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

2007－ARC－172 (46)
2007－HPC－109 (46)
2007／3／2

NAREGIミドルウェアβ-gLite間における相互ジョブ起動実験

中田 秀基 †1 佐藤 仁 †2 佐賀 一繁 †3
畑中 正行 †4 佐伯 裕治 †3 松岡 聡 †2,†3

グリッド標準化団体OGFでは，グリッドミドルウェア間の相互運用技術を確立するために．GIN(Grid Interoperation Now)と呼ばれるコミュニティグループの活動が行われている．この活動の一環として，NAREGIのNAREGIミドルウェアβとEGEE (Enabling Grids for E-Science in Europe)のミドルウェアgLiteの間でジョブ起動およびデータ転送に関する相互運用実験を行い，NAREGIミドルウェアβからgLite，およびgLiteからNAREGIミドルウェアβへのジョブ起動を実現した．前者はgLiteの個々の計算資源 (Compute Element)を擬似的にNAREGIミドルウェアβの資源として取り込み，NAREGIミドルウェアβのスケジューラの配下に置くことで実現した．後者は，NAREGIミドルウェアβ全体をひとつの計算資源としてgLiteに対して公開し，gLite側で明示的にNAREGIミドルウェアβを選択する方式で実現した．実験の結果，以下を確認することができた．1) 証明書や仮想組織管理のレイヤでは相互運用性に問題はない，2) 情報サービスのレイヤでも相違が吸収できる，3) その情報を用いて相互のジョブ起動が可能である，

Job invocation interoperability between NAREGI Middleware Beta and gLite

Hidemoto Nakada ,†1 Hitoshi Sato ,†2 Kazushige Saga ,†3
Masayuki Hatanaka ,†4 Yuji Saeki †3
and Satoshi Matsuoka †2,†3

As grid middlewares getting mature, the importance of the inter-operation among them is getting more significant. There is a community group called GIN(Grid Interoperation Now) in the OGF (Open Grid Forum), a standardization body for grid related technologies, which aims to establish interoperation technologies among several grid middlewares. We performed experiments on inter-operation between NAREGI Middleware beta and EGEE gLite, as one of the contributions for the group. For the experiments, we implemented several modules to enable information exchange and mutual job submission. As the result of the experiment, we confirmed the follows: 1) The security layer, such as certficate and virutal organization management, there is no essential difference between them, 2) While information services differs substantially, the resource information can be translated to enable information exchange, 3) Jobs can be mutually submitted based on the exchanged information.

1. はじめに

広域に分散した異なる管理主体に属する資源を共有することで効率的な大規模計算を実現するグリッド技術が普及しつつある．グリッドを実現するためのミドルウェアとして，さまざまなシステムが提案されているが，一般にこれらの間には相互運用性がなく，さらなる普及の妨げとなっている．今後のグリッド技術の更なる発展と普及のためには，ミドルウェア間の相互運用技術を確立する必要がある．

このような観点からグリッド標準化団体であるOGF[1]では，GIN(Grid Interoperation Now)[2]と呼ばれるコミュニティグループの活動が行われている．このグループの目的は既存のグリッドミドルウェアの間で最低限の相互運用実験を行い，その結果を標準化にフィードバックすることである．われわれはこの活動の一環として，NAREGIのNAREGIミドルウェアβとEGEE (Enabling Grids for E-Science in Europe)[3]のミドルウェアgLite[4),5]の間でジョブ起動およびデータ転送に関する相互運用実験を行った．本稿ではこのうち，ジョブ起動の相互運用実験に関して報告する．

われわれは，NAREGIミドルウェアβからgLite，

†1 産業技術総合研究所 National Institute of Advanced Industrial Science and Technology (AIST)
†2 東京工業大学 Tokyo Institute of Technology
†3 国立情報学研究所 Nationa Institute of Inofrmatics
†4 富士通株式会社 Fujitsu Limited

