社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－ARC－162 (14)
2005－HPC－101 (14)
2005／3／8

短時間処理向け資源管理を実現するための Globus Toolkit の性能評価

村木健介[†]　川崎康博[†]　水谷泰治[†]
伊野文彦[†]　萩原兼一[†]

本稿では，グリッドにおいて 10 秒程度の短時間処理を実現することを目標として，標準ミドルウェア Globus Toolkit が内包する資源情報収集システム MDS （Monitoring and Discovery Service）の性能評価を示す．評価には，64 台の PC クラスタを用い，資源情報の要求から収集を終えるまでの応答時間を計測した．その際，MDS を用いた短時間処理の実現可能性を明らかにするために，資源の数あるいはキャッシュ機構の有無などを変更して応答時間の変化を調べた．実験の結果，キャッシュ機構やキャッシュするデータサイズを短時間処理向けに設定することにより，64 台程度であれば，500 ミリ秒で資源情報を収集できることが分かった．

Performance Evaluation of Globus Toolkit to Manage Resources for Rapid Turnaround Processing

Kensuke Muraki,[†] Yasuhiro Kawasaki,[†] Yasuharu Mizutani,[†]
Fumihiko Ino[†] and Kenichi Hagihara[†]

The objective of our research is to realize rapid turnaround processing on Grids (for example, within ten seconds). In this paper, we present a performance evaluation of the Monitoring and Discovery System (MDS), which is distributed with the Globus Toolkit. To evaluate its performance, we measured the response time on a 64-node cluster of PCs. The response time here is defined as the time required for collecting resource information from all the computing nodes. We also analyzed the performance characteristics with different configurations such as numbers of nodes and cache mechanisms, aiming at investigating the possibility of rapid turnaround processing using MDS. As a result, we found that an appropriate configuration enables MDS to gather resource information from 64 nodes within 500 milliseconds.

1. はじめに

グリッド技術とは，異なる組織の計算資源（以下，資源）を統合し，1 つの仮想的な高性能計算環境を構築する技術である．その用途は様々であり，例えば，家庭や企業内の遊休資源を用いた大規模計算[1] や大量データ処理[2] などが挙げられる．また，このようなシステムを構築するための標準ミドルウェア GT[3]（Globus Toolkit）も開発されている．これらは，数日以上の実行時間を要する長時間処理に対し，高スループット（秒間当たりの処理数）かつ高性能な計算機環境を提供できる．

一方，長時間処理と同様に，短時間処理をグリッドにおいて実現できれば，グリッド技術の有用性は高まる．例えば，病院内の数百台からなる遊休資源を用い，1 台で 1 時間を要する医用画像処理を 10 秒程度に短縮できれば[4]，手術などの時間制約の強い場においてもグリッド技術は有用である．この際，より多くの資源を多くの組織から集めるためには，GT などの既存の標準ミドルウェアを用いることが不可欠である．

しかし，多くの既存ミドルウェアは長時間処理を対象にしているため，短時間処理のための設計が必要である．具体的には，実行対象とする処理そのものを開始するまでの時間が長く，全体として 10 秒程度の短時間内に処理を完了できない可能性がある．例えば，Zhang ら[5] によると，GT は 500 台からなるグリッドにおいて，各資源の遊休状態に関する情報（以下，資源情報）を収集するために 20 秒弱を要する．

この場合，収集のための時間は処理対象そのものよりも長い．最悪の場合，グリッドによる実行時間短縮に失敗することもありえる．更に，この時間は資源の

† 大阪大学大学院情報科学研究科コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

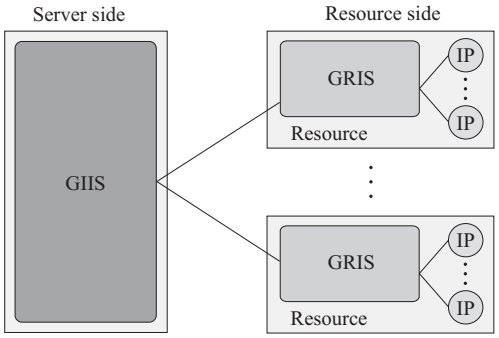


図 1 MDS のアーキテクチャ

数とともに増大するため，実行時間短縮のためにより多くの計算資源を統合することは，短時間処理の実現を妨げる．ゆえに，短時間処理のためのミドルウェアは，自身が迅速に動作できる必要がある．

