Rocksを用いた仮想クラスタ構築システム

中田秀基[†] 横井威[†] 関口智嗣[†]

アプリケーションの動的配備を実現するための一手法として，実行環境であるOSも含めて配備を行う方法がある．この機能を実現する方法としては専用のハードウェアと管理ソフトウェアを使用する方法があるが，高価であり一般的ではない．われわれは，仮想計算機と，OSインストールシステムNPACI Rocksを用いて，安価な通常のPC上でOSとアプリケーションを一体として動的に配備するシステムを構築している．ユーザは，Rocksの Rollパッケージの形で使用するアプリケーションを指定する．システムは計算機クラスタを一体として管理し，ユーザからのリクエストに応じて実計算機をユーザに割り当て，実計算機上の仮想計算機にOSとアプリケーションを配備してユーザに提供する．本稿では本システムの構想と設計，現在までに実装されたプロトタイプに関して述べる．

A virtual cluster installation system with Rocks

Hidemoto Nakada,[†] Takeshi Yokoi[†] and Satoshi Sekiguchi [†]

Dynamic deployment of application can reduce the system management cost drastically. One way to enable this is to deploy application execution envionment, i.e. OS(operating system) along with the application. Although proprietary hardware and software that enables this are already provided by several venders, but it to expensive for usual users. We are constructing a system that enables dynamic deployment of OS and application with virtual machine mechanism and OS install system; NPACI Rocks. On this system users provides their application as a Rocks' Roll. The system manages computer cluster as a whole, allocate computers on requests from users, deploys OS and specified applications and provides them to the users. This paper describes the design concepts of the system and its prototype implementation.

1. はじめに

昨今急速に聞く機会の増えたSOA(Service Oriented Architecture)という言葉に代表されるように，サービスコンポーネントやアプリケーションを動的に配備する要請が高まっている．アプリケーションの動的配備を確実に実現するためには，アプリケーションが実行される環境そのものも動的に構成する必要がある．このための方法の一つとして，OSを含んだ計算機システム全体を動的管理の対象とし，動的に配備するという手法がある．

このような機能を実現する機構として，特殊なハードウェアと管理用のソフトウェアを搭載した計算機が，各ベンダから出荷されている．これらの計算機では，専用ハードウェアを利用することで，外部から電源を制御し，OSの再インストールなどの作業を行うことができる．しかし，これらの計算機は一般に通常の計算機と比較して非常に高価である．また，各ベンダの提供するソフトウェアには互換性がなく，一つのシステムを使用すると，そのベンダにロックインされてしまう可能性がある．

この問題を解決する一つの手法として，近年技術開発がすすんでいる仮想計算機システムを使用する方法が考えられる．仮想計算機を用いることで，通常の安価なPC上で，OSを含んだ計算機システム全体を管理対象として取り扱うことが可能になる．

われわれは，仮想計算機システムとOSのリモートインストールシステムを利用し，アプリケーションとその実行環境であるOSを一体として配備する機構を開発中である．われわれのシステムは計算機クラスタを管理し，ユーザからのリクエストに応じて実計算機をユーザに割り当て，実計算機上の仮想計算機にOSとアプリケーションを配備してユーザに提供する．

仮想計算機システムとしては，Xen[1]およびVMWare[2]を用い，リモートインストールシステムとしてはNPACI Rocks[3),4)]を用いる．さらに，本システムそのものもRocksによって簡便に配備することができる．

本稿の以下の構成を以下に示す．2で本システムの想定する使用法について述べる．3では本稿で利用するリモートインストールシステムRocksと，仮想計算

† 産業技術総合研究所 National Institute of Advanced Industrial Science and Technology (AIST)

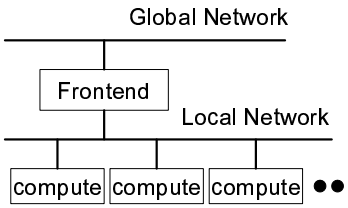

図 1 サービスプロバイダに提供する仮想クラスタの概念図

機システム Xen および VMWare について概説する．4 で本システムの設計を示す．5 で現在までに実装されたプロトタイプに関して述べる．6 で関連研究に関して述べる．7 で今後の課題について述べる．