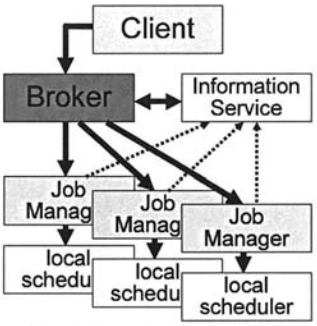

図 1 グリッドミドルウェアの一般形

および gLite から NAREGI ミドルウェア β へのジョブ起動を実現した．前者は gLite の個々の計算資源 (Compute Element) を擬似的に NAREGI ミドルウェア β の資源として取り込み，NAREGI ミドルウェア β のスケジューラの配下に置くことで実現した．後者は，NAREGI ミドルウェア β 全体をひとつの計算資源として gLite に対して公開し，gLite 側で明示的に NAREGI ミドルウェア β を選択する方式で実現した．

実験の結果，以下を確認することができた．1) 証明書や仮想組織管理のレイヤでは相互運用性に問題はない，2) 情報サービスのレイヤでも相違が吸収できる，3) その情報を用いて相互のジョブ起動が可能である，

2. グリッドミドルウェアの構成と相互運用

グリッドミドルウェアにはさまざまな構成要素がありうるが，一般的な構造を図 1 に示す．グリッドミドルウェアは，ブローカ，ジョブマネージャ，情報サービスから構成される．ジョブマネージャは各サイトに存在し，各サイトのローカルスケジューラ (バッチスケジューラ) へのインターフェイスとなる．情報サービスは各サイトからサイトの負荷情報を始めとするさまざまな情報を収集し，ブローカに提供する．ブローカは，クライアントからのジョブ実行リクエストに対して，情報に基づいて，ジョブマネージャを割り当て，実行を依頼する．ジョブは，ローカルスケジューラ経由で実行される．この際，各モジュール間の通信は，すべて適切に認証，認可されていなければならない．

したがって，グリッドミドルウェア間でジョブを相互に実行するためには，1) セキュリティの相互運用，2) 情報サービスの相互運用，3) ジョブサブミッションの相互運用，の 3 点が実現されなければならない．

ジョブサブミッションの相互運用においては，2 つの方針が考えられる．ジョブマネージャレベルの相互運用と，ブローカレベルの相互運用である．前者では，各ミドルウェアのブローカが直接ジョブを互いのジョブマネージャに投入する (図 2)．各ミドルウェアが管理している資源に対して，いわば横からジョブがサブミットされることになる．したがって，ブローカの管理ポリシとは相反するジョブのアロケーションが行われる可能性がある．

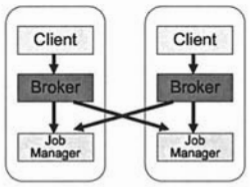

図 2 ジョブマネージャレベルの相互運用

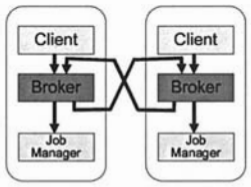

図 3 ブローカレベルの相互運用

これに対して，後者では，ブローカが互いのブローカに対してジョブを投入する (図 3)．相互運用ジョブは，2 段にブローカを経由することになる．各計算資源は，属するミドルウェアのブローカ経由でしかジョブを受け付けないことになり，ブローカによるポリシの強制がより効率よく機能する．

3. NAREGI ミドルウェアの概要

NAREGI ミドルウェア β は，複数サイトリソースのコアロケーションとワークフロー実行にポイントを置いたミドルウェアである．NAREGI ミドルウェアの第一世代である α 版 [6] が，UNICORE をベースにしていたのに対し，β 版は WSRF を基盤とし，Globus Toolkit 4[7] の技術を利用して構成されている．OGF で策定されつつあるさまざまな標準技術を積極的に適用している点にも特徴がある．