そこで本研究では，グリッドにおける短時間処理の実現を目的として，そのための標準ミドルウェアを開発する．本稿では，開発のための予備実験として，既存の標準ミドルウェア GT が内包する資源情報収集システム MDS[6]（Monitoring and Discovery System）に対する性能評価を示す．更に，その評価結果を基に，短時間処理を目的とする資源情報収集システムの設計について考察する．

評価実験には，64 台構成のクラスタを用い，資源情報の要求を受けてからその収集を終えるまでの応答時間を計測した．その際，MDS を用いた短時間処理の実現可能性を明らかにするために，資源や資源情報の数，あるいはキャッシュ使用の有無を変更することにより応答時間の特性を解析した．

以降では，まず 2 章で資源情報収集システム MDS の概要を示す．次に，3 章で MDS の性能評価を示す．その結果をもとに，4 章で MDS の応答時間を短縮する方法を述べ，短時間処理のための設計について考察する．5 章で関連研究を紹介する．最後に，6 章で本稿をまとめる．

2. MDS（Monitoring and Discovery System）

本章では，MDS の概要について述べる．MDS は，グリッドを構成する資源の静的および動的情報を収集できる．例えば，CPU の駆動周波数や OS の種類などの静的情報や，CPU の負荷やメモリの空き容量などの動的情報を管理できる．

2.1 アーキテクチャ

図 1 に，MDS のアーキテクチャを示す．MDS は階層構造に基づいていて，3 つの要素からなる．各要素は，役割に応じて GIIS（Grid Index Information Service），GRIS（Grid Resource Information Service）および IP（Information Provider）と呼ばれる．中間層の GRIS は，各資源に 1 つずつ存在し，その資源情報をまとめて管理する．最下層の IP は，GRIS の管理下にあり，資源情報の種類ごとに 1 つずつ存在する．例えば，CPU の負荷を監視する IP や OS の種類を保持する IP などがある．最後に，最上層の GIIS は複数の資源をまとめ，それらの資源情報を収集するサーバの役割を担う．

IP や GRIS の数は任意に変更できる．例えば，新しい資源情報を追加する場合は，その情報を監視する IP を実装し，GRIS に登録すればよい．同様に，新しい資源を追加する場合は，その資源で GRIS を動作させ，GIIS に登録すればよい．

2.2 情報収集の流れ

資源情報を収集する際の MDS の動作を示す．なお，以下の例ではキャッシュ機構（後述）はないものとする．

(1) 待ち受け：GIIS が情報収集の問い合わせを受ける．
(2) 資源への情報要求：GIIS は，管理下の GRIS に対して資源情報を要求する．
(3) 資源情報の収集：GRIS は，管理下の IP に対し資源情報を要求する．IP は，担当する資源情報を取得し，GRIS に送信する．
(4) 資源からの情報受信：GRIS は，IP から収集した資源情報をまとめて GIIS へ送信する．
(5) 収集情報の提示：GIIS は，GRIS から収集した資源情報をまとめて出力する．

なお，初期設定において，GIIS および GRIS はすべての IP が取得する資源情報をやりとりする．一方，特定の資源情報のみを要求することもでき，その場合，処理（1）において資源情報の種類を指定する．例えば，CPU に関する情報のみを GIIS へ問い合わせ，その情報のみを収集できる．

以降では，上記の処理（1）～（5）を完了するまでに要する時間を応答時間と呼ぶ．

2.3 キャッシュ機構

応答時間を短縮することを目的として，MDS はキャッシュ機構を備える．キャッシュは GIIS および GRIS に存在し，あらかじめ定めた有効期間の間，各々が持つ資源情報を保持する．有効期間内に資源情報の要求があった場合，各々はキャッシュが保持する資源情報を返す．したがって，GRIS および GIIS におけるキャッシュが有効である場合，各々，処理（3）およ

び処理（2）～（4）を省ける．

このように，キャッシュ機構は GIIS および GRIS 間の情報送信や IP による資源情報の取得を省け，応答時間を短縮できる．しかし，キャッシュが保持する資源情報は必ずしも最新の遊休状態を表さない．したがって，キャッシュの有効期間が不適切に長い場合，過去の資源情報に基づいて遊休資源を誤って選択する可能性があるため，注意を要する．

