グリッドコンピューティングを用いた分枝限定法による最適化問題計算

合田　憲人

グリッドコンピューティングは，ネットワーク上の計算機やストレージを用いて高性能計算を実現するための基盤技術である．本稿では，グリッドコンピューティング技術を簡単に紹介しながら，グリッドコンピューティングの最適化問題計算への応用事例として，分枝限定法による最適化問題計算をグリッド上で実現した例を紹介する．

キーワード：グリッドコンピューティング，分枝限定法，最適化問題

1. はじめに

近年，並列計算技術やネットワーク技術の進歩により，インターネットに代表される広域ネットワーク上の計算機やストレージを利用した分散計算が普及しつつある．ネットワークに接続された複数の計算機やストレージを用いて計算を行うことは，既存のインターネット技術のみでも実現はできるが，この場合，ユーザには高度な知識と非常に煩雑な操作が求められる．

例えば，インターネットを経由して遠隔地にある計算機やストレージを利用する場合は，ユーザは利用する計算機やストレージごとにユーザ認証の手続きを行わなければならない．また，ユーザの計算を実行可能な（複数の）計算機をネットワーク上から探し出して確保しなければならない．このような処理は，計算機やネットワークの専門家にとっても難しく，アプリケーションユーザにとっては大きな困難を伴う．

この問題を解決するための基盤技術がグリッドである．グリッドは，上記の例であげたようなユーザ認証や計算機の探索，確保といった処理をグリッドミドルウェアと呼ばれるソフトウェアがユーザに代わって行う．したがって，ユーザは，計算機やネットワークに関する高い知識をもつ必要はなく，自分が実行したいアプリケーションのことのみを考えればよい．グリッドの用途は計算だけでなく，データ処理や実験，観測，ビジネス利用など多岐にわたるが，このうち計算を目的とした技術がグリッドコンピューティングと呼ばれる．

最適化問題は，与えられた制約条件の下で目的関数の値を最小（または最大）にする解を求める問題であり，オペレーションズリサーチ，制御工学，情報工学等の様々な工学分野の問題を解決するために取り扱われている．しかしながら，対象とする問題が大きくなるとともに求解に要する計算時間が非常に大きくなり，その計算に数時間から数ヶ月を要する問題も多く，解決があきらめられている大規模問題も少なくない．そのため，大規模問題の高速求解を実現するための技術として，グリッドコンピューティングに対する期待は大きい．

本稿では，グリッドコンピューティング技術を簡単に紹介しながら，グリッドコンピューティングの最適化問題計算への応用事例として，分枝限定法による最適化問題計算をグリッド上で実現した例を紹介する．

2. グリッドコンピューティング

グリッドコンピューティングの基本的なアイディアは，ネットワーク上の計算機やストレージなどの資源と，これらを利用するユーザをまとめて，仮想的なグループ（仮想組織と呼ばれる）を構成することにある[1]．ユーザは，自分が利用したい計算機やストレージが含まれる仮想組織に登録されれば，登録された仮想組織内の資源を簡単に利用することができる．

これを実現するためには，様々な要素技術が必要となる．図1は，グリッドのインフラ，アプリケーションおよび要素技術の関係を階層的に表現した図である．この図のように，グリッド上でアプリケーションプロ

