モニタリングツールを利用することにより，プログラムの検証や負荷分散（より負荷の低い計算機上でジョブを起動する等）を効率よく行うことができる．

(3) グリッド上で並列プログラムを作成するためのプログラミングツールがあること．
C/C++等の逐次計算のための言語処理系に加えて，PC クラスタやグリッド上で実行可能な並列プログラムを作成するための処理系を用いてプログラムが開発されることが予想されるため，これらのプログラミングツールを用意しておく必要がある．

上記の要求要件は，全てソフトウェアに対する要件である．ハードウェアに対する要件は特に定めなかったが，グリチャレ 2006 では各サイトに設置される計算資源の CPU 数が 100CPU 程度以上となるように配慮した．理由は，3.2 節に示すように 200〜300CPU 以上の計算資源を提供するサイトが含まれることから，計算能力が著しく低いサイトが実際には利用されない可能性があるためである．

3.2 ハードウェア構成

GCF の計算資源は，**表 1** に示すように，6 サイトに分散された 7 台の PC クラスタから構成され，ノード総数は 606 ノード（1212CPU）に達する．各 PC クラスタは 1 台のゲートウェイノードと複数の計算ノードから構成され，ゲートウェイノードは，各 PC クラスタのログインノードとして利用されるとともに，プログラムのコンパイル等の作業にも用いられる．ゲー

表 1 グリッドチャレンジ実験環境

名称	サイト	ゲートウェイノード (CPU, memory, NIC, OS)	計算ノード (CPU, memory, NIC, OS)	計算ノード数 / CPU 数	IP アドレス
F32	産総研	Xeon 3.0GHz dual, 4GB, 1000BASE-T, Linux RedHat 8.0	同左	128/256	global
SAKURA	産総研	Opteron 2.2GHz dual, 2GB, 1000BASE-T, SUSE Linux Enterprise Server 8	Opteron 1.8GHz dual, 3GB, 1000BASE-T, SUSE Linux Enterprise Server 8	16/32	global
DIS	東工大 (すずかけ台)	Athlon MP 2000+ 1.6GHz dual, 512MB, 100BASE-TX, Vine 2.6	同左	50/100	private
PrestoIII	東工大 (大岡山)	Athlon MP 1900+ 1.6GHz dual, 768MB, 1000BASE-T, Debian Sarge 32bit 環境	Opteron 246/242 2.0/1.6GHz dual, 4/3/2GB, 1000BASE-T, Debian Sarge 32bit 環境	103/206	global
Tau	東大 (本郷)	Xeon 2.4GHz dual, 2GB, 1000BASE-T, Redhat Linux 7.3	Xeon 2.4/2.8GHz dual, 2GB, 1000BASE-T, Redhat Linux 7.3/Fedora Core Linux 3	175/350	global
Chikayama	東大 (柏)	Xeon 2.4GHz dual, 1GB, 1000BASE-T, Debian GNU/Linux	Xeon 2.4GHz dual, 2GB, 1000BASE-T, Debian GNU/Linux	64/128	global
Xenia	同志社大	Xeon 2.4GHz dual, 1GB, 100BASE-TX, Debian GNU/Linux 3.1 (Sarge)	同左	63/126	private

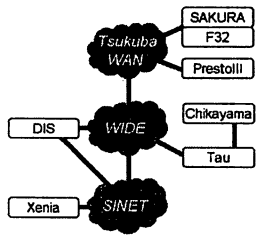

図 1 PC クラスタ間のネットワーク構成

トウェイノードと計算ノードは，ともに x86 系 CPU を搭載し，これらの OS は Linux である．計算ノードに割り当てられている IP アドレスは，表 1 に示すとおり，グローバルアドレス (global) を持つ PC クラスタとプライベートアドレス (private) を持つものが混在する．