2.4 応答時間に影響を与える項目

MDS の設定や動作の中で，以下の項目が応答時間に影響を与えると予想する．

- GIIS が GRIS から収集する資源情報のデータサイズ
- GIIS に接続する GRIS の数
- IP が資源情報を取得する際の実行時間
- キャッシュの使用の有無

長時間処理を対象にしている GT の初期設定において，以上の項目については応答時間を考慮していない．しかし，短時間処理を扱う際には応答時間の短縮を要する．したがってこれらの項目の設定を変更する必要があるが，応答時間がどの程度短縮できるかは明らかでない．そこで，これらの項目が応答時間にどの程度影響を与えるかを調べる．

3. 性 能 評 価

本章では MDS の応答時間を評価する．実験の目的は，MDS の設定による応答時間のスケーラビリティを知ることであり，2.4 節で述べた設定項目をもとに，以下の設定について評価する．

実験 1： GRIS が GIIS に送信する資源情報のデータサイズ

実験 2： GIIS に接続する GRIS の数

実験 3： GRIS に登録する IP の数

実験 1〜3： キャッシュ使用の有無

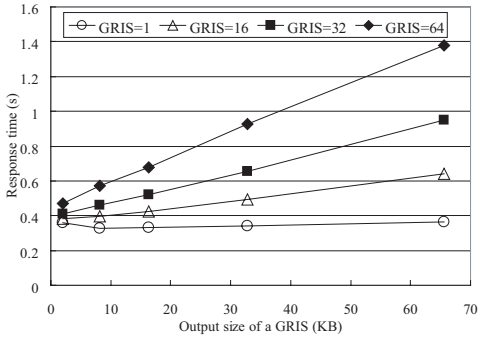
以上の評価を行うことで，応答時間の短縮方法を知ることができる．

3.1 実 験 環 境

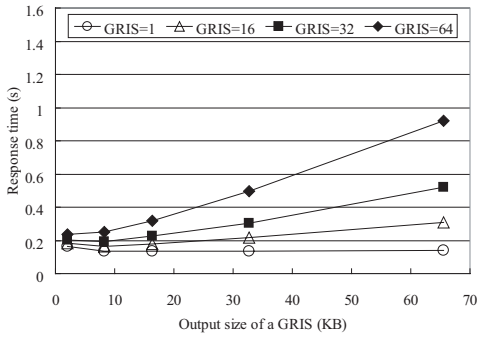
実験では PC クラスタを使用した．PentiumIII 1.133GHz プロセッサを 2 つ搭載した PC1 台において GIIS を動作させ，PentiumIII 1GHz プロセッサを 2 つ搭載した PC64 台において GRIS を動作させる．メモリの容量はいずれも 2GB である．OS は Red Hat Linux を用いた．GIIS と GRIS の間は Fast Ethernet ネットワーク（通信バンド幅 100Mb/s）で接続する．

3.2 応答時間の測定について

同じ条件での応答時間のばらつきは微小であるが，



(a) キャッシュなし



(b) キャッシュあり

図 2 GRIS の送信データサイズと GRIS の数による応答時間の比較

以降の実験結果においては 3 度計測した平均を応答時間の値とした．キャッシュ機構については，GIIS と GRIS 全てのキャッシュがない場合と，キャッシュがある（GIIS のキャッシュが資源情報を保持している）場合について計測した．

3.3 実験 1 - GRIS の送信データサイズによる応答時間の比較

実験 1 は GRIS の送信データサイズを変更する．この実験では，IP として文字列を出力するだけのプログラムを用いて，その文字列の長さを変えることで GRIS の送信データサイズを変更する．MDS の初期設定における送信データサイズが約 30KB であることから，実験におけるデータサイズは 2〜64KB の範囲とした．また IP の総実行時間を固定するために，実験における IP の数は 8 つに固定した．IP の総実行時間が変わると，送信データサイズの変更による応答時間の正確な比較ができないためである．IP の数を 8 としたのは，GRIS の送信データサイズ設定の便宜

表 1 データサイズが 62KB 増加した際の応答時間の増加

GRIS	増加時間 （秒）	
の数	キャッシュなし	キャッシュあり
16	0.26	0.13
32	0.54	0.32
64	0.91	0.69

上の理由による．

実験結果について，まず図 2(a) にキャッシュがない場合の応答時間を示す．系列の GRIS は GIIS に接続する GRIS の数を表す．図 2(a) では，GRIS の送信データサイズに対して，応答時間が線形に増加している．GIIS が収集するデータサイズの増加により，GIIS の負担が増加することが原因だと考えられる．

