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

2006－ARC－167 (37)
2006－HPC－105 (37)
2006／3／1

事前予約機能を持つローカルスケジューリングシステムの設計と実装

中田秀基[†1,†2] 竹房あつ子[†1] 大久保克彦[†1,†3]
岸本誠[†1,†4] 工藤知宏[†1]
田中良夫[†1] 関口智嗣[†1]

グリッド上で複数の資源を同時に確保 (コアロケーション) するには，各サイトにおける事前予約が不可欠である．現在計算資源の多くでは，プライオリティと First Come First Served を組み合わせたスケジューリングポリシが用いられているが，このスケジューリングポリシと事前予約をどのように組み合わせるべきかに関しては，明らかになっていない．われわれは，この問題を検討する研究環境を整備することを目的とし，1) OpenPBS の亜種である TORQUE のスケジューラモジュールを記述するための API を整備し，2) これを用いて事前予約機能を持つスケジューラモジュールを実装した．さらに WSRF を用いた外部インターフェイスを実装し，Globus Toolkit Ver.4 の GRAM と連動したグリッド環境での予約と実行を実現した．

Design and Implementation of a Local Scheduling System with Advance Reservation

Hidemoto Nakada ,[†1,†2] Atsuko Takefusa ,[†1]
Katsuhiko Ookubo,[†1,†3] Makoto Kishimoto,[†1,†4]
Tomohiro Kudoh ,[†1] Yoshio Tanaka [†1] and Satoshi Sekiguchi [†1]

While advance reservation is an essential capability for co-allocating several resources on Grid environments, it is not obvious how it can be combined with priority-based First Come First Served scheduling, that is widely used as local scheduling policy today. To investigate this problem, we 1) developped Java API to implement scheduling modules for TORQUE, a variant of OpenPBS, 2) implemented a scheduler module that have advance reservation capability with the API. We also provide an external interface for the reservation capability based one WSRF. Using with job submission module from Globus toolkit 4, users can make reservation for resources and submit jobs over Grid.

1. はじめに

グリッドの使用目的のひとつとして，ネットワーク上に散在する複数の資源を同時に利用することで大規模な計算を行うことがある．グリッド上で計算機資源およびネットワーク資源を同時に確保 (コアロケーション) するには，各資源における事前予約が不可欠である．スーパスケジューラと呼ばれるグローバルなスケジューラが，各資源を管理するローカルスケジューラに対して事前予約を行うことによって，特定の時間帯においてすべての資源を特定のユーザの使用に対して同時に確保することで，すべての資源を利用した計算が可能になることを保証する (図 1)[1]．

†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 数理技研 SURIGIKEN Co., Ltd.
†4 エス・エフ・シー S.F.C Co., Ltd.

これまで行われてきた多くの大規模計算実験では，電子メイルなどの通信手段を用いて，人間が介在，調停することで資源の事前予約を行ってきた．しかし，実際に大規模計算資源としてグリッドを運用するためには，完全な自動化が必要なことは明らかである．このためには，現在広く用いられている，プライオリティと First Come First Served に基づくスケジューリングポリシと，事前予約を整合させたスケジューリングポリシを構築する必要がある．

この問題を研究するためのテストベッドとして，スケジューリングポリシを自由に改変でき，事前予約に対応したスケジューリングシステムが必要である．しかし，ソースが入手可能で改変したソースの再配布が可能なスケジューリングシステムにはこの条件を満たすものがない．

商用のスケジューリングシステム LSF[2] や PBS Professional[3] は予約機能を持つが，当然有償であるし，改変して再配布することはできない．Maui スケ

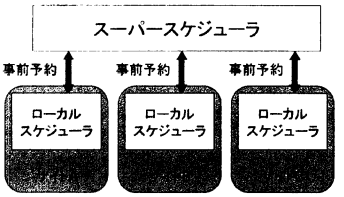

図 1 スーパースケジューラによる同時確保

ジューラ[4] は，OpenPBS[5] や TORQUE[6]，Grid Engine[7] に対してさまざまな機能を提供するスケジューラモジュールで，事前予約機能も提供している．しかし，Maui スケジューラはソースコードを公開しているものの，改変したソースを自由に配布することはできない．また，内部的にインターフェイスが整備されていないため，改変してスケジューリングポリシを変更することは困難である．