図 1 は，PC クラスタ間のネットワークトポロジの概略を示す．産総研および東工大 (大岡山) に設置された PC クラスタ (F32, SAKURA, PrestoIII) への通信は，Tsukuba WAN[9] を経由して行われる☆．東工大 (すずかけ台) および東大に設置された PC クラスタ (DIS, Chikayama, Tau) への通信は，WIDE[10] を経由して行われ，また同志社大に設置された PC クラスタ (Xenia) への通信は SUPER SINET[11] を経由する．

☆ 東工大 (大岡山) に設置された PrestoIII は，東工大学内ネットワークには接続されておらず，専用回線により Tsukuba WAN に接続されている

3.3 ソフトウェア構成

3.1 節に示したソフトウェアの要件を満足するために，GCF を構成する PC クラスタ上には**表 2** に示すソフトウェアがインストールされている．各ソフトウェアの詳細は以下の通りである．

3.3.1 ジョブの遠隔起動

参加者の PC クラスタへの遠隔ログインや遠隔プロセス起動は，SSH を用いて行うこととし，RSH や TELNET 等のセキュリティ上問題のあるツールの利用は禁止されている．また，グリッド上の計算機に対するログインやジョブ起動は，対象とする計算機数が多い場合はユーザにとって大きな負荷となる．GCF では，各 PC クラスタ上に複数の計算機に対して SSH を用いたログインやプロセス起動を一括して行うためのツールである GXP[12] がインストールされている．

PC クラスタ上でジョブを実行する場合，ユーザは，SSH や GXP を用いて PC クラスタ上の計算ノード上でプロセスを起動すること以外に，バッチキューイングシステムを経由してジョブを起動することができる．GCF 上の PC クラスタのうち，PrestoIII および Xenia 上では PBS[13]，その他の PC クラスタ上では Sun Grid Engine (SGE)[14] がインストールされている．

PC クラスタ上には，グリッドミドルウェアとして Globus Tool Kit 2.4(GT2)[15] がインストールされ

表 2 PC クラスタ上のソフトウェア構成

遠隔ログイン・プロセス起動ツール	SSH, GXP
バッチキューイングシステム	Sun Grid Engine, PBS
グリッドミドルウェア	Globus Tool Kit 2.4
モニタリングツール	Ganglia
プログラミングツール	C/C++, Python, Perl, MPICH 1.2.7, Ninf-G 2.4

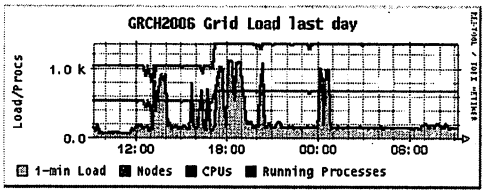

図 2 Ganglia による負荷情報

ており，主にユーザ認証やジョブ投入に用いられる☆．Globus を経由するジョブ起動では，PC クラスタのゲートウェイノード上で起動される gatekeeper プロセスを経由して，バッチキューイングシステムにジョブが投入される．

3.3.2 モニタリングツール

PC クラスタ上の CPU やメモリの負荷情報をユーザに提供するモニタリングツールとして，Ganglia[16] が各 PC クラスタ上にインストールされている．本ツールは，GCF を構成する PC クラスタ毎の負荷情報を提供するほか，GCF 全体の負荷情報も提供し，ユーザは Web 上でこれらの情報を閲覧できる．例えば，**図 2** は，グリチャレ実施期間中の GCF 全体の CPU 負荷情報の例であり，本図より 1000 個以上のプロセスが実行されている様子がわかる．

3.3.3 プログラミングツール

PC クラスタ上では，C/C++，Python，Perl 等のプログラミングツールのほか，単一 PC クラスタ上での並列プログラミング用に MPICH 1.2.7[17] がインストールされている．また，グリッド向けのプログラミングツールとしては，GridRPC[18] の参照実装である Ninf-G 2.4[19] がインストールされている．

4. 運　　用

本節では，GCF 上でのユーザ管理およびジョブ実行に関する運用方針について述べるとともに，運用中に発生した障害事例を報告する．

4.1 ユーザ管理

ユーザ管理は，グリッド実験環境の運用上最も重要な項目の一つである．GCF では，SSH および Globus によるジョブ起動が行われることから，以下の方針によりユーザ管理を行った．

