グリッド技術を用いた効率的並列処理による脳機能解析システム

甲斐島　武[†]　市川昊平[†]　高坂貴弘[†]
伊達　進[†]　水野（松本）由子[††]　下條真司[†††]

グリッド環境での効率的な協調並列型プログラムの実行に対する需要は高い．今日までにグリッド環境上での分散型プログラムを効率的に実行するグリッドスケジューリングシステムが開発されているが，協調並列型プログラムのスケジューリングにはいまだ数多くの課題が残されている．本研究では，協調並列型ジョブとして脳機能解析を取り上げ，グリッド環境上での効率的実行を可能とする計算サーバ自動選択システムを開発した．本システムでは，ユーザが計算サーバ選択ポリシを決定し，これらのポリシと各 CPU やネットワークの性能などの基本情報に基づいて，提案する評価式より，ジョブの計算サーバへの効率の良い割り当てを決定する．

Brain Function Analysis System using Efficient Parallel Processing on Grid Environment

Takeshi Kaishima,[†] Kohei Ichikawa,[†] Takahiro Kosaka,[†]
Susumu Date,[†] Yuko Mizuno-Matsumoto[††]
and Shinji Shimojo[†††]

A scheduling system that can efficiently complete the computation of a problem in the class of parallel and cooperative jobs is now demanded among scientists and researchers. However, many solutions for efficiently completing on the Grid mostly target a class of distributed jobs until today. In this paper, a scheduling system whose information-processing mechanism automatically considers dynamic status of computer resources and a flexible expression based on users' policy are proposed. And its efficiency is demonstrated by using a brain function analysis application which belongs to class of parallel and cooperative jobs.

1. 背景と目的

グリッドは広域ネットワーク上に分散した計算および情報リソースを活用して大規模計算を実現する広域分散計算技術であり，近年この技術を用いた実際科学への応用研究が世界各国で推進されている．大阪大学サイバーメディアセンターが中心となって推進しているバイオグリッドプロジェクト[1)] では，グリッドによって集約した計算資源を用いてより正確かつ高速に脳機能解析を行うことを目的の 1 つとしている．今日，多くの先進諸国は高齢化社会への移行が急速に進んでおり，今後アルツハイマー病などの痴呆症といった脳に起因する病気が増加することが予測されている．しかし，ヒトの脳は肺や腎臓といった他の器官に比較して機能的にも構造的にも非常に複雑であるため，現在の科学技術においてもその機能は明らかになっていないという現状がある．

脳機能解析は，脳波計（Electroencephalography: EEG）や脳磁計（Magnetoencephalography: MEG）から取得される 1 次元波形データから脳内の信号源を求めることにより，脳内に脳機能マップを構築することを目的として研究が進められている[2)]．このような，結果から原因を求めるという逆問題においては，膨大な計算空間を探索しなければならない．特に，脳機能解析においてはほぼ無限大の計算量が必要とされる．したがって，この解析計算をグリッドで効率よく分散することにより高速計算することは非常に有意義である．

また，今日までにグリッド環境上での計算を効率的かつ容易にするグリッドスケジューリングシステムが開発されている[3)4)]．しかし，分割したジョブ同士の同期や通信などの相互作用を必要とする協調並列型

† 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†† 大阪城南女子短期大学
Osaka Jonan Women's College
††† 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka University

ジョブの問題をグリッド環境上で効率的に実行するためには，いまだ数多くの課題が残されているのが現状である．

本研究では，グリッド環境上での協調並列型ジョブの演算高速化を目的としたスケジューリングシステムの実現を目的とした．協調並列型ジョブとして，特にMPI（Message Passing Interface）の API（Application Programming Interface）で記述されたものに主眼を置いた．その主な理由として，多くの協調型ジョブが MPI を用いて記述されており，グリッド環境上での利用に対する需要が高いことが挙げられるからである．本研究では，グリッドスケジューリングシステムの実現に必要不可欠である，グリッド環境の動的および静的な情報を利用する情報処理機構に着目して，計算サーバ自動選択システムを構築した．