我々は，再配布可能なバッチスケジューリングシステム OpenPBS の亜種のひとつである TORQUE[6] を用いて予約可能なスケジューリング機構を実現した．具体的には，OpenPBS の持つスケジューラインターフェイスを利用する Java の API を作成し，この API を用いて予約機能を持つスケジューラモジュールを実装した．

さらに，この予約インターフェイスをサイト外部に対して公開するためのインターフェイスを，WSRF(Web Services Resource Framework)[8] に基づいて設計・実装した．このインターフェイスは Globus Toolkit Ver. 4[9],[10] のセキュリティ機構を用いた認証・認可を行う．このインターフェイスと Globus Toolkit 4 のジョブ起動機構 GRAM を併用することで，グリッド環境での資源予約と予約された資源でのジョブ実行を安全に行うことが可能になる．

本稿の以下の構成を以下に示す．2 節では，本稿の対象とするローカルスケジューラ TORQUE に関して概説する．3 節で，提案するスケジューラの設計と実装について述べる．4 節で WSRF による外部インターフェイスと Globus Toolkit 4 の GRAM との併用について述べる．5 節はまとめである．

2. TORQUE の概要

2.1 TORQUE の構造

TORQUE は，オープンソースのバッチスケジューリングシステムの一つである OpenPBS から派生した亜種のひとつである．OpenPBS の開発は事実上停止しているが，TORQUE は CLUSTER RESOURCES 社によって管理されており，最近になってメジャーバージョンアップも行われるなど，生きたプロジェクトである．ソースの改変や再配布も，ライセンス条項の制約のもと認められている．

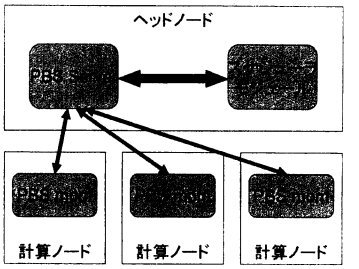

図 2 TORQUE の概要

TORQUE は大別して下記の 3 つのモジュールによって構成される (図 2)．

- PBS サーバ
  システムの中心となるモジュール．ひとつの PBS プールにひとつだけ置かれる．ユーザからのジョブキューイングリクエストを受け，キューを管理する．また，各計算ノード上の PBS mom と通信し，計算ノードのステイタスも管理する．
- スケジューラモジュール
  ジョブに対して計算資源を割り当てるモジュール．PBS サーバと対になる存在で，多くの場合同一計算機上に置かれる．PBS サーバ同様，ひとつの PBS プールにひとつだけ置かれる．
  PBS サーバは，ジョブサブミットや実行中のジョブの終了など，何らかのイベントが発生した際にスケジューラモジュール に対してスケジュールの依頼を行う．スケジューラモジュールは，この依頼をトリガとして PBS サーバに対してジョブや計算ノードのステイタスの問い合わせを行い，実行すべきジョブとノードを決定し，PBS サーバに，ジョブの実行を指示する．PBS サーバはこの指示に基づいて，ジョブの実行を行う．
- PBS mom
  各計算ノードを管理するモジュール．計算ノードの状態をモニタし，PBS サーバに報告する役割を担う．また，ジョブが割り当てられると，そのジョブを実際にプロセスとして起動し，プロセス状態を管理する．

2.2 TORQUE のユーザ認証

PBS サーバ はユーザからのリクエストを受けて動作を行うが，その際にリクエストを発したユーザを認証する必要がある．これは次の過程で行われる．

(1) ユーザがコマンドプログラム (qsub など) を実行する．
(2) コマンドプログラム は PBS サーバにソケット接続する．この段階では，このソケットは認証されていないため，このソケットを通じた操作は拒絶される．
(3) コマンドプログラム は root に set uid された pbs_iff と呼ばれるコマンドを fork, exec する．

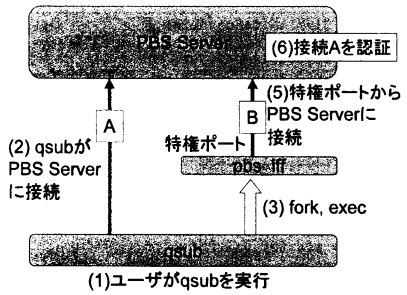

図 3 TORQUE の認証

この際に，PBS サーバに接続しているソケットのファイルディスクリプタ番号を指定する．

(4) pbs_iff は コマンドプログラムから引き継いだファイルディスクリプタのポート番号を調べる．さらに自プロセスの実ユーザ ID を見ることで，コマンドプログラムを起動したユーザのユーザ ID を検出する．

