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

2004－ARC－157　(7)
2004－HPC－ 97　(7)
2004／3／1

Condorの汎用グリッドインターフェイスの設計とUNICOREへの適用

中田　秀基 [†1,†4]　Jaime Frey [†2]　山田　基弘 [†3]
伊藤　泰善 [†3]　中野　恭成 [†3]　松岡　聡 [†4,†5]

本稿では、Condorシステムにおける、汎用の外部グリッドシステムへのインターフェイス機構の設計および実装に関して述べる。Condorシステムは特定の外部グリッドシステムに対するインターフェイスを持っているが、これを使用するにはCondorシステム内部の変更が必要になり、新たな外部グリッドシステムのサポートを追加することが難しかった。われわれは、任意の外部グリッドシステムへのインターフェイスを容易に構築するための汎用インターフェイスを設計し、この問題を解決した。さらに、この汎用インターフェイスを用いて、UNICOREシステムを外部グリッドシステムとして利用するブリッジを実装し、汎用インターフェイスによる手法の有効性を確認した。

Design and implementation of Condor-UNICORE bridge

Hidemoto Nakada [†1,†4]　Jaime Frey [†2]　Motohiro Yamada [†3]
Yasuyoshi Itou [†3]　Yasumasa Nakano [†3]
and Satoshi Matsuoka [†4,†5]

In this paper, we describe design and implementation of a Generic Grid interface for Condor. Though Condor has intefaces for specific Grid systems, such as Globus GRAM, it is not easy to add new interface for other Grid systems, since it will require some code modification inside the Condor. With our new interface, supporting a new Grid system can be established without any code modification in Condor itself. We also implemented a bridge for UNICORE system and validated that our approach is effective.

1. はじめに

Condor[1]はウィスコンシン大学で開発された、スケジューリングシステムで、キャンパス内での遊休資源の有効利用による、ハイスループットコンピューティングを指向している。Condorは通常のジョブ起動方法以外に、Globus toolkit[2]のGRAM gatekeeperをはじめとするいくつかのジョブ起動システムに対するインターフェイス機構を持っており、これらのグリッドシステムを経由してジョブを起動することができる。しかし、このインターフェイス機構は個々のグリッドシステムに応じてCondorのシステム内部に作りこまれており、新たなグリッドシステムへの対応を行うには、Condorの内部を変更する必要があり煩雑である。

われわれは、Condorシステムに外部のジョブ起動システムに対する汎用のインターフェイスを追加した。さらに、外部ジョブ起動システムとしてUNICORE[3),4)]をターゲットとし、このインターフェイス機構を利用したブリッジを作成した。この結果、汎用インターフェイスの有効性を確認した。

本稿の構成は次のとおりである。2節と3節で, 本稿の対象システムであるCondorとUNICOREについて概要を述べる。4節で汎用グリッドインターフェイスの設計を説明する。5節でこのグリッドインターフェイスを用いたUNICOREへのブリッジを詳述する。6節でまとめと今後の課題を述べる。

2. Condorの概要

Condorは、米国ウィスコンシン大学のグループが開発したハイスループットコンピューティングを指向したキューイングシステムである。当初はキャンパス内の遊休計算機を有効利用することを目的としていたが、現在では後述するCondor-Gの機能により、Globusのメタスケジューラとしても広く使用されている。

Condorは、計算可能な計算機の集合をCondorプールと呼び、ユーザがサブミットしたジョブに対して、Condorプールに属する計算機に割り当て、実行する(図1)。このジョブに対して計算機を割り当てる方法に、マッチメイキングと呼ばれる方法が使用される。

†1 産業技術総合研究所　National Institute of Advanced Industrial Science and Technology (AIST)
†2 ウィスコンシン大学　University Wisconsin
†3 富士通株式会社　Fujitsu Limited
†4 東京工業大学　Tokyo Institute of Technology
†5 国立情報学研究所　National Institute of Information