あいだ　けんと
国立情報学研究所
〒101-8430 千代田区一ツ橋 2-1-2

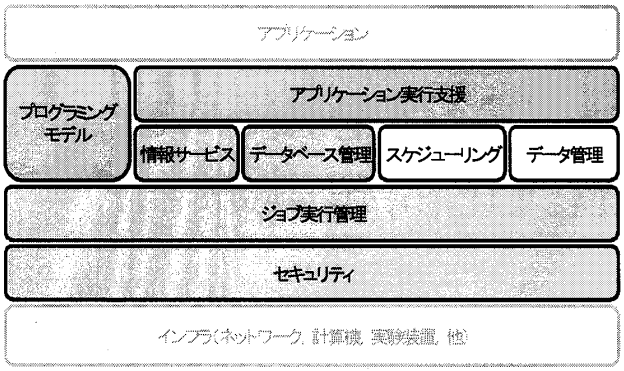

図1　グリッドの要素技術

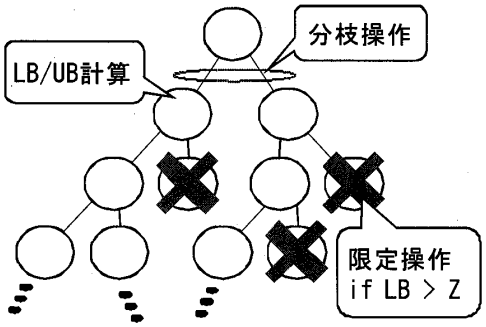

図2　探索木

グラムを実行するためには，その中間層に表記されている様々な要素技術（サービスとも呼ばれる）が必要となる．

ユーザがグリッド上の計算機やストレージにアクセスして，ジョブが安全に実行されるためには，セキュリティ技術が重要である．グリッドで用いられるセキュリティ技術の一つであるシングルサインオンは，ユーザがアプリケーションプログラム実行時に一度だけユーザ認証の手続きを行うことにより，複数の計算機やストレージ上の認証手続きを自動的に行う．

グリッド上で利用可能な計算機やストレージに関する情報は，情報サービスによって提供される．例えば，ユーザのアプリケーションプログラムを実行可能な計算機を探したい場合は，情報サービスへ問い合わせればよい．グリッド上の計算機に対してアプリケーションプログラム（ジョブ）の実行を依頼する，または実行を制御するためには，ジョブの実行管理技術が必要である．また，適切な計算機を選択するスケジューリング技術やジョブが入出力するファイル管理技術は，ジョブの実行性能の向上に役立つ．グリッド上で実行されるジョブがデータベースにアクセスする場合には，グリッドのためのデータベース管理技術が必要となる．

一方，グリッド上でソフトウェアを開発するためには，グリッド上のプログラミングモデルが必要となる．さらに，アプリケーション実行支援技術は，ユーザにより使いやすいアプリケーション実行環境を提供し，より高度な処理を可能とする．

3.　分枝限定法とその並列化

最適化問題とは，以下の式で定式化されるように，与えられた制約条件 S の下で目的関数 $f(x)$ の値を最小（または最大）にする解を求める問題であり，オペレーションズリサーチ，制御工学，情報工学等の様々な工学分野の問題を解決するために解かれている．

$$\text{minimize/maximize } f(x_1, x_2, \cdots, x_n)$$
$$\text{subject to } (x_1, x_2, \cdots, x_n) \in S$$

最適化問題の求解手法として知られる分枝限定法[2]は，与えられた問題を複数の子問題に分割することを再帰的に繰り返すことにより，最適解を探索する手法である．ここで生成される子問題はそれぞれ独立して計算することができる．そのため，これらの子問題を異なる計算機上で並列に計算することにより，計算時間を短縮できることが知られている．

3.1　分枝限定法

分枝限定法の手順は，図2に表されるような探索木により説明することができる．分枝限定法では，初めに探索木の根ノードに相当する問題を複数の子問題(図中では2個の子ノードに相当）に分割する．本操作は，分枝操作と呼ばれる．次に生成された各子問題について，目的関数の下界（LB）と上界（UB）が計算され，さらに，これまでに計算された子問題の上界を比較することにより，その最小値である暫定値(Z) を計算する[1]．

以上の操作は，生成される子問題について再帰的に繰り返され，その結果，図2に示される探索木が生成される．暫定値は，冗長な探索を省くために行われる限定操作に用いられる．限定操作では，各子問題についてその下界（LB）と暫定値（Z）との比較が行われ，下界が暫定値よりも大きい子問題については，それ以上分枝操作を続けてもよりよい解が得られないため，探索木から削除する．最後に，以上の操作を繰り返すことにより得られる暫定値と最小下界が一致するかその差が一定の閾値以下になった場合，全体の計算

[1] ここでは目的関数を最小化する最適化問題を想定している．

が終了する．

3.2 並列プログラミングモデル

グリッド上で実行される並列プログラムを作成する方法は，主にノード間直接通信モデルとマスタ・ワーカモデルに分けることができる．

ノード間直接通信モデルは，計算機間でメッセージを交換しながら処理を進めるモデルであり，PCクラスタ上での並列プログラミング方式として広く用いられているMPI (Message Passing Interface)[3]がその代表例である．