(5) pbs_iff は 1024 番未満の特権ポートから PBS サーバにコネクトし，PBS サーバに，コマンドプログラムがアクセスしているポート番号とユーザ名を通知する．

(6) PBS サーバは，特権ポートからのアクセスであることから pbs_iff を信頼し，通知されたポート番号が通知されたユーザからのアクセスであると判断する．

この様子を図 3 に示す．

2.3 TORQUE のカスタマイズ

TORQUE のスケジューラモジュールは，動作をある程度カスタマイズすることができる．BaSL(Batch Scheduling Language) と呼ばれる簡易言語でキューや各サーバの状態に応じてサーバへのジョブの割り当てを決定するなどの操作が可能となっている．BaSL による記述は比較的容易であるが，簡易言語であるため記述力に限界があり，複雑な操作を記述することができない．また，C 言語でスケジュールポリシを記述することもできるが，API が明示的に定義されていないため，実装の詳細を熟知していないかぎり，実際に記述することは困難である．

TORQUE ではスケジューラモジュールを交換することも可能である．PBS サーバとスケジューラ間の通信は，テキストベースのプロトコルで行われており，このプロトコルを解釈すれば，スケジューラモジュールを独自実装のものと挿げ替えることができる．Maui スケジューラはこのインターフェイスを用いて TORQUE との連携を実現している．

われわれは，Maui 同様にこのインターフェイスを利用して，独自実装のスケジューラモジュールを提供することとした．さらに，TORQUE からの状態取得と操作を行う API を提供することで，容易に独自のスケジューリングポリシを実装できる環境を整備した．

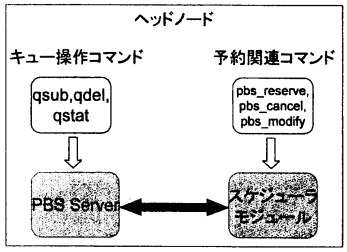

図 4 提案システムの概要

```
qsub -W x=rsvid:XXXXXXX
```

図 5 ジョブサブミット時の予約 ID 指定

3. スケジューラモジュールの設計と実装

3.1 提案システムの概要

本研究では，TORQUE のスケジューラモジュールを Java で実装するための API を整備し，この API を用いて事前予約機能を導入したスケジューラモジュールを実装する．スケジューラモジュールは，内部に予約テーブルを保持し，予約関連情報の管理を行う．予約テーブルには，予約に割り当てられたユニークな予約 ID と，予約開始時刻，予約終了時刻，ノード数，予約ノードを登録する．

qsub などのジョブの制御コマンドは PBS サーバと通信するが，予約関連のコマンドは，PBS サーバとではなくスケジューラモジュールと通信する必要がある．このためスケジューラモジュールに Java RMI による外部との通信インターフェイスを設け，予約関連コマンドとの通信を実現した．図 4 に提案システムの概要を示す．

スケジューラモジュールは，個々の予約に対してユニークな予約 ID を割り当てる．この予約 ID をサブミットコマンドのオプションに指定することによって，予約されたスロットでの実行を指定する (図 5)．サブミットコマンドで付加された予約 ID は，ジョブの付加情報としてスケジューラモジュールに渡される．スケジューラモジュールは予約 ID を用いて予約テーブルから対応する予約時間帯を検索し，その予約時間内であれば，ジョブを予約されているノードで実行する．

3.2 スケジューラ実装のための API の概要

Java によるスケジューラモジュール の実装を容易にするために，API を用意した．API の核となるインターフェイスである PBSInterface を図 6 に示す．このインターフェイスは，PBS サーバを抽象化したものであり，PBS サーバからの情報取得 (ノード情報，キュー情報など) や，PBS サーバへのジョブ操作 (ジョブの実行，削除，停止など) などの機能を提供する．

```
public interface PBSInterface{
   void   setSocket(Socket socket);
   Socket getSocket();
   void   authenticateUser(String userName,
                           int    localPort);
   void   disconnect();

   ServerStatus           statusServer();
   BatchReplyStatusNode   statusNode();
   BatchReplyStatusQueue  statusQueue();
   BatchReplyStatusSelect selectStatus(
                            String queueName);

   void runJob   (String jobId, String destination);
   void runJob   (String jobId,
                  Collection<NodeInfo> nodes);
   void deleteJob(String jobId);
   void holdJob  (String jobId, HoldJobType holdType);
   void rerunJob (String jobId);
   void modifyJob(String jobId,
                  String attr,
                  String value);
}
```