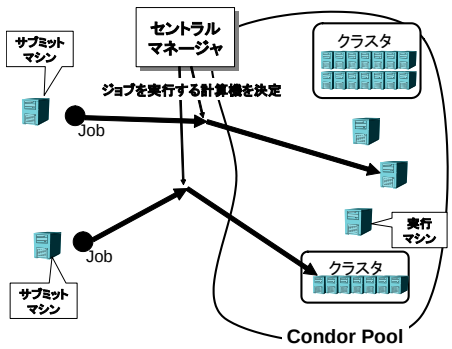

図 1 Condor の概要

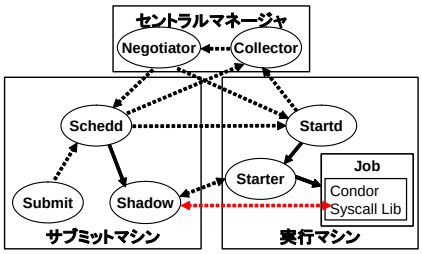

図 2 Condor のデーモン

さらに、計算が適宜チェックポイントされ、サーバフェイル時の再実行や優先順位の高いジョブによるプリエンプションがサポートされていることが、Condor の特徴である。

2.1 Condor によるジョブの実行

図 2 に condor システムを構成するデーモンの相関図を示す。ユーザは Condor のシステムコールライブラリとリンクされたジョブを用意する。次にジョブを記述したサブミットファイルを用意し、コマンドを用いてサブミットマシンの Schedd(Schduling Daemon) に投入する。Schedd は定期的にサブミットされたジョブの情報をセントラルマネージャ の Collector に送信する。実行マシンでは、Startd が自らの状態をモニタし、定期的に Collector に報告する。

Central Manager では、Collector に集まった情報を用いて、定期的に、ジョブと実行マシンの「相性」を判定し、相性がよい組み合わせで実行を行うように、サブミットマシンと実行マシンに指令する。相性を判定する過程はマッチメイキングと呼ばれる。

Schedd は、Startd に対してジョブを送信し、実行を依頼する。実行マシン上で起動したジョブは、システムコールライブラリによって適宜チェックポイントがとられる。

2.2 ClassAd

Condor では上述のマッチメイキングを、ClassAd と呼ぶデータ構造に基づいて行う。Schedd や Startd が Collector に送信する情報はすべて ClassAd で行われる。ClassAd は属性に対する制約を記述する記述形式で、ユーザによるアドホックな拡張が可能であることが特徴である。ClassAd を用いることによって、柔軟なマッチメイキングが可能となっている。

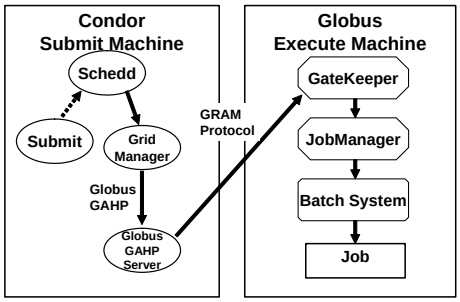

図 3 Condor-G によるジョブの起動

2.3 Condor-G

Condor は、 Globus Toolkit のジョブ起動サービスである GRAM を用いて管理されているホストに対して、ジョブを起動する機能を持っている。この機能を Condor-G[5] と呼ぶ。

Condor-G によるジョブ起動の様子を図 3 に示す。Condor-G によるジョブ起動は、通常のジョブ起動の方法とまったく異なり、Schedd が主導する。まず、Schedd は GridManager と呼ぶプロセスを起動し、このプロセスにジョブの起動を依頼する。GridManager はさらに、実際に Globus のプロトコルを実行する Globus-GAHP サーバと呼ばれるプロセスを起動する。GirdManager と Globus-GAHP サーバは後述する GAHP[6] と呼ばれるプロトコルで通信を行う。