マスタ・ワーカモデルは，計算機群を1台のマスタと複数のワーカに分け，マスタが複数のワーカに計算を割り当てることにより並列計算が進められるモデルである．具体的には，まず全体の計算が複数の計算(タスク）に分割され，マスタは未処理のタスクを保持するとともに，アイドル状態（計算を実行していない状態）のワーカに未処理タスクを割り当てる．タスクを割り当てられたワーカは，タスクの計算を行うとともに計算結果をマスタに通知する．

3.3 分枝限定法プログラムの並列化

繰り返しになるが，分枝限定法では，それぞれの子問題の計算は並列に実行できる．そのため，分枝操作により生成される子問題を次々と計算機に割り当てるプログラムをマスタ・ワーカモデルにより作成することが容易である．

分枝限定法プログラムのマスタ・ワーカモデルによる並列化は，図3に示すように集中制御型と分散制御型に分けることができる．集中制御型では，マスタ(M）が分枝操作を行って探索木を生成し，探索木の葉に相当する子問題の計算をワーカ（W）に割り当てる．一方，分散制御型は，各ワーカ（W）が分枝操作を行って探索木の一部（部分木）を生成する方式であり，ワーカ上で新たに生成された子問題は，マスタに集められ，アイドル状態のワーカに割り当てられる．

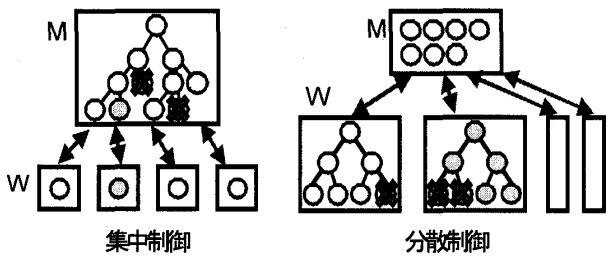

図3 集中制御と分散制御

分枝限定法プログラムの並列化では，暫定値の共有方法についても考えなければならない．分枝限定法では，最新の暫定値を用いて限定操作を効率よく行い，探索空間をより小さくすることが重要となる．したがって，並列計算においても，あるワーカ上で新しい暫定値が得られた場合，それをなるべく早く他のワーカに伝える必要がある．暫定値の他のワーカへの通知が遅れると，他のワーカ上では古い暫定値を用いたまま限定操作が行われるため，本来は不要な子問題の計算を行うことになってしまう．

4. グリッド上での分枝限定法の並列化例

本節では，マスタ・ワーカモデルを用いた分枝限定法プログラムのグリッド上で並列化事例を紹介する．この事例では，国内の4か所に分散したPCクラスタからグリッド実験環境を構築し，その上でBMI固有値問題[4][5]を分枝限定法により解くプログラムを並列化した[6]．

BMI固有値問題は，以下の式で表される双線形行列関数の最大固有値を最小化する問題であり，制御工学やオペレーションズリサーチの分野で扱われているが，大規模問題の計算には大きな計算時間を要することが知られている．

$$F(x,y)=F_{00}+\sum_{i=1}^{n_x}x_iF_{i0}+\sum_{j=1}^{n_y}y_jF_{0j}+\sum_{i=1,j=1}^{n_x}\sum^{n_y}x_iy_jF_{ij}$$

where

$x=[x_1,\cdots,x_{n_x}]\in\Re^{n_x}$, $y=[y_1,\cdots,y_{n_y}]\in\Re^{n_y}$

$F_{ij}=F_{ij}^T\in\Re^{m\times m}$, $i=0,\cdots,n_x$, $j=0,\cdots,n_y$

4.1 分散制御による並列化

マスタ・ワーカモデルでは，マスタがワーカにタスクを割り当てる場合およびワーカからマスタが結果を受け取る場合に，マスタとワーカ間に通信が発生する．これは，通信時間の大きなグリッド上では大きな問題となる．特に，分枝限定法プログラムでは，数千から数万個の子問題計算を行う場合も多く，さらに1つの子問題の計算時間が比較的短い場合も多いことから，通信時間が全体の計算時間に与える影響は大きい．

本事例では，通信時間削減のため，分散制御型の並列化を行った．集中制御型では，計算を行う子問題ごとに，マスタからワーカへ子問題の入力データを送信しなければならない．これに対して分散制御型では，ワーカ上で分枝操作が行われるため，マスタから1個の子問題の入力データを送信するだけで，複数の子問題計算を行うことができる．