図 6 PBSInterface クラス

PBSInterface を用いた，簡単な FIFO スケジューラの実装例を図 7 に示す．提供 API を使用することで非常に簡単にスケジューリングアルゴリズムが記述できていることがわかる．

3.3 予約情報の永続化

予約テーブルに登録された情報は永続化する必要がある．PBS のヘッドノードをリブートしただけで予約情報が失われてしまっては困るからである．提案システムでは，永続化のために Java ネイティブのオブジェクトデータベースである db4objects[11] を用いた．db4objects は，非常に簡便なインターフェイスを提供しており，JDBC を用いて関係データベースをアクセスする方法と比較してはるかに容易に実装することができた．

3.4 予約コマンドとその認証

表 1 に予約に用いるコマンド群を示す．これらのコマンドは Java で記述された実体を呼び出すシェルスクリプトである．Java で記述されたコマンド本体は，Java RMI(Remote Method Invocation) でスケジューラと通信する．この際，スケジューラは予約をリクエストしてきたユーザを認証する必要がある．このために，接続の時点で認証を行うよう独自の ServerSocket と Socket を実装し，これらを用いるよう RMI の SocketFactory を変更した．

認証の技術としては，TORQUE の他の部分との相性を考慮し，2.2 で述べた pbs_iff を用いた手法を用いた．ただし，Java ではソケットのファイルディスクリプタ番号を取り出すことができないため，pbs_iff 相当のコマンドに渡す値をファイルディスクリプタ番号ではなくポート番号とした．

4. WSRF による外部インターフェイス

サイト間をまたがった予約を実現するには，サイト外部に対して予約インターフェイスを提供する必要

```
public class SimpleFifoScheduler {
  public static void main(String[] args) {
    // start scheduling server
    PBSServerConfig servConf = new PBSServerConfig();
    ScheduleStarter starter =
      new ScheduleStarter(servConf);
    PBSInterface pbs = new TorqueImpl();

    // get scheduling order, and run
    ScheduleOrder order;
    PBSSchedulerCommandType cmd =
      PBSSchedulerCommandType.NULL;
    do {
      order = starter.waitOrder();
      Socket socket = order.getPBSServerSocket();
      pbs.setSocket(socket);
      cmd = order.getSchedulerCommand();

      if (cmd.mustRunSchedule()) {
        try {
          schedule(pbs);
        } catch (PBSException e) {}
      }
      socket.close();
    } while (cmd != PBSSchedulerCommandType.QUIT);
  }

  private static void schedule(PBSInterface pbs)
          throws PBSException {
    ServerStatus server = pbs.statusServer();
    if (!server.isReadyToUse() ||
        server.getQueuedJobs() == 0)
      return; // no jobs to schedule

    Collection<NodeStatus> nodes =
      pbs.statusNode().getAllStatus();

    for (QueueStatus queue : pbs.statusQueue()) {
      if (!queue.isReadyToRun() ||
          queue.getQueuedJobs() == 0)
        continue;    // no jobs to run

      for (JobStatus job :
           pbs.selectStatus(queue.getName())) {
        if (!job.isReadyToRun())
          continue;    // cannot run now

        for (NodeStatus node : nodes) {
          if (!job.isRunnableOn(node))
            continue;    // node is down

          String jobId = job.getJobId();
          String destination = node.getName();
          pbs.modifyJob(jobId, "comment",
                        "Job started on " + new Date());
          pbs.runJob(jobId, destination);
          return;
        }
      }
    }
  }
}
```

図 7 提供クラスを用いた FIFO スケジューラ例

がある．われわれは，Grid 上のプログラミング環境のデファクトスタンダードとして広く用いられている Globus Toolkit 4 (GT4) を用いて，WSRF による外部インターフェイスを実現した．

WSRF は，Web サービス関連の標準化団体で標準化が行われている規格で，一般には状態を持たない Web サービスに対して，リソースと呼ばれる形で状態を導入する．GT4 は，WSRF の処理系とその上に実装されたいくつかのグリッド関連サービスの集合体である．実装されているグリッド関連サービスとしては，ジョブ起動のための GRAM4，情報サービスであ