続いて，図 2(b) にキャッシュがある場合の結果を示す．図 2(a) と (b) を比較すると全体として，キャッシュがある場合はキャッシュがない場合の 1/2 程度の応答時間であることが分かる．例えば，図 2(a) において GRIS の送信データサイズが 32KB および GRIS の数が 64 の場合において応答時間は 0.9 秒であるのに対し，2(b) においては 0.5 秒である．応答時間が減少するのは，資源情報をキャッシュがすることにより IP の実行と GRIS のデータ送信が省かれるためである．

続いて，応答時間の増加に着目する．表 1 にデータサイズが 2KB から 64KB に増加した際の応答時間の増加を示す．表 1 より，キャッシュがある場合の増加時間は，キャッシュがない場合の増加時間の約 5〜8 割であることが分かる．例えば GRIS の数が 64 においては，キャッシュがない場合，GRIS 1 つあたりのデータサイズが 62KB 増加すると応答時間が約 0.9 秒増加する．一方，キャッシュがある場合は，GRIS 1 つあたりのデータサイズが 62KB 増加すると応答時間が約 0.7 秒増加しており，キャッシュがない場合の約 0.9 秒増加と比べると，違いは小さい．この結果より，送信データサイズの増加による応答時間の増加を削減するという点では，キャッシュの効果が小さいといえる．

3.4 実験 2 - GRIS の数による応答時間の比較

実験 2 は GRIS の数を変更する．使用する IP の種類や数は実験 1 と同様であり，GRIS の数は 1〜64 の範囲とした．

図 2 では，キャッシュの保持の有無に関わらず，GRIS の数の増加に伴い応答時間も増加している．キャッシュがある場合に GRIS の数の増加に伴い応答時間が増加する理由を考察する．2.3 節で示したようにキャッシュがある場合は，キャッシュからのデータの取得しか行わない．したがって，応答時間が GRIS の数に応じて増加している理由は，GRIS の数の増加に応じてキャッシュのデータが増加するため，GIIS がキャッシュからデータを読み出す時間が増加するからであると考えられる．

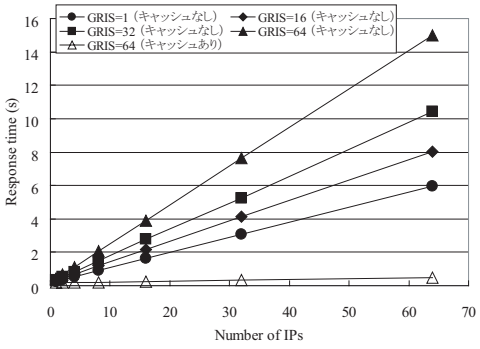


図 3 登録 IP の数による応答時間の比較

3.5 実験 3 - IP の数による応答時間の比較

実験 3 は GRIS に登録する IP の数を変更する．この実験では，指定されない資源情報を取得する IP の振る舞いを知るために，IP_i（数を増やす IP）と IP_f（数を増やさない IP）の 2 種類に分け，GIIS への問い合わせ時には IP_f が出力する資源情報を指定する．IP_i の数を増やしつつ応答時間を計測した．

図 3 に実験の結果を示す．横軸に IP_i の数，縦軸に応答時間をとった．系列は GIIS に接続する GRIS の数で分け，さらに GRIS の数が 64 の場合ではキャッシュがある場合とない場合で分けた．キャッシュありの系列が 1 つである理由は，図 3 中では GRIS の数による違いが微小であることによる．

図 3 より，ユーザが指定して受け取る資源情報には IP_i が取得する情報が含まれないにも関わらず，応答時間は IP_i の数に応じて増加していることが分かる．また，GRIS の数が 1 でキャッシュがない場合を例にとると，IP_i の数が 63 増加すると応答時間は約 6 秒増加している．IP_f の実行時間は約 0.1 秒，IP_i は約 0.02 秒であるので，$0.02 + 0.1 \cdot 63 = 6.32$ となり，全ての IP の実行時間と応答時間はほぼ一致している．

これらの結果より，問い合わせ時における資源情報の指定の有無に関わらず，MDS は IP を全て実行することがわかる．

図 3 のキャッシュがある場合においては応答時間はほぼ横ばいである．IP の実行時間が増加しても，キャッシュがある場合はその実行時間は全て省略されるためである．3.3 節において，GRIS の送信データサイズの増加による応答時間の増加に対してはキャッシュの効果は小さいことを述べたが，一方で IP の実行時間