PC クラスタ上でのローカルアカウントの管理はサイト毎に行うが，ユーザに対しては全 PC クラスタ上で同一のアカウント名を割り当てる．また SSH 実行時認証は公開鍵を用いた認証のみとし，パスワード認証は禁止する．

Globus によるジョブ起動では GSI によるユーザ認証が行われるため，Globus のユーザ証明書が必要となる．グリチャレ 2006 では，グリチャレ用の認証局（CA）が東工大（すずかけ台）において運用された．CA ソフトウェアには，Globus Tool Kit の simpleCA を用いた．ユーザ証明書発行に関する手続きは以下のとおりである☆☆．

(1) ユーザ証明書の発行を希望するユーザが，申請書を CA 管理者までメールにて送付する．
(2) CA 管理者が，ユーザのグリチャレ参加申込時の情報とユーザ証明書申請書の情報を照合し本人確認を行う．本人確認終了後，CA 管理者はユーザ証明書をメールにて送付する．
(3) ユーザ証明書を受け取ったユーザは，GCF の管理者メーリングリスト（各サイトの管理者が含まれる）にユーザ証明書を送付し，grid-mapfile への登録を依頼する．
(4) 各サイトの管理者が，ユーザアカウントの grid-mapfile への登録を行い，当該ユーザに連絡する．

4.2 ジョブ実行

PC クラスタ上でのジョブの実行方式は，インタラクティブ方式およびバッチ方式の 2 つの方法があげられる．インタラクティブ方式では，ユーザが使用する計算ノードを指定して SSH または GXP 経由でプロセスを起動する．一方バッチ方式では，ユーザがバッチキューイングシステムに対してジョブ実行を依頼し，バッチキューイングシステムが空いている（または負荷の低い）計算ノード上でユーザプロセスを起動する．

大学の計算機センター等で計算サービスを提供する実運用大規模計算機上では，ジョブ実行がバッチ方式に限られる場合は多く，実際のグリッド環境では利用可能なジョブ実行方式が混在する可能性がある．GCF では，利用可能なジョブ実行方式は各サイトの運用方針に従う．

グリチャレ規定課題部門では，決勝期間中にユーザが全計算資源を専有利用できる専有スロットが用意される．専有スロット期間は，割り当てられたユーザ以

☆ グリチャレ実施期間中には，最新バージョンとして既に GT4 が公開されていたが，各サイト管理者の多くが，GT2 の運用経験を有していたため，GT2 を採用した．

☆☆ 本手続きはグリチャレ用の実験的なものであり，必ずしもセキュリティ上頑健なものではない．

外のユーザによるプロセス起動は禁止される．しかし，他ユーザのプロセス起動を強制的に排除するためのソフトウェア上の仕組みは特に設けず，ユーザ間の紳士協定により専有スロットを実現している．

4.3 障　　害

迅速な障害対応を実現するためには，障害事例とその対応に関する情報の蓄積が重要である．以下では，GCF 運用中の主な障害とその対応について報告するとともに，障害対応に有効な技術について議論する．

(1) 計算ノード障害
一部の PC クラスタにおいて数台の計算ノードに障害が発生した．主な原因は，OS のハングアップ，ハードディスクの障害によるものであり，計算ノードの再起動や部品交換によって復旧された．大規模 PC クラスタの運用では，障害の所在を迅速に特定することは重要であり，グリチャレにおいても Ganglia のようなモニタリングツールが有用であった．

(2) 電源障害
一部の PC クラスタ上で 10 台前後の計算ノードが数日間停止した．原因は電源容量不足によってブレーカが遮断されたためであり，計算ノードの電源配分を見直すことにより復旧された．電源容量の確保は，初歩的な問題のように思えるが，大学の研究室等の小規模なグループで大規模 Beowulf 型 PC クラスタを運用する上では難しい問題である．今後，PC クラスタ上での電源管理ツールや PC 消費電力の低電力化技術の発展が期待される．