表 1　予約関連コマンド

コマンド名	機能	引数	出力
pbs_reserve	予約をリクエストする	[-R スケジューラホスト名] -s 開始時刻 -e 終了時刻 -n ノード数	予約 ID
pbs_rsvcancel	予約をキャンセルする	[-R スケジューラホスト名] -r 予約 ID	
pbs_rsvmodify	予約の修正を行う	[-R スケジューラホスト名] -r 予約 ID [-s 開始時刻] [-e 終了時刻] [-n ノード数]	
pbs_rsvstatus	予約の状態を表示	[-R スケジューラホスト名] [-r 予約 ID]	予約状態

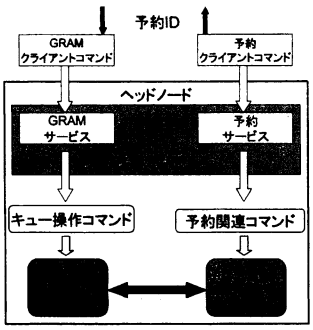

図 8　WSRF による外部インターフェイスと，GT4 との連携

る MDS4 などがある．

われわれは GT4 を用いて，WSRF に基づく予約インターフェイスを実装した．このインターフェイスと，GT4 のジョブ起動機構 GRAM を併用することで，外部からの安全な資源予約と，予約した資源へのジョブサブミットを実現することができる．

4.1　WSRF による予約インターフェイスの設計

WSRF においては一般に，サービスは対になる Factory サービスによって作成される．我々は，予約に対応する PBSReservation と，これを生成する PBSReservationFactoryService の二つのサービスを実装した．これらのサービスのオペレーションを表 2 に示す．

PBSReservationFactoryService は createPBSReservation オペレーションのみを持つ Factory サービスである．このオペレーションは，予約に必要な，開始時間，終了時間，ノード数などの情報を受け取り，PBSReservation サービスのインスタンスを生成し，それに対する参照である EPR (End Point Reference) を返却する．このとき，実際の予約はまだ行われていない．

PBSReservation サービスには 4 つのオペレーションがある．これらのオペレーションは，表 1 で示した予約コマンドに対応しており，サーバ側でそれぞれの予約コマンドを起動する．すべてのオペレーションは出力を持たない．これらのオペレーションは操作のトリガに過ぎず，操作の出力は後述するサービスのリソースプロパティに反映されるからである．

表 3 に PBSReservation のリソースプロパティを示す．オペレーションを実行すると，サーバ側で予約コマンドが実行される．コマンドの実行が終了すると，それぞれのプロパティの値が更新される．

表 3　PBSReservation のリソースプロパティ

リソースプロパティ名	意味
StartTime	予約開始時刻
EndTime	予約終了時刻
NodeNum	予約ノード数
Caller	証明書のグローバルユーザ名
LocalUsername	サーバ側でのユーザ名
ReserveId	予約 ID
ResultStatus	コマンド実行結果
ResultStdout	コマンド標準出力
ResultStderr	コマンド標準エラー
IsBusy	コマンド実行状態

4.2　予約インターフェイスの認証

予約インターフェイスでは，接続してきたクライアントのユーザを認証し，ローカルサイトのユーザにマップしなければならない．われわれはこれを，GT4 の提供する標準的な機能を用いて実現した．ユーザの認証には PKI に基づく証明書を用いた．証明書に書かれたグローバルユーザ名をローカルサイトのユーザ名にマップするためには，grid-mapfile と呼ばれるファイルを用いる．これらはジョブ起動のための GRAM インターフェイスでも用いられている方法である．

4.3　GRAM との連携

Globus のジョブ起動インターフェイスである GRAM は，Job Manager と呼ばれる Perl で記述されたモジュールを経由して，TORQUE などのバッチキューイングシステムとインターフェイスする．

予約機構を利用するには，TORQUE に対して予約 ID を提示しなければならない．すなわち GRAM のインターフェイスを介して予約 ID を TORQUE に受け渡す必要がある．このために，GRAM のジョブ記述言語である RSL の拡張機構を用いた．この機能は RSL に `extensions` タグで記述された内容を Job Manager に受け渡す機能である☆．