Globus-GAHP サーバは、GRAM の gatekeeper と通信を行い、ジョブの起動を依頼する。Gatekeeper は jobmanager を経由してバッチシステムにジョブをサブミットし、バッチシステムがジョブを実際に起動する。

2.4 GAHP の概要

GAHP(Grid ASCII Helper Protocol) は、Condor の内部で用いられているテキストベースのプロトコルの枠組みである。

GAHP では、クライアントとサーバの間で、1 行ずつの文字列を交換する。クライアントからサーバへは **Request Line** が送られ、サーバからは、**Return Line** と **Result Line** が送られる。Return Line は、Request Line が有効であるかどうかを示すもので、Request Line のパーズが完了すると即座に返送される。これに対して、Result Line は、Request の処理が終わった際にリクエストとは非同期に返却される (図 4)。Request Line と Result Line の対応をつけるために ID を用いる。Request Line にはクライアントが定めた ID が付与される。サーバはこの ID を Result Line に含めて返送する。

各 Line は スペースで分割された文字列コマンドから構成され、**CR LF** で終端される。文字列コマンド

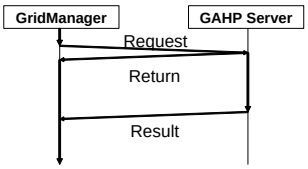

図 4 GAHP のメッセージパッシング

自身にスペースを含む場合には、スペースをバックスラッシュでエスケープする。バックスラッシュ自身もバックスラッシュでエスケープする。

GAHP 自身はプロトコルのフレームワークであり、実際に使用されるコマンドのセットを規定しない。Globus-GAHP では、Globus 用のコマンドセットを定義し、それを使用している。

2.5 Condor GlideIn

Condor-G によって、Globus で管理されている資源に対しても Condor からのジョブ起動が可能になるが、そこで使用できる Condor の機能は非常に限定されたものになる。Central Manager によるスケジューリング機能は限定的になり、リモートシステムコールやチェックポインティングなどはサポートされない。

この問題を解決する手法として Condor GlideIn が実装されている。これは、Condor-G の仕掛けを利用して Condor の実行マシンの機能をサーバ側に起動し、Globus の資源を正式な Condor プールの一部にしてしまうという方法である。実際に計算を行うジョブは、改めて通常の経路でサブミットされることになる。

この方法は Globus のジョブが起動される実行マシンが、グローバルアドレスを持つ場合にしか機能しないという問題点があるが、スケジューリングやチェックポイントを利用できるため非常に有用である。

3. UNICORE の概要

UNICORE は、欧州富士通研究所が中心となって開発したグリッドミドルウェアである。UNICORE は、スーパーコンピューティングセンターをのスーパーコンピュータを、遠隔地からシームレスに利用することを目的に設計されており、この使用法に対する要請から、ファイアウォールに対する配慮が十分になされている。

UNICORE の概要を図 5 に示す。UNICORE では、ファイアウォールに囲まれたサイトを Usite(UNICORE Site) と呼ぶ単位で抽象化している。各 Usite の中には、実際に計算を行う Vsite(Virtual Site) と呼ばれる単位が 1 つ以上存在する。Usite がスーパーコンピュータセンタに、Vsite がスーパーコンピュータや、計算機クラスタなどの計算システムに相当する。

UNICORE ではユーザのサブミットするジョブが、単一のタスクではなく、複数の依存関係を持つタスクからなるワークフローとなっている。このワークフローは Java のオブジェクトとして記述されており、このオブジェクトをシリアライズしたものをモジュール間でやり取りすることで、ジョブのサブミッションを行う。このワークフローを表現したオブジェクトを AJO(Abstract Job Object) と呼ぶ。