NAREGI ミドルウェア β は，多くのモジュールから構成されるが，ミドルウエアの主要モジュールは以下の以下の 3 つである (図 4)．

- **Super Scheduler(SS)**
  SS は，一般的なグリッドミドルウェアの構成上はブローカに相当するモジュールであるが，ブローカリングに加えて，コアロケーションとワークフロー実行も司る．ユーザはグラフィカルな UI でワークフローを作成し，SS に投入する．SS は，ワークフローの各ジョブに対して，後述する GridVM にコンタクトして計算資源を予約し，GridVM を経由してにジョブを投入する．この際，計算資源の検索には後述する IS を用いる．さらにサイトを横断したコアロケーションが必要な場合には，SS が複数のサイトで同時実行可能な時間スロットを発見し，その時間帯を同時に予約することによって，コアロケーションを実現する．
- **GridVM**
  GridVM は，サイト内の計算資源を管理するモジュールで，前出の一般的構成上のジョブマネージャに相当する．ローカルキューイングシステムをラップする形で，事前予約機能を提供すると同

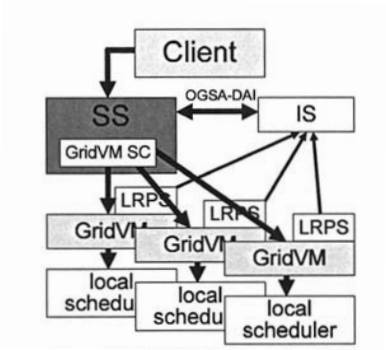

図 4 NAREGI ミドルウェア β の概要

時に，計算資源使用に対する細粒度のコントロール，およびアカウンティングのための利用情報の取得を行う．その名前から連想されるいわゆる仮想計算機機構を用いたものではないことに注意されたい．

- **Information Service(IS)**
  IS は情報を収集し，提供する枠組みで，WSRF をベースとしたデータベースインターフェイスである資源情報は標準化団体 DMTF[8] で定義されている CIM[9] スキーマで記述されバックエンドのリレーショナルデータベースに収められる．IS への呼び出しアクセスは，OGF で標準化されているデータベースアクセスプロトコルである OGSA-DAI[10] によって行う．情報のアップデートは LRPS(Local Resource Provider Service) と呼ばれる Web Service が行う．

4. gLite の概要

gLite は EGEE (Enabling Grids for E-Science in Europe) が開発中のグリッドミドルウェアである．gLite は主に以下のモジュールから構成される．

- **WMS (Workload Management System)**
  WMS はブローカに相当するモジュールである．クライアントからのリクエストを受け取り後述する BDII からの情報に基づきブローカリングを行う．ブローカリングには，Condor Project[11] で開発された ClassAd[13] によるマッチメイキングが用いられている．
- **BDII (Berkeley Directory Information Index)**
  情報サービスに相当するモジュールである．LDAP(Lightweight Data Access Protocol) プロトコルでアクセスが可能となっている．保持される資源データのスキーマには GLUE スキーマ [14] を用いている．
- **CE (Computing Element)**
  ジョブマネージャに相当するモジュールである．

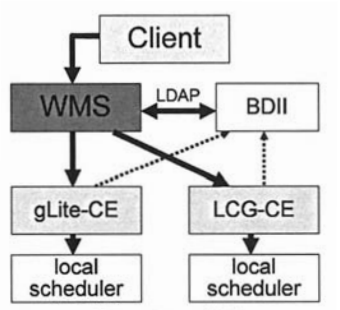

図 5 gLite の概要

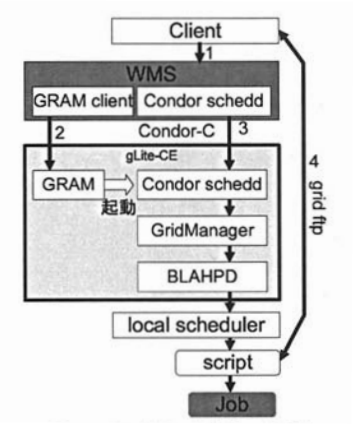

図 6 gLiteCE によるジョブ実行

gLite には LCG-CE と gLite-CE の 2 つの CE 実装が存在し，これらはまったく実装が異なる．LCG-CE は，EGEE に選考する LCG(LHC Computing Grid) プロジェクトからキャリーオーバされたモジュールで，その実体は Globus の pre-WS GRAM[15] である．これに対して，gLite-CE は Globus GRAM と Condor を併用した複雑なモジュールとなっている．