2. 脳機能解析の協調並列型ジョブ実行の問題

協調並列型アプリケーションとして，本研究では脳機能解析手法の 1 つである独立成分分析（Independent Component Analysis: ICA）[5] をアプリケーション例として採用した．MEG は，脳の神経活動に伴って発生する微弱磁場を，頭皮上から超伝導量子干渉素子（Superconducting Quantum Interference Device: SQUID）センサを用いて非侵襲で計測するものであり，日常臨床への応用が期待されている高度先進医療機器である．しかし，頭皮上で計測される信号は，ノイズ信号や多数の部位で同時に発生する脳活動信号の混合信号であるため，各信号成分の種類と発生部位を同定する手法が必要である．ICA は，MEG の複数のセンサから計測される脳機能 1 次元波形データの統計的な独立性に着目し，ノイズや脳内から発生した多数の原信号に分離する手法であり，異常波だけを抽出し解析する，同時に発生している複数信号間の相互関係を調べるといった，動的で複雑な脳機能を解析することが可能である．信号を分離するだけでなく，さらに各分離信号の発生部位を推定することも可能で，例えば **図 1** に示す例では，てんかん特有の異常波（図左）がどの部位で強く発生しているかを図右が表している．この結果から右側頭葉付近に脳疾患の可能性があると推定できる．

本研究では，ICA を実装するアルゴリズムとして Joint Approximate Diagonalization of Eigenmatrices（JADE）[6] を用いた．本手法の特徴は特に安定性の点で非常に期待の高いものであるが，一方で非常に大きな計算時間が必要であるという問題がある．分離する信号数を n とすると，$O(n^6)$ の計算量にもなる．

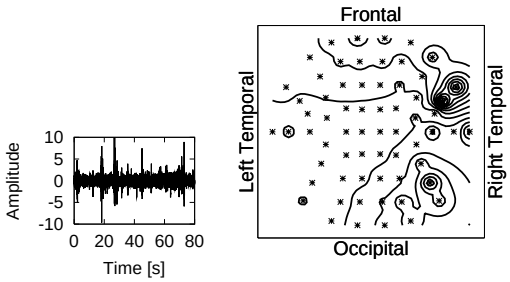

図 1 MEG データへの ICA の適用例
Fig. 1 An application example of ICA to MEG data

本研究ではセンサ数が 64 の MEG を対象としたが，MEG のセンサ数は増加を続け現在は 300 を越えるものまで実用化されている．JADE は 4 次のクロスキュムラントからなる行列を対角化する手法で，各行列の対角化がそれぞれ独立に計算できることから，並列化により計算時間の問題の解決を図った．

本研究で MPI を用いて実装した JADE を単純にグリッド環境上で実行した場合に発生する問題として，計算資源の使用率が非効率になるという問題がある．その例を **図 2** のガントチャートに示す．ただし，本研究では従来のクラスタ環境である同質な資源を仮定して作成されたプログラムを対象としているため，このプログラムの内部でスケジューリングは行っていない．つまり，各ワーカプロセスが同質の資源上で実行していると仮定して，4 次クロスキュムラント行列のデータを均等に分配すればほぼ同時に処理が終了することを想定している．JADE の計算においては，ワーカプロセスのすべての計算結果を 1 周期ごとにマスタプロセスに集約する操作が必要である．なぜなら，この集合演算により得られた結果が，各ワーカプロセスでの次の計算に必要なためである．しかし，この集約操作を効率よく行うためには，各プロセスがほぼ同じ時間で操作を終了する必要がある．グリッドにおいては，さまざまな性能の計算機が多様な負荷状況で稼働しているために，プロセスの実行時間をジョブ投入前に予測する事はほぼ不可能であると考えられる．

グリッドにおいては，そのような分散する計算機の CPU 性能などの静的な情報やシステム利用率などの時々刻々と変化する動的な情報を検索する仕組みが提案・実装されている．しかし，このようなメカニズムを通して提供される情報を実際に利用するのはグリッドに関する知識を持たないユーザであるのが一般的である．そのために，取得された情報の意味を解釈し最も効率良く協調並列型ジョブを実行することは非常に難しい．MPI 実行環境で実際にジョブを割り当てる

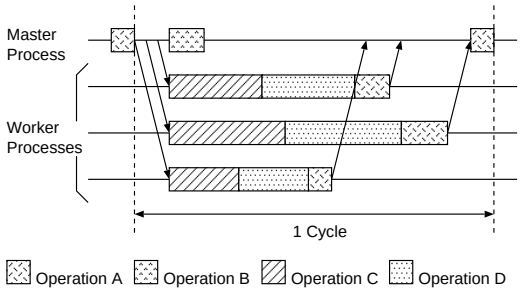

図 2 並列版 JADE のガントチャート
Fig. 2 Guntt chart of parallel JADE

際には，使用可能なホスト名を列挙したリストを参照して，そのリストの先頭から指定数の CPU を用いるという実装が行われている．言い換えると，何回ジョブの実行を行っても CPU の選択は毎回同じであり，CPU やネットワークの性能や負荷は考慮されない．異質性を考慮するためには，実行に先だってユーザが各資源の情報を調査，比較し，選定した後にリストを書き換える必要がある．しかしこの方法はユーザへの負担が大きく，また資源の数が増えると全ての資源の動的挙動を把握することができず効率的な資源の選択は不可能であり，現実的でない．