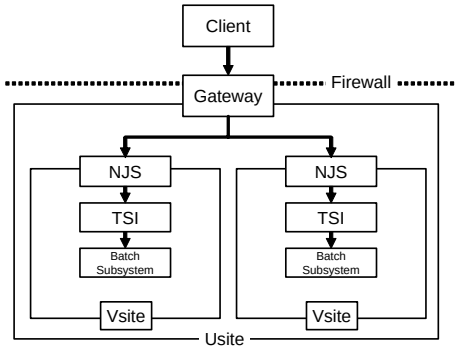

図 5 UNICORE の概要

各 Usite には、Gateway と呼ばれるデーモンプロセスが 1 つ存在する。この Gateway が外部からの通信すべてを中継する。このため、Gateway だけを Fire-Wall 上に露出しておくだけで、Usite に含まれるすべての Vsite の計算機資源を外部から使用可能になる。

Usite には 1 つ以上の Vsite があり、各 Vsite には NJS(Network Job Supervisor) と呼ばれるデーモンが存在する。NJS は実際にタスクを実行するモジュールで、ワークフローを解釈するワークフローエンジンであると同時に、後述の TSI と連動して、各タスクを処理するエンジンとしての役割も果たす。

TSI(Target System Interface) は、バッチサブシステムと NJS の間を取り持つモジュールである。バッチサブシステムとしては、PBS や LSF などのバッチジョブキューイングシステムの他、Fork による直接の起動も使用できる。

4. 汎用グリッド インターフェイスの設計

4.1 Condor-G インターフェイスの問題点

前節で述べたとおり、Condor-G では GridManager と GAHP サーバが分離されており、Globus のプロトコルを実際に処理する部分は GAHP サーバに分離されている。したがって、理論的には、他のグリッドシステムでのジョブサブミッションを可能にするためには、Globus-GAHP サーバ部のみを他のグリッドシステム向けに書き直せばよい。

ここで問題になるのは、Globus-GAHP のプロトコルである。Globus-GAHP のコマンドセットのセマンティクスは、Globus の GRAM API のセマンティクスに緊密に束縛されている。実際に Globus と通信するのは GAHP サーバであるが、API のセマンティクスをハンドルするロジックは GridManager に収めら

表 1 汎用 GAHP コマンドセット

名称	JOB_CREATE
Request	`UNICORE_JOB_CREATE <req id> <job classad>`
Return	`<S|E>`
Result	`<req id> <S|F> <job handle> <error string>`
意味	ジョブを作成し、そのジョブに対するハンドルを返す。ジョブの作成に必要な情報は ClassAd で与えられる。
名称	JOB_START
Request	`UNICORE_JOB_START <req id> <job handle>`
Return	`<S|E>`
Result	`<req id> <S|F> <error string>`
意味	指定されたジョブハンドルの指すジョブを起動し、その成否を返す。
名称	JOB_STATUS
Request	`UNICORE_JOB_STATUS <req id> <job handle>`
Return	`<S|E>`
Result	`<req id> <S|F> <result classad> \ <error string>`
意味	指定されたジョブハンドルの指すジョブのステイタスを返す。ステイタスは ClassAd として表現される。
名称	JOB_DESTROY
Request	`UNICORE_JOB_DESTROY <req id> <job handle>`
Return	`<S|E>`
Result	`<req id> <S|F> <error string>`
意味	指定されたジョブハンドルの指すジョブを破棄する。この際 GAHP サーバ内のジョブ関連情報だけではなく、対象グリッドシステム内部の情報も破棄する。

れているのである。このため、このコマンドセットを維持したまま他のグリッドシステムに適用することは非常に難しい。そもそも、外部グリッドシステムのロジックを GridManager に持つことは望ましいとはいえない。新たにグリッドシステムを追加するたびに、比較的複雑な構造を持つ GridManager に手を入れなければならないからである。

4.2 汎用 GAHP コマンドセット

上に示した問題点を解決するために、すべてのグリッドシステムを統一的に扱う枠組みとして、われわれは汎用の GAHP コマンドセットを定義した。汎用 GAHP コマンドセットはグリッドシステムに対してニュートラルに設計されているため、GAHP サーバ側を書き直すだけでさまざまなグリッドシステムに対応することができる。