の増加に対しては，キャッシュの効果が大きいことが分かる．例えば図 3 の IP と GRIS の数が共に 64 の場合，キャッシュを利用することで応答時間を約 15 秒から約 0.5 秒に短縮している．

4. MDS において応答時間を短縮する方法

実験結果から分かる，MDS の応答時間を短縮する方法について述べる．長時間処理を対象にして MDS を利用する場合はここで述べる方法をとることはない．以下に，応答時間短縮の効果が高い順に述べる．

C1： 資源情報を GIIS で常にキャッシュする
キャッシュ機能は応答時間の短縮に非常に有用であるが，キャッシュの有効期間を過ぎると効果がなくなる．また，キャッシュの有効期間を長くするほどキャッシュから得られる情報の信頼度が低くなる．そこで，ある程度の時間経過毎にキャッシュデータを更新し，常に新鮮な情報を保持する．現在の MDS にはキャッシュを自動更新する機能はないため，新しい機能を MDS に加える，あるいは MDS の使用方法を工夫するなどを試みる必要がある．キャッシュ機能は長時間処理を対象とした場合も利用するが，常にキャッシュする使い方はしない．

C2： GRIS が送信するデータサイズを小さくする
キャッシュを利用しても，データサイズが大きいと応答時間は長くなるため，データサイズの縮小は重要である．

C3： GIIS に接続する GRIS の数を小さくする
GRIS の数が大きくなるほど，データサイズの増加および GIIS の受信負荷の上昇が起こるため，GRIS の数を小さくするのは重要である．

C4： IP の数を小さくする
3.5 節で，IP は情報が要求されていなくても問い合わせ時に実行されると述べた．したがって IP の数が少ないほど応答時間が短くなる．ただし，C1 を利用した場合 C4 は応答時間に対して効果はない．

C5： IP のプログラムの実行時間を短くする
C1 を利用した場合 C5 は応答時間に対して効果はないが，実行時間を短くすることで計算資源を節約することになる．キャッシュを自動更新する際は IP を何度も実行するため，計算資源の節約の点で有用である．同様のことは C4 に対してもいえる．

しかし，これらの方法をとることで，得られる情報量の減少あるいは情報の信頼度低下の欠点が生じる．したがって取り入れる際にはその欠点も考慮する必要がある．以降では，その考慮点について示す．

C1 では，IP をある程度の時間経過ごとに実行する必要があり，その度に計算資源を消費する．したがってキャッシュ更新の頻度を調整し，計算資源に大きな負担を与えないように工夫する必要がある．

C2 では，データサイズを小さくしつつも，得るべき資源情報を省いてはならない．したがってなるべく小さいデータサイズで必要な資源情報を表す必要がある．

C3 では，GIIS に接続する GRIS の数を小さくすると，収集対象の資源数が小さくなる．その解決として，GIIS である PC を複数用意して GIIS 1 つあたりに接続する GRIS の数を小さくする方法が考えられる．この方法により，応答時間を短縮しつつ，収集対象の資源数を保つことができる．

C4 では，IP の数を減らすことは得られる資源情報の種類が減ることを意味する．したがって短時間処理を対象とした場合に不必要な情報を決めなければならない．

C1 〜 C5 の方法を行った場合，MDS の応答時間がどの程度に短縮できるかを考察する．例えば資源数が 500 および GRIS のデータサイズが 32KB の場合，C1〜C3 の方法を行うことで応答時間を 0.5 秒程度にできると考えられる．その方法について詳しく述べる．まず，C3 の方法により，GIIS である PC を 8 台用意すると GIIS1 つにつき GRIS の数は 64 以下になる．そこで C1 の方法を行い，必ず情報が GIIS にキャッシュされている状況にする．3.3 節で述べたように GRIS の送信データサイズが 32KB および GRIS の数が 64 の場合，応答時間は 0.5 秒であるので，さらに C2 の方法を用いると，GIIS が情報を収集する時間は 0.5 秒以下になる．したがって 500 台の資源から 0.5 秒以下で情報を収集できることになる．またその際，C4 および C5 は，計算資源の消費を抑える目的で使用する．