3. 提案する計算サーバ自動選択システム

まずフレームワークの設計に際し，ユーザは実行しようとするジョブの特性について知っているものと仮定する．もしユーザとデベロッパが異なる場合は，その特性を記述したファイルを，デベロッパがソースプログラムか実行プログラムとともに配布しているものとする．なぜなら，ユーザ（もしくはデベロッパ）の作成したプログラムの特性は，ユーザ（もしくはデベロッパ）以外の誰も知り得ず，提案する計算サーバ選択フレームワークもそれを知らないためである．もしユーザがその特性を熟知していないとしても，パラメータを容易に変更できるインタフェースをフレームワークが提供することでユーザは即座にパラメータを変更することができ，またその特性を知るのにも役立つ．

グリッドでは無限とみなしてよいほどの多種多様な資源があり，これらの内でネットワーク的に近く同質の資源を選択して利用することができれば，協調並列型ジョブ実行の問題の有効な解決策になる．この選択操作は，動的な資源の情報を管理する機構と，その情報を検索し利用する資源を決定する部分からなり，ユーザは資源の異質性を考慮することなくグリッドの資源を有効に利用することが可能となる．本研究では，

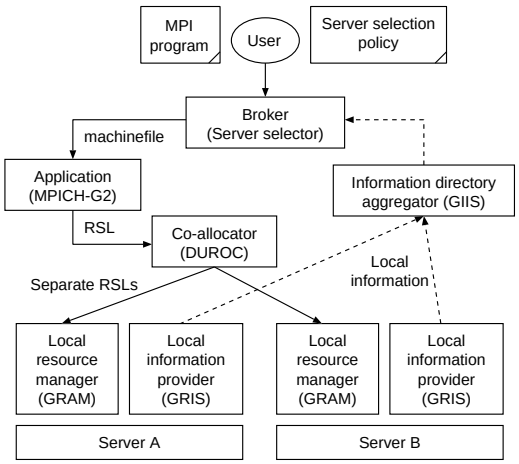

図 3 計算サーバ自動選択システムの概念図
Fig. 3 Overview of automated server-selection system

図 3 に示す計算サーバ自動選択システムを提案した．

提案システムは次の 3 段階より構成される．第 1 に，グリッド環境を実現するツールキット Globus の情報管理サービスである Monitoring and Discovery Service（MDS）[7] を用いて計算サーバの性能や負荷情報を取得する．MDS は Grid Resource Information Service（GRIS）と Grid Index Information Service（GIIS）の 2 つからなり，各サーバの情報は GRIS により取得し定期的に送信され，全サーバの情報が GIIS で管理されている．ユーザからジョブの要求があると，サーバ選択フレームワークは GIIS から必要な情報を取得する．第 2 に，MDS の情報に基づいて，ジョブ特性を記述した評価式により，どのサーバにジョブを割り当てれば効率的かを示すジョブの割り当てマップを作成する．第 3 に，並列計算に用いられるグリッドミドルウェアである MPICH-G2[8] からグリッド資源管理サービスである Globus Resource Allocation Manager（GRAM）や Dynamically-Updated Request Online Coallocator （DUROC）を用いて実際に計算サーバにジョブを割り当てる．GRAM は単一サイト内でのジョブ実行を管理し，DUROC は複数の GRAM を管理し複数サイト間でのジョブ実行を管理する[9]．資源要求に関する情報は Resource Specication Language（RSL）を用いて交換する．

4. 効率的ジョブ実行のための計算サーバ選択方針

提案するシステムの中枢となる情報処理機構の特徴は，グリッドソフトウェアの開発者の記述する性能要求と，グリッド環境の情報サービスから取得されるプ

リミティブな情報から，マッチング評価を行い自動的に最適な計算サーバを選択することである．同質な資源からなるクラスタと異なり，異質な資源から構成されるグリッド環境ではジョブを実行する時点でその資源の性能を正しく評価することは難しい．以下では，どのような情報を利用すれば資源の性能を推定できるかについて述べる．