表 1 に、汎用 GAHP コマンドセットを示す。表中の `<s|F>`は 文字列 `s` もしくは `F` を示す。`<s|E>` も同様である。S は Success を、F は Failure を、E は Error を示す。コマンド名に UNICORE という名前が現れているものの、コマンドセットの設計は UNICORE のプロトコルにはまったく依存していないことに注意されたい。

汎用コマンドセットは、ジョブの生成、起動、状態検出、破棄の 4 つのコマンドからなる。ジョブ生成

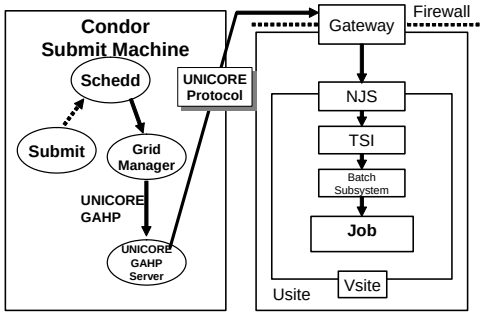

図 6 UNICORE ブリッジを用いたジョブ起動

時のジョブ記述やジョブの状態としては、Condor の ClassAd を使用する。この理由としては、Condor 内部で ClassAd が用いられているため既存部分との親和性が高い、入出力ライブラリが用意されているため実装が容易、任意の属性を追加できるため拡張性が高い、が挙げられる。ClassAd には XML 表記と独自のテキスト表記の 2 つの表現方法があるが、われわれは XML 表記を採用した。

5. 汎用 GAHP コマンドを用いた UNICORE ブリッジの実装

汎用 GAHP コマンドを用いて、他グリッドシステムへのブリッジを作成するには、まず、ジョブサブミッションに用いる ClassAd の属性値を定義し、その ClassAd を解釈して、実際に他のグリッドシステムにジョブを発行する GAHP サーバを作成すればよい。

GAHP プロトコルはテキストフォーマットであるため、GAHP サーバは実装する言語を選ばない。UNICORE のプロトコルは Java に強く依存しているため、本研究では、UNICORE GAHP サーバを Java で実装した。

UNICORE ブリッジを用いた Condor からのジョブ起動の様子を図 6 に示す。

5.1 ClassAd の設計

ClassAd には、ジョブの実行と情報の伝達に十分なだけの情報が含まれていなければならない。表 2 に UNICORE GAHP で使用する ClassAd の属性を示す。上段は Condor が標準で使用している属性、下段が今回用意した属性である。

UNICORE-GAHP サーバはジョブのステイタスを GridManager に返さなければならない。しかし、UNICORE のジョブステイタスは 3 層からなる比較的複雑な構造をとっているのに対し、Condor のジョブステイタスは 6 つの状態からなる単純な構造で、セマンティクスにギャップがある。UNICORE-GAHP サーバは、UNICORE のジョブステイタスを表 3 に示すマッピングにしたがって、Condor のジョブステイタスにマップし、GridManager に返している。