効率のよい並列計算を実現するためには，マスタ・ワーカ間の通信時間はワーカ上でのタスクの実行時間よりも十分に小さいことが必要となる．分散制御型では，ワーカ上で生成される部分木の大きさを通信時間に対して十分に大きくするように調整することで，さらに効率のよい並列計算を実現することもできる．

4.2 階層的マスタ・ワーカモデル

分散制御型の並列化を行うことにより，ある程度の通信時間削減は可能であるが，通信時間の大きいグリッド上ではまだ不十分である．これに対して，本事例では，以下に説明する階層的マスタ・ワーカモデルを用いることにより，通信時間のさらなる削減を行っている．

階層的マスタ・ワーカ方式では，単一のマスタと複数のワーカからなるグループが複数構成され，スーパバイザと呼ばれるプロセスがこれら複数のグループを統括する．タスクのワーカへの割り当ては，スーパバイザからマスタへの割り当て，マスタからワーカへの割り当てという2段階で行われ，ワーカ上の計算結果の返却は，逆の順序で同様に行われる．

本方式では，同一グループに属するマスタとワーカを1台のPCクラスタに配置することにより，マスタとワーカ間の通信をPCクラスタ上の高速ネットワーク内に局所化し，マスタ・ワーカ間の通信時間を軽減できる．また，マスタの処理を複数の計算機に分散するため，単一マスタの処理がボトルネックとなる問題も解決できる．

図4は，階層的マスタ・ワーカ方式による並列分枝限定法アルゴリズムの概略を示す．図中，S，M，Wはそれぞれ，スーパバイザ，マスタ，ワーカを意味する．また，マスタとワーカのグループ，例えば図中のマスタ（M_1）とワーカ（$W_{11}, W_{22}, \cdots$）は，1台のPCクラスタに配置されることで，通信時間が削減される．

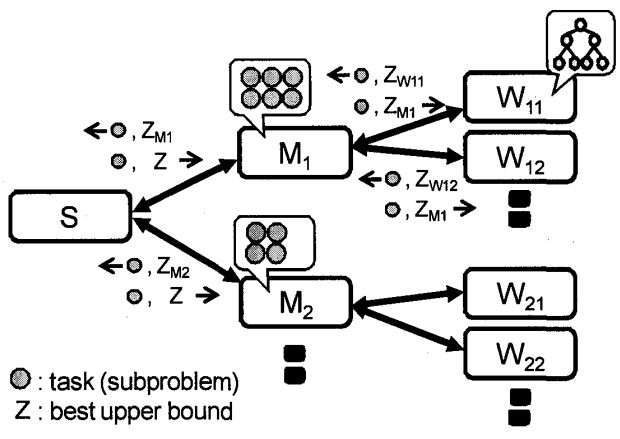

図4　階層的マスタ・ワーカ方式を用いた分枝限定法プログラムの並列化

各PCクラスタ上では，マスタ内に未処理の子問題を保存するキューがあり，マスタが複数のワーカに対してキュー中の子問題を割り当てる．ワーカでは，割り当てられた子問題に分枝操作を行って新たな探索木（部分木）を生成し，部分木上の子問題について下界/上界計算，限定操作を実施した後，得られた解（暫定解），暫定値，限定操作により削除されなかった子問題をマスタに送信する．マスタは，受信した子問題をマスタ内のキューに登録し，アイドル状態のワーカにキュー中の子問題を割り当てる．

分枝限定法の並列化では，ワーカ間での暫定値の通知が重要であるが，本事例では，次のように暫定値が通知される．各ワーカは，新たな暫定値（Z_{Wi}）が得られた場合，その値をマスタに通知する．マスタは，ワーカから受信した暫定値（Z_{Wi}）がマスタ上の暫定値（Z_{Mj}）よりも小さい場合はZ_{Mj}の値をZ_{Wi}の値に更新する．次にマスタは，アイドル状態のワーカに対して子問題を割り当てる際に，更新されたZ_{Mj}を送信する．

またマスタは，ワーカ上で分枝操作により生成される部分木の大きさを指定することにより，ワーカに割り当てるタスクの粒度を調整することができる．これは，ワーカ上の1タスク当りの計算量は，ワーカ上の分枝操作により生成される部分木の大きさに依存するためである．ここで，部分木の大きさは，部分木の深さにより指定される．例えば図4では，右上のワーカ（W_{11}）上で深さ2の部分木が生成されている．