2. システムの想定

本システムには，クラスタプロバイダ，サービスプロバイダ，ユーザの三者が関与する．クラスタプロバイダは本システムを使用して所有するクラスタを管理し，サービスプロバイダに提供する主体である．サービスプロバイダはクラスタを利用して，ユーザにサービスを提供する．

サービスプロバイダに提供される仮想クラスタの一般形を図 1 に示す．仮想クラスタは 1 つ以上のゲイトウェイとなるフロントエンドノードと 1 つ以上の計算ノードから構成される．ゲイトウェイと計算ノードはプライベートアドレスのローカルネットワークで接続されている．クラスタプロバイダは，ゲイトウェイのグローバルネットワークへのインターフェイスの IP アドレスをサービスプロバイダに提供する．

2.1 クラスタ提供シナリオ

下に，本システムを利用したクラスタ提供のシナリオを示す．

- まず，クラスタプロバイダは本システムを利用し，クラスタをインストールする．
- 次にサービスプロバイダがクラスタプロバイダに対して仮想クラスタ構築を依頼する．その際にサービスプロバイダは，使用開始/終了時刻，使用計算機台数，デプロイされるべきアプリケーション，必要メモリなどの情報を提供する．アプリケーションは，なんらかの形でサービスプロバイダが用意する．
- クラスタプロバイダは本システムを用いて仮想クラスタを実クラスタ上に構築し，サービスプロバイダに提供する．
- サービスプロバイダはアプリケーションを利用したサービスをユーザに対して提供する．

2.2 想定使用法

想定される使用法の一つとして，計算ファームが挙げられる短期的に膨大な量の計算をしなければならない会社があると仮定する．この会社はサービスプロバイダとして，1 月の間 100 台の計算機の使用する契約をクラスタプロバイダとかわす．予約した時刻には，仮想的なクラスタで Condor[5] などのジョブスケジューリングシステムが準備され，ゲイトウェイが引き渡される．これで，会社内の研究者がユーザとなり，計算をサブミットすることができるようになる．Condor のフロッキング機構を利用すれば，個々の研究者はなんら設定を変更することなく増大した計算パワーを享受することができる．

他の想定使用法としては，アプリケーションプログラムのテスト環境としての使用や，Web アプリケーション提供環境としての使用が考えられる．

3. Rocks と仮想計算機システム

本節では，本システムで使用するクラスタインストールシステム Rocks および Xen，VMWare について述べる．

3.1 Rocks によるクラスタのインストール

Rocks は，NPACI(National Partnership for Advanced Computational Infrastructure) の一環として SDSC(San Diego Supercomputer Center) を中心に開発されたクラスタ管理ツールである．クラスタのノード群に対して一括で同じソフトウェアパッケージをインストールすることができる．OS としては，RedHat Enterprise Linux をベースとした CentOS を使用している．

Rocks の対象とするクラスタの構造は図 1 に示したものと同じである．クラスタは frontend と compute node 群で構成される．frontend と compute node 群はプライベートなローカルネットワークを共有する．frontend は このローカルネットワーク向けと，グローバルネットワーク向けの 2 つのネットワークインターフェイスを持ち，compute node 群のルータとしても機能する．

Rocks による compute node のインストールは，Redhat 系 OS のインストール機構である anaconda を用いて，BIOS からの PXE(Preboot eXecution Environment) ブートで行われる．compute node は起動すると DHCP でアドレス取得の要求を出す．frontend はこの要求に対してアドレスとホスト名をアサインする．compute node は，TFTP を使用して初期ブートイメージを取得して起動し，さらに HTTP プロトコルで実際に使用するカーネルやパッケージを取得し，インストールをおこなう．