表 2　Job classAd の属性値

属性	意味	例
Cmd	実行ファイルのパス名	/home/foo/a.exe
Args	実行ファイルに渡す起動時の引数	arg1, -a
Env	実行時にセットする環境変数	LANG=en_US
IWD	クライアント側のベースディレクトリ	/home/nakada/condor
In	標準入力に与える入力ファイル	input.dat
Out	標準出力からの出力を収めるファイル	ouput.dat
Err	標準エラーからの出力を収めるファイル	Error.dat
TransferInput	ステージインするファイル名	a.exe, input.dat
TransferOutput	ステージアウトするファイル名	out.dat
JobStatus	Condor におけるジョブのステイタス	表 2 第 2 カラム参照
ErrMessage	エラーメッセージ	Script reported no errors
RemoteWallClockTime	ジョブの実行時間	123.0
ByteSent	総送信バイト数	1023004
ByteRecvd	総受信バイト数	1023004
ExitBySignal	実行プロセスがシグナルで終了したか	TRUE
ExitCode	実行プロセスの Exit コード	1
ExitSignal	実行プロセスの終了シグナル	9
上 Condor 汎用 / 下 UNICORE 依存		
UnicoreUsite	Unicore の Usite の gateway の FQDN とポート名	fujitsu.com:1234
UnicoreVsite	上記 Usite 内の Vsite の名前	NaReGI
KeystoreFile	キーストアファイル名	/home/foo/key
PassphraseFile	パスフレーズファイル名	/home/foo/passwd
UnicoreJobId	ハンドルとなるジョブの ID	fujitsu.com:1234/NaReGI/1374036929
UnicoreJobStatus	UNICORE におけるジョブのステイタス	表 2 第 3 カラム参照
UnicoreLog	UNICORE の log ファイルのファイル名	/var/log/unicore.log

表 3　ジョブステイタスのマッピング

Condor ジョブステイタス	コード	UNICORE ジョブステイタス
I (Idle)	1	CONSIGNED PENDING READY EXECUTING QUEUED SUSPENDED
R (Running)	2	RUNNING
X (Removed)	3	
C (Completed)	4	SUCCESSFUL FAILED_IN_EXECUTION KILLED FAILED_IN_INCARNATION FAILED_IN_CONSIGN NEVER_TAKEN
H (Held)	5	HELD
U (Unexpected)	0	

5.2　パスフレーズの管理

UNICORE ではクライアントとサーバ間の認証に X.509 形式の証明書を用いている。UNICORE GAHP サーバは UNICORE のクライアントとして機能するため、キーストアに格納された証明書にアクセスする。このキーストアを開くためにはパスフレーズが必要である。

GAHP サーバは、クライアントマシンがクラッシュした際などにも自動的に再起動できなければならない。この場合、ユーザにパスフレーズを入力させることはできないので、何らかの方法でパスフレーズを管理しなければならない。

本実装では、キーストアのパスフレーズをユーザアカウントでのみ読み出し可能なファイルに書き込むこととしている。GAHP サーバはこのパスフレーズファイルからパスフレーズを読み出し、それを用いてキーストアから証明書を取得する。

5.3　UNICORE プログラム実行の概要

5.3.1　サブミットファイルの記述

Condor 経由で UNICORE にジョブを投入する際には、通常の Condor システムの場合と同様に、Condor のサブミットファイルを記述する。この際に、UNICORE に特有の ClassAd 属性を書く必要がある。

サブミットファイルは、内容的には ClassAd と類似しているが、属性名は微妙に異なる。UNICORE 固有の ClassAd 属性は、Condor システムのサブミットファイル解釈部には認識されない。しかし、属性名の前に'+' をつけることで、ClassAd に直接反映されることができる。サブミットファイルの例を図 7 に示す☆。

5.3.2　ジョブのサブミット

ジョブのサブミットも通常の Condor と同様に

☆ Universe が Globus となっているが、これは歴史的経緯で、現在のところ GAHP サーバを経由する起動はすべて globus ユニバースに分類されているためである。この点は、今後修正される予定である

```
Universe   = globus
+SubUniverse = unicore
Executable = a.out
output     = tmpOut
error      = tmpErr
log        = tmp.log
+UnicoreUsite = fujitsu.com:1234
+UnicoreVsite = NaReGI
+KeystoreFile  = /home/foo/key
+PassphraseFile = /home/foo/passwd
Queue
```

図 7 サブミットファルの例

condor_q コマンドを用いて行う。この際、事前にパスフレーズファイルを用意する必要がある。

Schedd はジョブサブミットファイルを受け取ると、ClassAd に変換する。次に、GridManager を起動し、GridManager がさらに UNICORE GAHP サーバを起動する。GridManager は、UNICORE_JOB_CREATE コマンドでジョブを作成を指令する。