(3) サーバ障害
NFS サーバに障害が発生し，PC クラスタ全ノード上でユーザがログインできない障害が発生した．またバッチキューイングシステムに障害が発生し，ジョブの起動ができない障害が発生した．いずれの場合も，ソフトウェアの再起動等により解決された．大規模 PC クラスタ上でサーバプログラムを運用する場合，小規模 PC クラスタに比べてサーバプログラムへの負荷も大きいため，より安定した運用を実現するためには冗長化等の対策も必要であろう．

(4) Ganglia
実験環境の運用開始時に一部のサイトについて Ganglia によるモニタリング情報を収集できない障害が発生した．原因は，Ganglia が情報収集を行うために使用するポートがサイトのファイアウォールにより遮断されていたためであり，ファイアウォールの設定を行うことで解決された．また一部のサイトについて，Ganglia のデーモンプログラムが稼動するノードの負荷が非常に高くなり，著しく反応が遅くなる障害が発生した．Ganglia の設定において，当該サイトからの情報収集を行う間隔を少し長めにすることで一時的に対処できたが，その後も不安定な状態が発生することもあった．

(5) ユーザプロセスの暴走
分散計算環境上でプログラム開発と実行を繰り返す過程では，ユーザプロセスが暴走することも少なくない．グリッド上の複数の PC クラスタ上でプロセスが暴走した場合にこれら全てを停止する作業は煩雑であり，一部のプロセスが停止されずに残る可能性もある． GCF では，各 PC クラスタ上に GXP を用いて複数の（または全ての）計算ノード上のユーザジョブを停止するコマンドが用意された．また，ユーザが暴走に気づかずプロセスが長時間放置されると，他のユーザジョブの実行にも大きな影響をおよぼしかねない．この問題については，インタラクティブ環境上でユーザプロセスの監視を行い，異常を発見した場合にユーザや管理者に通知する機能が有効と考えられ，このような機能を持つ管理ツールの開発が期待される．

(6) 専有スロット
規定課題部門決勝のために設けられた専有スロット期間に，スロットを割り当てられたユーザ以外のユーザジョブが計算ノード上で実行される場合があり，ユーザまたは管理者が手動でジョブを停止する等の対処を行った．グリチャレの専有スロットはユーザ間の紳士協定により運用されていたが，より確実に専有スロットを確保するためには，計算ノードを予約して専有状態を保証する仕組みが必要である．単一 PC クラスタ上でバッチジョブを実行するために予約を実現する機能が一部のバッチキューイングシステム[13),14)] 上で実現されているが，グリッドのような複数の PC クラスタ間で連携して予約を実現する技術や，インタラクティブ環境上で予約を実現する技術については研究段階であり，今後，これらの技術の実用化が期待される．

上記のような障害発生時には，情報をユーザへ適切に通知することも重要である．GCF では，参加者向けホームページにより障害情報を通知したが，情報の掲載は以下の方針に従った．

(1) 少数の計算ノードの障害が発見された場合は，原則として数日のうちに当該サイトの管理者により復旧されることを見込んで，ユーザへの通知を行わない．ただし，ユーザは Ganglia によるモニタリング情報により，各計算ノードの状態について自ら調査することができる．

(2) 多数の計算ノードに障害が発見された場合や長期間にわたり計算ノードが使用できなくなる場合等，ユーザへの影響が大きい障害が発生した場合は，ユーザへ通知する．影響の大きい障害

の定義は以下の通りである.

- ハードウェアの障害等, 復旧がすぐには見込めない.
- サーバの障害やジョブの暴走等によりPCクラスタ全体または大部分が利用できない.
- メンテナンス等によりPCクラスタ全体または大部分が停止する.

5. 構築手順

GCFの構築には約3ヶ月間を費やし, その後1ヶ月半の間運用した. 本節では, GCF運用開始に至るまでの構築手順をまとめる.