gLite CE は，ジョブ実行のプロトコルに Condor-C と呼ばれるプロトコルを用いる．Condor-C は，Condor から他の Condor にジョブをサブミットする機構である．gLite CE は Condor の schedd と呼ばれるデーモンでジョブをクライアントから受け取り，GridManager, 後述する BLAHPD を経由してローカルなバッチキューイングシステムに投入する．前述のとおり，gLite-CE では pre-WS GRAM が併用されるが，GRAM プロトコルは各ユーザごとに Condor schedd を起動するためのみに使用される．

BLAHPD は，さまざまなローカルバッチキューイングシステムのインターフェイスを抽象化するためのモジュールで，簡単なテキストベースのインターフェイスを持つ．このモジュールを変更することで，さまざまなローカルキューイングシステムに対応すること

ができる．

gLite CE に関しては特筆するべき点が 2 点ある．ひとつは，CE 上でユーザのために実行されるデーモン類やジョブの実行ユーザが，仮想ユーザとなることである．gLite CE では，個々のユーザのための個別のユーザ ID を設けず，仮想ユーザのプールを保持する．ユーザジョブに対して，実行のたびに仮想ユーザに割り当てられる．

もう一点は，schedd を介してキューイングシステムに投入されるスクリプトはユーザがサブミットしたジョブそのものではないことである．このスクリプトはクライアントが自動的に生成するもので，クライアントから実行ファイルを含むジョブの実行に必要なファイル群を gridFTP を用いてダウンロードし，実行し，結果をアップロードするように記述されている．

実行の様子を図 6 に示す．ユーザは gLite のクライアントノードからサブミットコマンドを用いて，WMS に対してジョブをサブミットする (1)．WMS はジョブを実行するサイトを決定し，pre-WS GRAM プロトコルを用いてそのサイト上に Condor schedd を起動する (2)．さらに，Condor-C プロトコルで起動した schedd に対してスクリプトを投入する．スクリプトは，GridManager，BLAHPD を経由してローカルキューイングシステムに投入される．スクリプトは，gridFTP を用いて，クライアントから実行ファイルと入力ファイルをダウンロードし，実行ファイルを実行し，結果をクライアントにアップロードする (4)．

5. 相互運用試験の方針

5.1 セキュリティ基盤の相互運用

セキュリティ基盤の相互運用性は，相互運用の根底をなすものであり，これが成立しなければ，他の相互運用性を確保することはできない．幸いにも gLite と NAREGI ミドルウェア β はセキュリティ基盤を共有しており，今回の実験においては，特に問題になることはなかった．gLite と NAREGI ミドルウェア β は双方とも，証明書発行機関を用いた PKI による認証を行うが，双方の証明書発行機関が IGTF(International Grid Trust Federation) 参加の PMA(Policy Management Authority) に属していることから，証明書の相互運用に問題はない．

次に問題になりうるのは，VO(Virtual Organization) の管理であるが，NAREGI ミドルウェア β は gLite の VOMS(Virtual Organization Membership Server) を流用していることから，この点でも問題は生じなかった．