まず実際に同じジョブを実行した履歴を利用して性能を推定する方法が考えられる．しかしこの方法は，グリッド環境においては多数のジョブ・多数の資源・多数のパラメータの組み合わせが膨大になり，その膨大な量の情報を管理することは管理者に余計な負担がかかり現実的ではない．しかも組合せの数が膨大になれば，実行時のジョブ・資源・パラメータの組み合わせが履歴に存在する確率は極端に低くなる．その他データの変更・負荷などの情報も考慮すると，さらに組み合わせ数が増大する．

実際のジョブを実行する直前に，同じ特性を持った短時間のジョブを実行する方法も考えられる．しかし「同じ特性を持った」短時間のジョブを作成することは難しい．サーバ自動選択システムはその特性を知らないため，それを自動的に生成することはできない．ユーザが作成する場合でも，まずその特性を熟知している必要があり，さらにその特性を反映する適切なパラメータを設定する必要がある．この方法はユーザに負担を増やすばかりでなく，そもそもそのようなジョブを作成することが不可能な場合もある．例えば，JADE は実対称行列を対角行列になるように収束させていくが，その収束の程度は一様でないため変換の初期と終期ではその挙動が大きく変わる．初期では CPU での計算が大きなウェイトを占めるが，終期ではネットワークでの通信が大きなウェイトを占める．

そこで，提案するフレームワークでは，ユーザが実行するジョブの特性を反映したサーバ選択のための評価式を記述することとした．評価式は実行時間を近似したものや，あるいは課金額の少ない資源を選択するようなものなどが考えられる．評価式に必要な情報の種類は，MDS であらかじめ登録しているものでもよく，新たに必要な情報があればそれを追加することは容易である．

グリッドソフトウェアのユーザもしくはデベロッパの記述する評価式の例を**図 4** に示す．`$(·)` は MDS に登録されている数値に置き換えらる．`$(Mds-Memory-Vm-Total-freeMB) >= 150` は論理式で，式が成立すれば 1，成立しなければ 0 になる．つまり成立しなければ評価式全体が 0 になるので，使用可能メモリ量が 150 MB 以上でなければならないという必要条件を表している．CPU の負荷 `$(Mds-Cpu-Load-1min)` に 1 を加えているのは，投入しようとしているジョブを考慮しているためである．この負荷を CPU の動作周波数 `$(Mds-Cpu-speedMHz)` で割り係数を掛けたものは CPU で消費される実行時間を表していると考えられる．係数をネットワークの帯域幅 `$(Nws-bandwidthTcp)` で割ったものと，ネットワークの遅延 `$(Nws-latencyTcp)` に係数を掛けたものは通信で消費される実行時間を表していると考えられる．この CPU と通信で消費される時間を合わせたものを全体の実行時間とすると，その逆数をとれば，評価値の高い資源を選択すればよいことがわかる．

表 1 実験機の性能
Table 1 Specs of examination machines

	CPU	Memory	OS
PC A	Pentium 4 1.7 GHz	512 MB	Linux 2.4.19
PC B	Pentium 4 2.53 GHz	883 MB	Linux 2.4.19
Cluster A (6 nodes)	Pentium 4 1.6 GHz	896 MB	Linux 2.4.20
Cluster B (12 nodes)	Pentium III 1.266 GHz	1 GB	Linux 2.4.18
Cluster C (40 nodes)	Pentium III 1.4 GHz	1 GB	Linux 2.4.9

提案システムでは，評価式と MDS から得られる情報から各ノードの評価値を求める．評価値はソートされ，高いものから指定されたノード数だけ選択される．その後，選択されたノードはソートの順にファイルに書き込まれ，MPI 実行環境からファイルが読み込まれてジョブが実行される．

このように自由度の高い設定が可能なフレームワークをユーザもしくはデベロッパに提供することで，ユーザがもしくはデベロッパがジョブ特性に忠実なチューニングを行うことができる．また評価関数が定式化されているため，特性について定量的な議論が行えるほか，デベロッパからユーザへの（もしくはユーザからユーザへの・ユーザからデベロッパへの・デベロッパからデベロッパへの）特性の伝達が確実に行える．