Rocks では，MAC アドレスに対する IP アドレス (とホスト名) の対応付けを DHCP を用いて行うため，対応表をユーザが手作業で用意する必要がない☆．この点は，Lucie[6] など他のインストールツールとの大

☆ ただし，最初にノード群をインストールする際には，順番に電源を投入することで，対応を教えてやる必要がある．

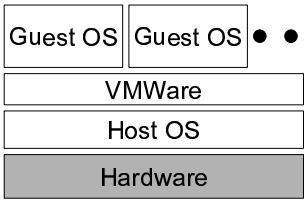

図 2 VMWare による仮想計算機の構成

きい相違の一つである．

3.1.1 Rocks の Roll

Rocks では Roll と呼ばれるメタパッケージによってアプリケーションを管理する．Roll には，RPM 形式のパッケージと，パッケージ間の依存関係，パッケージインストール後の後処理を記述することができる．依存関係と後処理は，XML で表現されるグラフ構造で指定する．クラスタ管理者は Roll を新たに追加することで，クラスタに新たな機能を追加することができる．

3.1.2 アプライアンスタイプ

Rocks では，クラスタ内の個々のノードに対して，異なる構成でのインストールを行うことができる．個々のノード構成をアプライアンスタイプと呼ぶ．アプライアンスタイプは通常 Roll によって規定される．

例えば，クラスタ内に Web サーバ群とデータベースノード群をインストールする場合には，Web サーバアプライアンスとデータベースノードアプライアンスの二つを定義し，個々のノードをインストールする．

3.1.3 Ganglia

Rocks は，ノードの監視に Ganglia[7),8)] を用いる．Ganglia はカリフォルニア大学バークレー校で開発された，拡張性に優れたクラスタモニタツールである．ユーザは，任意の情報をモニタシステムに登録することができる．

Rocks では，Ganglia のセンサー側，モニタ側の双方に python のインターフェイスを追加し，運用している．

3.2 VMWare

VMWare は，もっとも広く用いられている仮想計算機システムである．ブート時の BIOS エミュレーションも含め，非常に高度な仮想化を提供する．図 2 に VMWare による仮想計算機の概念図を示す．ホスト OS，ゲスト OS ともに改変の必要はなく，ホスト，ゲストともに，Windows，Linux に対応している．

3.3 Xen

Xen[1)] は，仮想計算機システムの一つで，Xen の特徴は仮想マシンモニタを用いる点に特徴がある．仮想マシンモニタがハードウェアの直上に位置し，いわゆる「ホスト OS」はドメイン 0 として，「ゲスト OS」はドメイン 1 以降 (ドメイン U と呼ばれる) として，仮想マシンモニタ上で動作する図 3．ドメイン 0，ドメイン U ともに，特殊なカーネルを用意しなければ

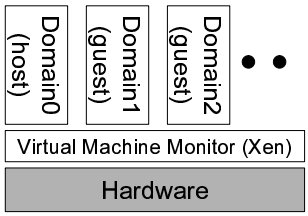

図 3 Xen による仮想計算機の構成

ならないため，対応 OS が事実上 Linux に限られてしまう問題点があるが☆, 実行が高速であることや，各ドメインに対する CPU 割り当てを固定することができるといった特徴がある．

Xen の仮想化は VMWare のそれとは異なり，完全な仮想化ではなく，起動時の BIOS エミュレーションは行わない．また，ディスクの仮想化もハードウェアレベルではないため，パーティション分割やフォーマットなどの低レベル操作を行うことはできない．

4. システムの設計

4.1 システムへの要請

システムへの要請は以下のようにまとめることができる．

- 図 1 に示した構造を持つ
- 任意のアプリケーションがインストール可能であること
- ノード単位でなくクラスタ単位でのさまざまな設定が可能であること
- 同じクラスタプロバイダ上の他のサービスプロバイダが情報を盗み見ることができないこと，また，他の仮想クラスタの情報が見えないこと
- クラスタプロバイダ自身の運用が容易であること