スーパバイザの役割は，PCクラスタ間の負荷分散とPCクラスタ間での暫定値の通知である．スーパバイザは定期的に各PCクラスタ上のマスタと通信を行い，PCクラスタ間で負荷の不均衡がある場合は，負荷の高い（多くの未処理子問題を保持している）マスタから負荷の低い（未処理子問題の少ない）マスタに子問題を移動する．また，マスタ上に保存されている暫定値（Z_{Mj}）がスーパバイザ上に保存されている暫定値（Z）よりも小さい場合，Zの値をZ_{Mj}に置き換え，新しいZを全マスタに通知する．

4.3 グリッド上での実験

本事例では，表1に示すような国内4か所に設置されたPCクラスタからグリッド実験環境を構築した．分枝限定法プログラムの実装では，表中のClient PC

表1　グリッド実験環境の構成

計算機名	仕様	設置場所
Client PC	PIII 1.0GHz 256MB mem. 100BASE-T	東京工業大学 すずかけ台 (神奈川県)
Blade	PIII 1.4GHz x2 512MB mem. 100BASE-T	東京工業大学 すずかけ台 (神奈川県)
PrestoIII	Athlon 1.6GHz x2 768MB mem. 100BASE-T	東京工業大学 大岡山 (東京都)
Sdpa	Athlon 2GHz x2 1024MB mem. 1000BASE-T	東京電機大学 (埼玉県)
Mp	Athlon 2GHz x2 1024MB mem. 100BASE-T	徳島大学 (徳島県)

がスーパバイザとして使用され，Blade，Presto III，Sdpa，Mpの各PCクラスタ内では，1台の計算ノードがマスタ，残りの複数計算ノードがワーカの役割を担当している．

Client PCと各PCクラスタ上には，グリッドミドルウェアとしてGlobus Tool Kit (Ver. 2.4)[7]が運用されている．Globus Took Kitは，グリッド上のミドルウェアのデファクトスタンダードとして知られているソフトウェアツールキットであり，図1に示したグリッドの要素技術のうち，主にセキュリティ，情報サービス，ジョブ実行管理の機能を提供している．具体的には，本実験では，各PCクラスタ上でのユーザ認証（シングルサインオン）がGlobus Tool Kitが提供するGSIと呼ばれる機能を用いて実現されている．また，各PCクラスタの情報は，同じくMDSと呼ばれる情報サービス機能によって収集，管理されている．各PCクラスタ上には，gatekeeperを呼ばれるプロセスが稼働しており，gatekeeperを通してClient PCから割り当てられた子問題計算（ジョブ）がPCクラスタの計算ノードに割り当てられる．

本実験で用いたプログラムは，Ninf-G[8]およびNinf[9]と呼ばれるグリッド上のプログラミングツールを用いて記述されている．これらは，GridRPCと呼ばれるグリッド上でリモートプロシージャコール(遠隔手続き呼び出し）を行うためのプログラムを作成するプログラミングツールである．GridRPCでは，ユーザプログラム中で関数呼び出しと似た要領で遠隔

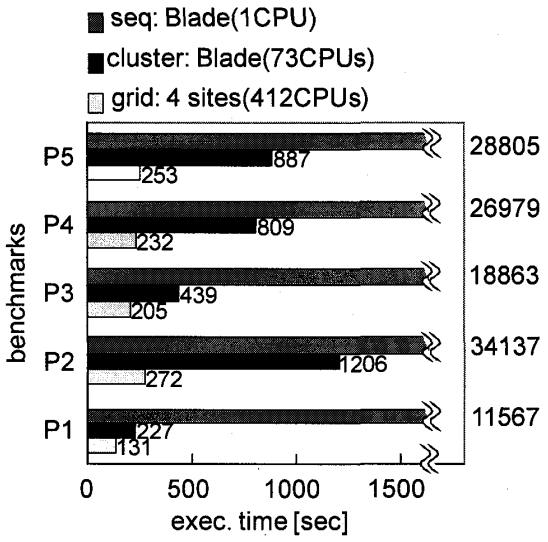

図5　グリッド上での実行時間

地の計算機上で計算を起動することが可能であり，マスタ・ワーカモデルにおいて，マスタからワーカに計算を割り当てる処理を簡単に記述することができる．

図5は，グリッド実験環境上で5種類（P1～P5）のBMI固有値問題計算を行った場合の実行時間の例を示している．この計算では，合計412 CPU（Blade上の73 CPU，Presto III上の129 CPU，Sdpa上の81 CPU，Mp上の129 CPU）を用いて，並列計算を行っている．