5. 評　　価

Globus 2.4 と MPICH-G2 1.2.5.1 を用いて，**表 1** に示す CPU の性能，負荷，管理ドメインなどの異なる 60 台のマシンからなる環境上で JADE の計算時間を測定した．使用する CPU 数を変化させて，本システムを使用したときと使用しなかったときの解析時間の比較を **図 5** に示す．評価式は先に挙げた 図 4 を

```
( $(Mds-Memory-Vm-Total-freeMB) >= 150 ) / ( 9260 * ( $(Mds-Cpu-Load-1min) + 1 )
/ $(Mds-Cpu-speedMHz) + .517 / $(Nws-bandwidthTcp) + 15.5 * $(Nws-latencyTcp) )
```

図 4 評価式の例
Fig. 4 An example of measure equation

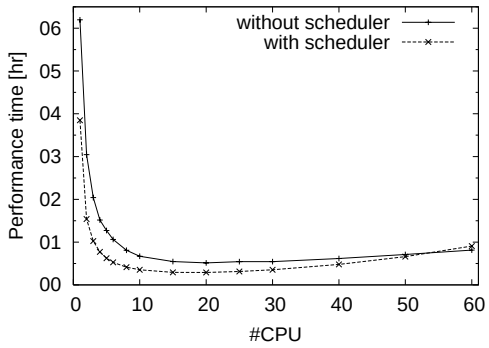

図 5 解析時間の比較
Fig. 5 Comparison of performance time

使用し，その係数はほぼ最短の解析時間となるように調整しておいた．

その結果，本システムを用いたときには CPU 数 60 以外の全ての範囲で解析時間が短縮できることが示された．CPU 数 60 の時は計算サーバ選択による効果がないため，選択に要するオーバヘッドが解析時間を大きくしたと考えられる．また，20 前後より CPU 数を増加させても逆に解析時間が増大していることから，JADE の効率的実行にはネットワークの性能や負荷に大きな影響を受けることが分かる．このように実験で得られた知識を評価式に反映させることで，より効率的なジョブの実行が可能となる．また特に，全 CPU 数に占める使用 CPU 数の小さい場合において短縮率が大きい．これは，全資源数の大きいグリッド環境において本システムが有効なものであることを示している．

広域環境におけるシステムの有効性を実証するため，昨年 11 月に Baltimore で行われた国際会議 SC2002[10] の会場，大阪大学サイバーメディアセンター，産業技術総合研究所ライフエレクトロニクス研究ラボ，Nanyang Technological University をネットワークで接続した環境でも実験を行った．本システムの利用により，ネットワークの状態も考慮した計算サーバの適切な選択が自動的に行われ，広域環境においても有効であることを確認した．

6. 関 連 研 究

グリッド環境での効率的な MPI プログラムの実行に対する需要は高い．MPI の集団通信をグリッド環境に最適化した研究[11] は，データの要素を各プロセスに分配するアルゴリズムを提案しているが，MPI のソースコードを変更する必要がある．Condor Matchmaking[12] は，ポリシや資源情報の記述とそれらの組合せを決定する手法で実績があるが，同時に複数の資源の組合せを決定することは不可能である．これらは，MPI プログラムを変更することなくグリッド環境で実行し，かつ柔軟なポリシで制御可能という特徴を有するものではない．

この他に，MPI の代表的な実装を行っている LAM[13] と MPICH[14] の各グループが，スケジューラに対するインタフェースの研究を行っている．MPI-2 規格から追加された動的プロセス管理を実現するために，オペレーティングシステムや，資源やプロセスを管理するスケジューラとのやりとりが必要なためである[15]．グリッド上での MPI の動的プロセス管理が実現すれば，MPI-1 プロセスモデルで仕様となっていた固定プロセス数という制限がなくなり，グリッドが目標としている必要時に必要量の資源利用が可能となる．

LAM では System Services Interface（SSI）という 2 段階のモジュールフレームワークインタフェースを提案している[16]．SSI は，複数のインタフェース実装のうち，どのインタフェース実装を利用するか，またその実装にパラメタを調整できるインタフェースを実行時に選択することを可能とするよう設計されている．現在，Globus 用の SSI も実装されており，ユーザ側から実装側にヒントを与える仕組みである info オブジェクトを用いて，ノード ID と MPI プログラム名の組からなるスキーマファイルによってプロセス実行の方法が記述されている．つまり全プロセスの生成時ではなく，必要プロセス生成時にスキーマファイルを更新すれば，より資源の状態を反映するようになると考えられる．本研究で提案した手法がほぼそのまま利用できることから，MPI の動的プロセス管理に適用することを今後の課題として考えている．