5.2 情報サービスの相互運用

NAREGI ミドルウェア β と gLite の情報サービスは相互に異なる資源データフォーマットと転送プロトコルを利用しているため，情報サービスの相互運用を

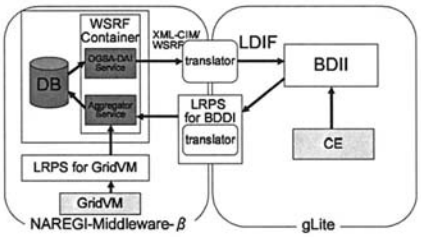

図 7 情報サービスの相互運用

実現するためには，データフォーマット変換と転送プロトコルの中継を行うモジュールが必要となる．我々はこのために 2 つのモジュールを開発した (図 7 中央)．NAREGI ミドルウェア β IS から情報を取り出して変換し，gLite BDII へストアするモジュールと，その逆を行うモジュールである．

前者は，独立したモジュールとして実装されている．定期的に起動し，OGSA-DAI プロトコルを用いて，IS から CIM フォーマットで情報を取得し，これを GLUE スキーマ情報に変換，LDAP で用いられる LDIF 形式にフォーマットし，LDAP プロトコルで BDII へストアする．

後者は，NAREGI ミドルウェア β IS を構成するサブモジュールである LRPS のひとつとして実装されている．同様に定期的に起動し，BDII から GLUE スキーマに従って記述された情報を LDAP プロトコルで取得し，CIM に変換する．データの更新があった場合，LRPS は IS の一部である Aggregator Service に notification を送信する．Aggregator Service はこれに答えて，LRPS に情報取得メッセージを送信し，CIM 形式の情報を取得，バックエンドのデータベースに格納する．

5.3 ジョブサブミッションの相互運用の方針

ジョブサブミッションの相互運用の実現には，2 にしめした 2 つの方針がありうる．我々は 2 つのミドルウェアの実装法を考慮し，NAREGI ミドルウェア β から gLite に関してはジョブマネージャレベルの相互運用を，gLite から NAREGI ミドルウェア β に関してはブローカレベルの相互運用を実現した．

6. NAREGI ミドルウェア β から gLite へのジョブサブミッション

前述したとおり，NAREGI ミドルウェア β から gLite へのジョブサブミッションに関しては，ジョブマネージャレベルで実現した．これは，gLite ではジョブマネージャレベルに pre-ws GRAM という広く知られたプロトコルが用いられているため，実装が容易なためである．これは，OGF GIN で現在推奨されている方針にも合致する．**??**に実装の概念図を示す．SS から GridVM に至るパスと並行して LCG-CE(pre-WS

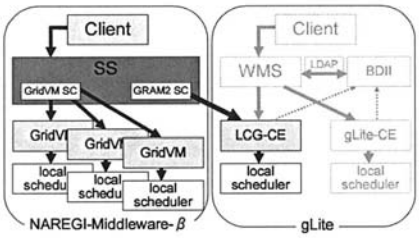

図 8 NAREGI ミドルウェアβから gLite へのジョブサブミッション

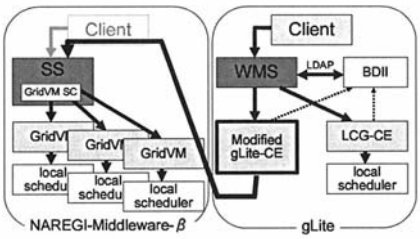

図 9 gLite から NAREGI ミドルウェアβへのジョブサブミッション

GRAM) へ至るパスが設けられている．

SS 内部には，SC (Service Container) と呼ばれるモジュールがあり，実際に使用される外部モジュールを抽象化している．GridVM を用いる際には，GridVM 向けの SC を経由して実際の動作が行われる．今回我々は，LCG-CE を用いるために，LCG-CE(pre-WS GRAM) に対応した SC を開発した．

SS は各ホストに対して LCG-CE SC を用いるか GridVM SC を用いるかを判断する必要がある．このための情報は，前述した情報サービスの相互運用によって IS に格納される．NAREGI ミドルウェアβでは下位のモジュールが，事前予約に対応していることを前提に，すべてのジョブを予約して実行する．しかし LCG-CE には事前予約機能はない．このためわれわれは，LCG-CE 向け SC に擬似予約機構を実装した．LCG-CE 向け SC は，事前予約リクエストに対して擬似的に予約が成功したように返答する．これによって，SS の主要部分を変更することなく相互運用を実現することができた．