以上の例から，C1〜C5 の方法が有用であることが分かる．資源選択システムの設計においては上記の方法の利点と欠点を踏まえ，応答時間の短縮を実現する．

5. 関 連 研 究

Kondo ら[7] は，数十分程度で完了する短時間処理を対象に，計算資源の選択手法を提案している．彼らの手法は，計算資源の優先度付け，計算資源の排除，およびタスク複製を組み合わせた発見的解法であり，中でも計算資源の優先度付けについては，数十秒程度の短時間処理に応用可能である．

MDS 以外の資源情報収集システムとして，Condor[8)9)] というシステムの一部である Hawkeye がある．Zhang ら[5)] によると，500 台からなる計算機環境において，資源情報を収集するために 30 秒弱を要し，同様の条件における MDS の応答時間より 10 秒程度長い．また，Hawkeye は計算資源の負荷を管理することに特化したシステムである．Ian foster によるグリッドの定義では標準で汎用のプロトコルやインタフェースの使用を定めているため，Hawkeye はグリッドの定義を満たさない．

企業や家庭内のみでなく，グローバルな環境の資源を利用し短時間処理を実現するには，グリッドの定義を満たすことが重要であり，したがって MDS を利用してシステムを構築することにメリットがある．

6. おわりに

本論文では，まずグリッド環境における短時間処理向け標準ミドルウェアの開発のために，資源情報収集システムを設計することを述べた．しかし，既存の標準ミドルウェア GT は長時間処理を対象にしており，短時間処理を扱う際には応答時間を短縮する必要がある．そのため，応答時間を評価基準として GT の資源情報収集システム MDS について評価実験を行い，MDS の応答時間はどのような方法で短縮できるかを調べた．実験から，4 章に挙げた方法により応答時間の短縮が可能であることが分かったが，それらの方法には得られる資源情報量の減少などの欠点がある．したがって，応答時間を短縮する際には設定を工夫する必要がある．

今後は，本論文で得られた知見をもとに，MDS を利用した短時間処理向け資源選択システムを設計し，実装する．

謝辞 本研究の一部は，科学研究費補助金基盤研究（B）（16300006）及び NEC システムプラットフォーム研究所の補助による．

参 考 文 献

1) Sullivan, Woodruff T., I., Werthimer, D., Bowyer, S., Cobb, J., Gedye, D. and Anderson, D.: A new major SETI project based on Project SERENDIP data and 100,000 personal computers, *Proc. 5th Int'l Conf. Bioastronomy*, p. 729 (1997).

2) Chervenak, A., Foster, I., Kesselman, C., Salisbury, C. and Tuecke, S.: The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets, *J. Network and Computer Applications* (2001).

3) Foster, I., Kesselman, C., Nick, J. M. and Tuecke, S.: Grid Services for Distributed System Integration, *IEEE Computer*, Vol. 35, No. 6, pp. 37–46 (2002).

4) Kawasaki, Y., Ino, F., Mizutani, Y., Fujimoto, N., Sasama, T., Sato, Y., Sugano, N., Tamura, S. and Hagihara, K.: High-Performance Computing Service Over the Internet for Intraoperative Image Processing, *IEEE Trans. Information Technology in Biomedicine*, Vol. 8, No. 1, pp. 36–46 (2004).

5) Zhang, X., Freschl, J. L. and Schopf, J. M.: A Performance Study of Monitoring and Information Services for Distributed Systems, *Proc. 12th IEEE Int'l Symp. High Performance Distributed Computing (HPDC'03)*, pp. 270–282 (2003).

6) Czajkowski, K., Fitzgerald, S., Foster, I. and Kesselman, C.: Grid Information Services for Distributed Resource Sharing, *Proc. 10th IEEE Int'l Symp. High Performance Distributed Computing (HPDC'01)*, pp. 181–194 (2001).

7) Kondo, D., Chien, A. A. and Casanova, H.: Resource Management for Rapid Application Turnaround on Enterprise Desktop Grids, *Proc. High Performance Networking and Computing Conf. (SC2004)* (2004).

8) Raman, R., Livny, M. and Solomon, M.: Policy Driven Heterogeneous Resource Co-Allocation with Gangmatching, *Proc. 12th IEEE Int'l Symp. High Performance Distributed Computing (HPDC'03)*, pp. 80–89 (2003).

9) Litzkow, M., Livny, M. and Mutka, M.: Condor - a hunter of idle workstations, *Proc. 8th International Conference. Distributed Computing Systems*, pp. 104–111 (1988).