図中の結果より，グリッド上で階層的マスタ・ワーカ方式を用いてアプリケーションを並列化することにより，アプリケーションの実行時間を短縮できることがわかる．例えば最も高い性能を示したP2では，逐次実行では，約9時間半の実行時間を要していたが，グリッドコンピューティングにより実行時間が4.5分に短縮されている．

4.4　ユーザインタフェース

一般的に，グリッド上でのプログラム開発や実行は，逐次計算機上での開発に比べて高度な技術を要する．これは多くのアプリケーションユーザにとっては，大きな問題となる．そこで，プログラム開発はグリッドプログラミングの専門家が（アプリケーション専門家と共同で）行い，プログラムを簡単に実行するためのユーザインタフェースを作成して，アプリケーションユーザに提供することが有効である．

本事例でも，図6に示すようなユーザインタフェースが用意されている．ユーザは，Webブラウザ上の本インタフェースを通して，入力パラメータのアップロード，計算の実行開始および停止，途中および最終

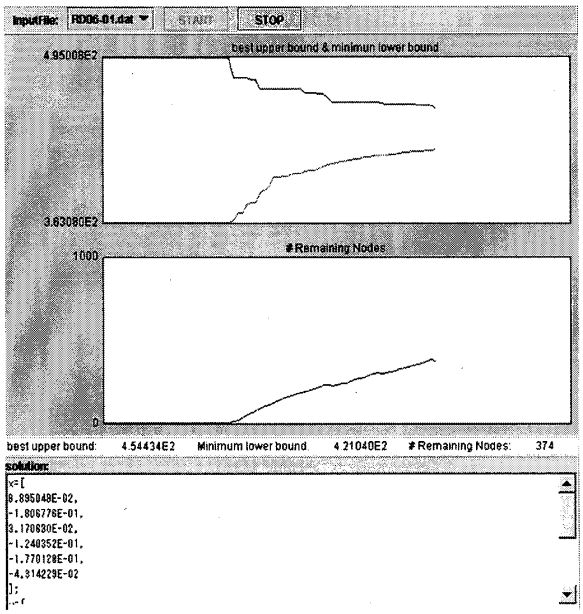

図6　ユーザインタフェース

結果の観測を行うことができる．図中，上段のウィンドウは目的関数の下界と上界の途中計算結果の推移，下段のウィンドウは未処理のタスク数を表示している．これらの途中結果は計算の進行と同時に表示され，ユーザは，途中結果の推移によっては，計算を途中終了し，新たなパラメータを入力して新たな計算を開始することができる．

5. まとめ

本事例では，グリッドコンピューティングを用いた最適化問題計算として，分枝限定法を用いた最適化問題計算の事例を紹介した．

参考文献

[1] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid : Enabling Scalable Virtual Organizations," International Journal of Supercomputer Applications, 15 (3), 2001.

[2] R. Horst, P. M. Pardalos and N. V. Thoai, "Introduction to Global Optimization," Kluwer Acadimic Publishers, 1995.

[3] MPI Forum : http://www.mpi-forum.org/

[4] K. C. Goh, M. G. Safonov and G. P. Papavassilopoulos, A Global Optimization Approach for the BMI Problem, Proc. of the 3 rd Conference on Decision and Control, pp. 2009-2014, 1994.

[5] M. Fukuda and M. Kojima, Branch-and-Cut Algorithms for the Bilinear Matrix Inequality Eigenvalue Problem, Computational Optimization and Applications, 19 (1) : 79-105, 2001.

[6] K. Aida, Y. Futakata and T. Osumi, "Parallel Branch and Bound Algorithm with the Hierarchical Master-Worker Paradigm on the Grid," IPSJ Trans. on Advanced Computing Systems, Vol. 47, No. SIG. 12 (ACS 15), 2006.

[7] Globus Toolkit, http://www.globus.org/

[8] 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.

[9] S. Matsuoka, H. Nakada, M. Sato and S. Sekiguchi, Design Issues of Network Enabled Server Systems for the Grid, Grid Computing-Grid 2000, LNCS 1971, pp. 4-17, 2000.