われわれはこの拡張機構を用い，TORQUE 向けの Job Manager を改変することによって，図 9 に示す書式で記述された予約 ID を，qsub コマンドのオプションとして引き渡すことを可能にした．

4.4　GT4 を用いた事前予約とジョブ実行

WSRF による外部インターフェイスを用いた予約と実行の流れを下に示す．

☆ 原稿執筆時点での最新版である Globus Toolkit ver. 4.0.1 ではこの機能は実装されておらず，Update package を別途導入する必要がある．今後のリリースではデフォルトで導入されると思われる．

表 2　予約関連サービスのオペレーション

オペレーション名	機能	入力	出力
PBSReservationFactoryService			
createPBSReservation	PBSReservation サービスを生成する	開始時間，終了時間，ノード数	生成したサービスの EPR
PBSReservation			
reserve	実際に予約を行う	なし	なし
getStatus	予約情報を取得しリソースプロパティに格納	なし	なし
cancel	予約をキャンセルする	なし	なし
modify	予約を変更する	開始時間，終了時間，ノード数	なし

```
<extensions>
   <schedulerAttrs name="reservationID">
         xxxxxxxxx
   </schedulerAttrs>
</extensions>
```

図 9　RSL 拡張による予約 ID の指定

(1)　クライアントは PBSReservationFactoryService に予約時間帯，ノード数を与えて PBSReservation サービスを生成し，その EPR を受領.

(2)　この EPR に対して reserve を発行.

(3)　EPR に対して ResultStatus リソースプロパティの取得を行い，予約が成功したことを確認.

(4)　EPR に対して ReservationID リソースプロパティの取得を行い，予約に対応する予約 ID を得る．この予約 ID を，前項で述べたように RSL に埋め込み，GRAM クライアントプログラムを用いてジョブを投入.

5. おわりに

グリッド上での資源の同時確保に必要不可欠な事前予約と，キューイングシステムに基づくスケジューリングの関係を調査するためのテストベッドとして，TORQUE のスケジューラモジュールを開発するための API を整備し，これを用いて事前予約機能を持つスケジューラモジュールを実装した．さらに WSRF を用いた外部インターフェイスを実装し，GT4 の GRAM と連動してのグリッド環境での安全な資源予約と予約された資源でのジョブ実行を実現した.

今後の課題としては以下が挙げられる。

- Grid Engine への対応
  今回作成したスケジューラモジュールは TORQUE のみに対応している．今後同一のスケジューラモジュールで TORQUE と Grid Engine の双方をサポートする予定である.
- FCFS 型キューイングシステムと整合性を持つ事前予約機構の検討
  現在の実装では，事前予約はジョブキューとは独立して管理されており，事前予約が常に優先される．よりキューイングシステムとの整合性を持つ事前予約機構のあり方について，考察，検討を進める.
- 実環境へのデプロイと運用
  われわれは，計算資源とネットワーク資源の同時確保を行うスーパスケジューラを実装している[1)]．今後，実験環境への本スケジューラのデプロイを進め，実環境での運用を行い，問題点の抽出を行っていく予定である.

謝　辞

本研究の一部は，文部科学省科学技術振興調整費「グリッド技術による光パス網提供方式の開発」による.

参考文献

1) 竹房あつ子, 林通秋, 長津尚英, 中田秀基, 工藤知宏, 宮本崇弘, 大谷朋広, 田中英明, 鮫島康則, 今宿亙, 神野正彦, 滝川好比郎, 岡本修一, 田中良夫, 関口智嗣: G-lambda: グリッドにおける計算資源と光パスネットワーク資源のコアロケーション, 情報処理学会 HPC 研究会 (to appear) (2006).
2) : LSF. http://www.platform.com/Products/-Platform.LSF.Family/.
3) : PBS Professional. http://www.altair.com/-software/pbspro.htm.
4) : Maui Cluster Scheduler. http://www.-clusterresources.com/pages/products/maui-cluster-scheduler.php.
5) : OpenPBS. http://www.openpbs.org/.
6) : TORQUE Resource Manager. http://www.-clusterresources.com/pages/products/torque-resource-manager.php.
7) : Grid Engine. http://gridengine.sunsource.net.
8) : WSRF. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf.
9) : Globus. http://www.globus.org.
10) 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).
11) : db4objects. http://www.db4o.com/.