7. gLite から NAREGI ミドルウェアβへのジョブサブミッション

7.1 設　　計

gLite のジョブサブミッション経路中，Condor schedd から BLAHP を経由する部分が，容易に改変可能に設計されている．われわれはこの部分を改変し，NAREGI ミドルウェアβへの呼び出しを実現した．具体的には NAREGI ミドルウェアβへのジョブ投入を行う BLAHPD を実装し，ローカルキューイングシステムへジョブをキューイングする代わりに NAREGI ミドルウェアβにジョブを投入するよう変更した．

セキュリティに関しては，gLite が委譲する証明書を利用して NAREGI ミドルウェアβへのジョブ投入を行う．Condor-C はプロキシ証明書をファイルとして準備し，環境変数にそのファイル名を設定し，BLAHPD に引渡す．BLAHPD はこのファイルから証明書をとりだし，再度 MyProxy に登録，それを利用して NAREGI ミドルウェアβにジョブを投入する．

7.2 NAREGI BLAHPD の実装

BLAHP プロトコルは，Condor から Globus などの他のグリッドミドルウェアを利用するために考案されたテキストベースのプロトコルである GAHP[12] の一種である．GAHP がマーシャリング手法やリクエストに対する応答などの大枠のみを定めているのに対して，BLAHP は具体的なコマンドのセットを定めている点が異なる．

BLAHP のコマンドセットは，Condor から UNICORE を呼び出すための UNICORE GAHP サーバ[16] のコマンド集合と類似している．このためわれわれは Java で記述された UNICORE GAHP サーバのソースコードを改変することで NAREGI ミドルウェアβ向けの BLAHPD を作成した．NAREGI ミドルウェアβへのジョブサブミッションには NAREGI ミドルウェアβの Java インターフェイスを利用した．

7.3 実装上の問題点

前述したとおり，gLite CE 上での UNIX ユーザは，仮想的なユーザとなる．これに対して，NAREGI ミドルウェアβのクライアントモジュールは，実際のユーザが使用することを仮定している．このため，NAREGI ミドルウェアβに対してジョブがサブミットされ，実行ファイル (ここでは gLite が送付したスクリプト) をステージする際に，問題が生じる．このファイルは，仮想的なユーザの所有物であるが，これに対して NAREGI ミドルウェアβが本来のユーザのプロキシ証明書で GridFTP を試みるためである．

我々は，すべてのユーザがアクセス可能な一時ディレクトリを作成し，そこにスクリプトをコピーすることでこの問題を解決した．一般に，ファイルをすべてのユーザに対してアクセス可能にすることはセキュリティ上好ましくないが，このスクリプトは gLite が用意するものでユーザジョブ情報は含まないため，この場合には問題はない．

8. 実　　験

作成した相互運用モジュールの動作を確認するために，簡単な実験を行った．実験は，NAREGI に設置した環境で行った．gLite と NAREGI ミドルウェアβ

表 1 実行時間 (秒)

Client \ Server	gLite-CE	LCG-CE	NAREGI
gLite	250	284	487
NAREGI	N/A	134	66

をインストールした計算機群をそれぞれ用意し，それらの間で相互にジョブをサブミットする実験を行った．それぞれのミドルウェアの各モジュールは個別の計算機にインストールした．使用した計算機は，Pentium 4 Xeon 3GHz dual, メモリ 1Gbyte, RedHat 8 で，100base/TX のネットワークで接続されている．gLite のバッチキューイングシステムとしては TORQUE を，NAREGI ミドルウェア β のバッチキューイングシステムとしては PBS Professional を用いた．