UNICORE GAHP サーバは、コマンドの引数として渡される ClassAd を解釈してパスフレーズファイルを取り出し、キーストアを開いて証明書を取り出す。今後 UNICORE GAHP サーバが UNICORE システムにアクセスする際には、この証明書を用いる。

次に、GAHP サーバはジョブを表現した ClassAd からそれに対応する AJO を作成する。この AJO の ID を用いて UNICORE の JobID を作成し、GridManager に返却する。

GridManager は、JobID を受け取ると、次に UNICORE_JOB_START コマンドで Job の実行を指令する。GAHP サーバは該当するジョブを Unicore システムにサブミットする。

5.3.3 ジョブの管理

UNICORE GAHP サーバは、UNICORE_JOB_STATUS コマンドを用いて定期的に UNICORE システムにアクセスし、ジョブのステイタスを取得する。ジョブの実行が終了していた場合には、ジョブの出力ファイルをクライアント側に転送する。

一方、 GridManager は、定期的に UNICORE GAHP サーバにアクセスし、ジョブのステイタスを取得する。ジョブが終了した場合、Schedd にジョブの終了を報告する。

最後に、UNICORE_JOB_DESTROY コマンドで、GAHP サーバ内の状態、および、UNICORE システム内の状態を破棄する。

5.3.4 クライアントクラッシュからのリカバリ

Condor はクライアントマシンがクラッシュし、リブートした際にもサーバ側で実行中のジョブを管理することができる。Schedd は実行中のジョブをファイルで管理しているため、リブート時にも、外部グリッドシステム上で実行中のジョブがあることを検出できる。Schedd は GridManager を再起動し、ジョブへの再接続を行う。GridManager は GAHP サーバを起動し、JobID を渡してジョブ Status の検出を行う。GAHP サーバはこの動作をトリガとして、外部グリッドシステムと接続して、ジョブとの再接続を行う。

6. おわりに

本稿では、Condor システムにおける、汎用の外部グリッドシステムへのインターフェイス機構の設計および実装に関して述べた。さらに、この汎用インターフェイスを用いて、UNICORE システムを外部グリッドシステムとして利用するブリッジを実装し、汎用インターフェイスによる手法の有効性を確認した。

今後の課題としては以下が挙げられる。

- パスフレーズ格納法の検討
  現在は、キーストアのパスフレーズはファイルに格納されているが、これはセキュリティ的には望ましい方法ではない。自動的な再起動が可能である、という要件を満たしつつより安全な手法を検討する。
- 汎用 GAHP コマンドの有効性の検証
  本稿で提案した汎用 GAHP コマンドを他のグリッドシステムに適用し、その有効性を確認する。

謝　辞

本研究の一部は文部科学省「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) Livny, M., Basney, J., Raman, R. and Tannenbaum, T.: Mechanisms for High Throughput Computing, *SPEEDUP Journal*, Vol. 11, No. 1 (1997).
2) Foster, I. and Kesselman, C.: Globus: A metacomputing infrastructure toolkit., *Proc. of Workshop on Environments and Tools, SIAM.* (1996).
3) Romberg, M.: The UNICORE Architecture - Seamless Access to Distributed Resources, *Proc. of 8th IEEE International Symposium on High Performance Distributed Computing (HPDC8)*, pp. 287–293 (1999).
4) UNICORE. http://www.unicore.org/.
5) Thain, D., Tannenbaum, T. and Livny, M.: Condor and the Grid, *Grid Computing: Making the Global Infrastructure a Reality* (Berman, F., Fox, G. and Hey, T.(eds.)), John Wiley & Sons Inc. (2002).
6) Globus ASCII Helper Protocol. http://www.cs.wisc.edu/condor/gahp/.