(1) キックオフミーティング (2005年11月21日)
(2) 計算資源情報収集 (2005年12月)
(3) 管理者アカウント作成 (2006年1月上旬)
(4) ソフトウェアインストール開始 (2006年1月上旬)
(5) 管理者/参加者ホームページの運用開始 (2006年1月下旬)
(6) CA運用試験 (2006年2月上旬)
(7) ジョブ起動試験 (2006年2月上旬)
(8) Ganglia運用試験 (2006年2月上旬)
(9) ユーザアカウント作成 (2006年2月上旬)
(10) 試験運用開始 (2006年2月15日)
(11) 本運用開始 (2006年2月22日)
(12) 本運用終了 (2006年3月31日)

6. おわりに

本稿では, Grid Challenge in SACSIS2006において使用されたグリッド実験環境の構築および運用事例を報告した. グリッド実験環境の構築・運用技術を確立するためには, その技術やノウハウの蓄積が重要である. 本稿が報告する事例が, これからグリッド実験環境を構築しようとする研究者・技術者の益となれば幸いである.

謝辞 Grid Challenge in SACSIS2006の運営に貴重な時間を費やしていただいたGrid Challenge in SACSIS2006実行委員会の皆様に感謝いたします. また, 運営にご支援いただいたSACSIS2006実行委員会および情報処理学会コンピュータサイエンス領域委員会に感謝いたします.

参考文献

1) 武宮博, 田中良夫, 中田秀基, 関口智嗣. MPIとGridRPCを利用した大規模Gridアプリケーションの開発と実行:Hybrid QM/MDシミュレーション. **情報処理学会論文誌**, 46(SIG12 ACS11):384–395, 2005.
2) K.Aida and T.Osumi. A case study in running a parallel branch and bound application on the grid. In *Proc. IEEE/IPSJ 2005 Symposium on Applications & the Internet (SAINT2005)*, pages 64–173, 2005.
3) 小野功, 水口尚亮, 中島直敏, 小野典彦, 中田秀基, 松岡聡, 関口智嗣, 楯真一. Ninf-1/Ninf-Gを用いたNMR蛋白質立体構造決定のための遺伝アルゴリズムのグリッド化. **情報処理学会論文誌**, 46(SIG12 ACS11):396–406, 2005.
4) D. Abramson and et al. Deploying scientific applications to the PRAGMA grid testbed: Strategies and lessons. In *Proc. Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, pages 241–248, 2006.
5) ApGrid. http://www.apgrod.org/.
6) TeraGrid. http://www.teragrid.org/.
7) Grid Challenge in SACSIS2006. http://www.hpcc.jp/sacsis/2006/grid-challenge/.
8) D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Shevon. Optimization by simulated annealing: An experimental evaluation; part I, graph partitioning. *Operations Research*, 37(6):865–892, 1989.
9) つくばWAN. http://www.tsukuba-wan.ne.jp/.
10) WIDE Project. http://www.wide.ad.jp/.
11) SINET. http://www.sinet.ad.jp/.
12) K. Taura. Gxp : An interactive shell for the grid environment. In *International Workshop on Innovative Architecture for Future Generation High-Performance Processors and Systems*, 2004.
13) Altair PBS Professional. http://www.altair.com/software/pbspro.htm.
14) Sun N1 Grid Engine. http://www.sun.com/software/gridware/.
15) I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *Int. J. of Supercomputing Applications*, 11(2):115–128, 1997.
16) M.L. Massie, B.N. Chun, and D.E. Culler. The ganglia distributed monitoring system: Design, implementation, and experience. *Parallel Computing*, 30(7):817–840, 2004.
17) The Message Passing Interface (MPI) standard. http://www.hpclab.niu.edu/mpi/.
18) H. Nakada, S. Matsuoka, K. Seymour, J. Dongarra, C. Lee, and H. Casanova. A GridRPC model and API for end-user applications. *GGF Document*, GFD-R.052, 2005.
19) Y. Tanaka, H. Nakada, S. Sekiguchi, T. Suzumura, and S.Matsuoka. Ninf-G: A reference implementation of RPC-based programming middleware for grid computing. *J. of Grid Computing*, 1(1):41–51, 2003.