表 1 に結果を示す．対照のため，クライアントとサーバが双方 NAREGI ミドルウェア β，gLite である場合も実験を行った．gLite に関しては，LCG-CE を用いた場合と，gLite-CE を用いた場合を実験はそれぞれ 10 回行い平均値を示した．ジョブは hostname を実行するだけのもので，ジョブ自体にかかる時間は無視できる．また，NAREGI ミドルウェア β も gLite も，ジョブ実行の終了を検出するためにはポーリングが必要であり，そのインターバルを 10 秒としたため，この結果には 10 秒以内の誤差がある．

gLite から NAREGI ミドルウェア β を呼び出すケースが特に時間がかかっていることがわかる．これは，この場合のみ gLite と NAREGI ミドルウェア β のブローカリングシステムの双方を使用するため，それぞれのオーバヘッドが加算されているからだと思われる．

9. おわりに

NAREGI ミドルウェア β と gLite の間でジョブ起動に関する相互運用実験を行った．実験の結果，以下を確認した．1) 証明書や仮想組織管理のレイヤでは相互運用性に問題はない，2) 情報サービスのレイヤでも相違が吸収できる，3) その情報を用いて相互のジョブ起動が可能である，

今後の課題には以下が挙げられる．

- 今回の実験は，NAREGI 内ローカルで行ったものであり，実環境のレイテンシやジョブ負荷が相互運用モジュールに与える影響はテストできていない．今後は実際に日欧間での実験を行う必要がある．
- gLite の実行機構は，実際にジョブを実行する計算機上に，gLite のライブラリがインストールされている事を仮定する．今回の実験では，NAREGI 環境の実行ノード上に gLite のライブラリをインストールしたが，このような制約は，実際に相互運用を行う場合には問題になる可能性がある．この制約を回避する実装方法を検討する必要がある．

謝　辞

本研究は文部科学省「経済活性化のための重点技術開発プロジェクト」の一環として実施している超高速コンピュータ網形成プロジェクト (NAREGI: National Research Grid Initiative) によるものである．

参考文献

1) Open Grid Forum. http://www.ogf.org.
2) Open Grid Forum Grid Interoperation Now Community Group. https://forge.gridforum.org/sf/projects/gin.
3) EGEE: Enabling Grids for E-SciencE. http://www.eu-egee.org/.
4) gLite: Lightweight Middleware for Grid Computing. http://glite.web.cern.ch/glite/.
5) EGEE Middleware Architecture and Planning, Technical Report DJRA1.4, EU Deliverables.
6) Matsuoka, S., Hatanaka, M., Nakano, Y., Iguchi, Y., Ohno, T., Saga, K. and Nakada, H.: Design and Implementation of NAREGI Super-Scheduler Based on the OGSA Architecture, Vol. 21, No. 4, pp. 521–528 (2006).
7) Foster, I.: Globus Toolkit Version 4: Software for Service-Oriented Systems, *IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779*, pp. 2–13 (2005).
8) Distributed Management Task Force. http://www.dmtf.org/search.
9) Common Information Model. http://www.dmtf.org/standards/cim/.
10) The OGSA-DAI Project. http://www.ogsadai.org.uk/.
11) Condor. http://www.cs.wisc.edu/condor/.
12) Globus ASCII Helper Protocol. http://www.cs.wisc.edu/condor/gahp.
13) Raman, R., et al.: Matchmaking: Distributed Resource Management for High Throughput Computing, *Proc. of HPDC-7* (1998).
14) GLUE Schema. http://glueschema.forge.cnaf.infn.it/.
15) Czajkowski, K., et al. : A Resource Management Architecture for Metacomputing Systems, *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing* (1998).
16) Nakada, H., Frey, J., Yamada, M., Itou, Y., Nakano, Y. and Matsuoka, S.: "Design and Implementation of Condor-UNICORE Bridge", *"Proceedings of HPC ASIA 2005"*, pp. 307–314 (2005).