4.2 実クラスタの構成

上記の要請を実現するために想定したクラスタの構成を図 4A に示す．クラスタは，クラスタマネージャ，計算ノード (p-node)，ゲイトウェイノード (GW-node) から構成される．クラスタマネージャはクラスタ全体を管理するサーバで，Rocks の Frontend ノードでもある．p-node は，実際にジョブを実行する仮想計算機を動かすノード，GW-node は，仮想クラスタの外部インターフェイスとなる仮想計算機を動かすノードである．p-node は内部のローカルネットワークにのみ接続されているが，GW-node は外部ネットワークへも接続されている．

4.3 システムの構成

クラスタマネージャの構成を図 5 に示す．クラスタマネージャノードは，2 つのネットワークインターフェイスを持つ．一つは管理ネットワークで，仮想クラス

☆ Version 3.0 以降を VT 対応の CPU で試用した場合にはドメイン U には任意の OS を用いることができる．

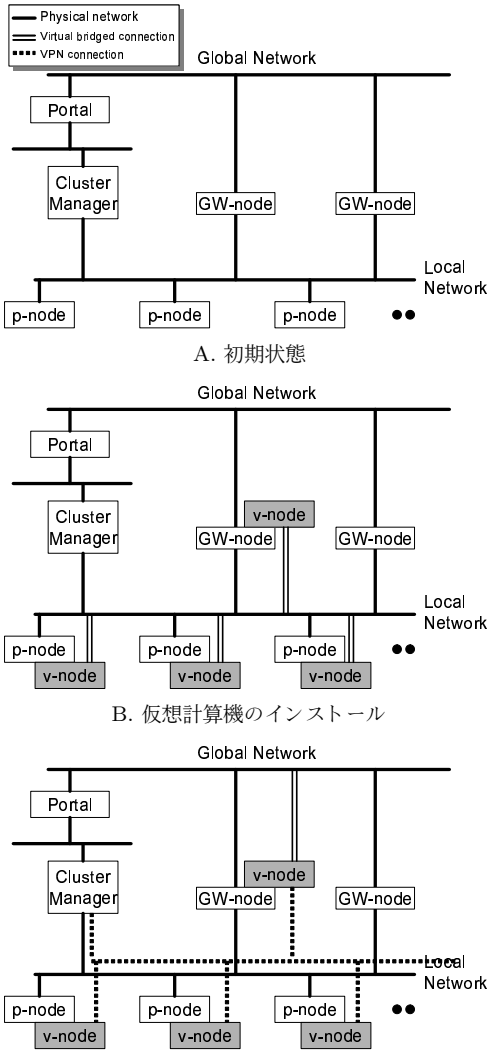

図 4 仮想クラスタのインストール

タ構成のリクエストはこのネットワーク経由で受け付ける．もう一つのネットワークは，クラスタ内ローカルネットワークで，他の計算ノードへのネットワークである．このネットワークを通じてインストールや制御を行う．

中心となるコンポーネントは，クラスタマネージャデーモンである．このデーモンが，Rocks の提供するコンポーネントや，VPN コンポーネント，クラスタモニタリングコンポーネントを使用して，仮想クラスタのコンフィギュレーションを行う．これらのコンポーネントはバックエンドのデータベースを共有する．また，クラスタマネージャサーバは，クラスタに対する予約動作も司る．

クラスタマネージャサーバは，Web インターフェイスと Web サービスインターフェイスを管理ネットワーク

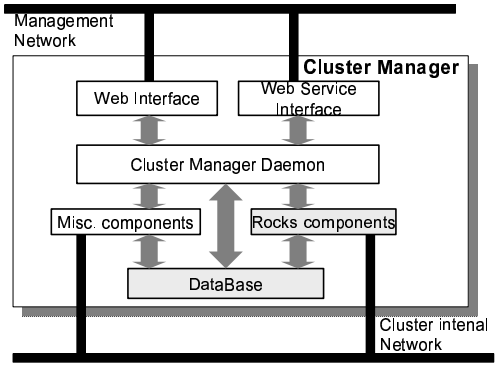

図 5 クラスタマネージャの構成

クに対して提供する．Web インターフェイスは，クラスタプロバイダによるクラスタの管理に用いる．仮想クラスタの予約要求は Web サービスインターフェイスで行う．このさらに上流にはポータルノードが位置し，ユーザに対して GUI インターフェイスを提供する．ポータルノードはクラスタマネージャに対して Web サービスで通信して実際の予約操作を行う．