MPICH では Process Manager Interface（PMI）

という MPI 実装やその他の並列処理ライブラリに依存しないインタフェースを提案している[17]．SSI のアプローチとは異なり関数によるインタフェースを提供しているため，info オブジェクトのように MPI 規格にとらわれない豊富な機能を提供するというアプローチを採用している．PMI の実現には，Key-Value Space（KVS）と呼ばれるキーと値の組からなる簡易データに対する put ／ get を通じた，プロセスの情報交換により行われる．現在の MPI 規格に則った SSI よりも標準化の点では劣るが，より高機能な PMI 拡張を MPI 規格に反映することで，スケジューラインタフェースの展開が進むと予想される．

7. 結　　論

本研究では，グリッド環境上での協調並列型ジョブの効率的実行を可能とするスケジューリングシステムとして，計算サーバ自動選択システムを提案，実装，評価した．その情報処理機構は，グリッドソフトウェアの開発者の記述する性能要求とグリッド環境が提供する情報サービスから取得されるプリミティブな情報から自動的に最適（同質）な計算サーバを選択することを実現する．本研究で行われた評価は，本提案システムの有効性を示すものである．今後，協調並列型 MPI ジョブのための最適サーバ選択の精度の向上，さらには時々刻々に変化する負荷状況などに応じてジョブをサーバ間で移動させるマイグレーションに関する研究を推進し，大規模科学計算技術や医療などの科学への応用を積極的に行っていくことを考えている．

謝辞　共同研究に協力いただいた産業技術総合研究所ライフエレクトロニクス研究ラボ，Nanyang Technological University の関係者に感謝いたします．また本研究は，科学研究費補助金特定領域研究（C）「Grid 技術を適応した新しい研究手法とデータ管理技術の研究」（13224059），および文部科学省科学技術振興費主要 5 分野の研究開発委託事業の IT プログラム「スーパーコンピュータネットワークの構築」の助成を受けて行われた．

参 考 文 献

1) biogrid: http://www.biogrid.jp/.
2) Niedermeyer, E. and da Silva, F. L.: *Electroencephalography*, Williams and Wilkins (1993).
3) Frey, J., Tannenbaum, T., Livny, M., Foster, I. and Tuecke, S.: Condor-G: A Computation Management Agent for Multi-Institutional Grids, *Cluster Computing*, Vol. 5, pp. 237–246 (2002).
4) Abramson, D., Giddy, J. and Kotler, L.: High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?, *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS 2000)*, pp. 520–528 (2000).
5) Hyvarinen, A., Karhunen, J. and Oja, E.: *Independent Component Analysis*, John Wiley & Sons (2001).
6) Cardoso, J. F.: High-order contrasts for independent component analysis, *Neural Computation*, Vol. 11, No. 1, pp. 157–192 (1999).
7) Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C.: Grid Information Services for Distributed Resource Sharing, *Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing (HPDC-10)*, pp. 181–184 (2001).
8) Karonis, N., Toonen, B. and Foster, I.: MPICH-G2: a Grid-enabled Implementation of the Message Passing Interface, *Journal of Parallel and Distributed Computing (JPDC)* (2003). to appear.
9) Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W. and Tuecke, S.: A Resource Management Architecture for Metacomputing Systems, *Proceedings of the forth IPDPS Workshops on Job Scheduling Strategies for Parallel Processing*, Lecture Notes in Computer Science, Vol.1459, Springer-Verlag, pp. 62–82 (1998).
10) SC2002: http://www.sc-conference.org/sc2002/.
11) Genaud, S., Giersch, A. and Vivien, F.: Load-Balancing Scatter Operations for Grid Computing, *Proceedings of the 12th Heterogeneous Computing Workshop (HCW 2003)*, p. 101a (2003).
12) Raman, R., Livny, M. and Solomon, M.: Matchmaking: Distributed Resource Management for High Throughput Computing, *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing (HPDC-7)*, pp. 140–146 (1998).
13) LAM: http://www.lam-mpi.org/.
14) MPICH: http://www.mcs.anl.gov/mpi/mpich/.
15) Gropp, W., Lusk, E. and Thakur, R.: *Using MPI-2: Advanced Features of the Message-Passing Interface*, MIT Press (1999).
16) Squyres, J. M., Barrett, B. and Lumsdaine, A.: The System Services Interface (SSI) to LAM/MPI (2003).
17) The MPICH Team: Process Management in MPICH2 (2003).