4.4 実クラスタのインストール

仮想クラスタを提供するために，クラスタプロバイダは実クラスタをインストールしなければならない．この過程は Rocks で行う．まず，クラスタマネージャを Rocks の Frontend としてインストールする．この際の Roll としてクラスタマネージャの他の機能もインストールする．

次に，その他のノードのインストールを行う．p-node と GW-node は基本的に同じアプライアンスとしてインストールするが，GW-node にはその後，グローバルネットワークへのインターフェイスを設定する．

4.5 仮想クラスタのインストール

図 4 に，仮想クラスタのインストールの様子を示す．図 4A は，初期状態である．

クラスタインストール時には，まず p-node および，GW-node 上で仮想ノード (v-node) を起動し，OS およびユーザの指定した Roll を配備する (図 4B)．起動した v-node はそれぞれブリッジネットワーク接続をローカルネットワークに対して持ち，このネットワーク経由でインストールが行われる．

個々のノードのインストールが終了したら，v-node 間の通信路を VLAN に移行する (図 4C)．個々の仮想クラスタごとに異なる VLAN タグを用いることで仮想クラスタ間でのデータ通信を切り分ける．また，クラスタマネージャは，すべての仮想クラスタの VLAN に参加する．これは，仮想ノード上の情報をクラスタマネージャで収集するためである．また，この段階で，GW-node からグローバルネットワーク空間へのブリッジ接続を行う．サービスプロバイダはこの接続を用いてクラスタにアクセスする．

これで仮想ノードと仮想ネットワークの構成が完了するので，この時点でサービスプロバイダに対して，GW-node のグローバルネットワークへのブリッジ接続の IP アドレスを提供する．

5. 実装の現状

5.1 Rocks による仮想計算機のサポート

本システムの構築には Rocks インストーラで，仮想計算機に対応する必要がある．この場合の対応とは，1) Rocks を用いて各計算ノードに仮想計算機システムをインストールできること 2) 各計算ノード上の仮想計算機を Rocks によってインストールできること，である．前者は比較的容易であるが，後者は必ずしもそうではない．

5.1.1 Xen Roll

Xen Roll は 2 つの役割を持つ．一つは各クラスタノードに Xen をインストールしデプロイすること，もう一つは，クラスタノード上に起動した仮想計算機をインストールするスクリプトを提供することである☆．

Rocks の Xen 対応は VMWare 対応と比較してはるかに難しい．Rocks は前述のように，PXE ブートと anaconda によるインストールに依存している．しかし Xen 上の仮想マシンは PXE ブートに対応しない．また anaconda は独自のデバイス検出を行うが，これが Xen の提供する仮想デバイスに対応していないため，検出に失敗してしまう．Fedora Core 5 の anaconda は Xen に対応しており，CentOS も次のメジャーリリースには対応してくると思われるが，現在のところ CentOS を anaconda を利用してインストールすることは難しい．

このため，現在は下記の手法で Xen 上の仮想計算機のインストールを実現している．

- Domain 0 上のスクリプトで仮想ディスクを作成．
- frontend ノードから kickstart ファイルを取得し，それに含まれているパッケージを yum を用いて仮想ディスク上に展開
- Domain U を起動し，初回起動時のスクリプトで，kickstart ファイルに指定された後処理を実行

この手法は CentOS の anaconda が Xen に対応するまでの暫定的な手法である．

5.1.2 VMWare Roll

VMWare Roll は VMWare Player と起動するスクリプトを計算ノードにインストールする．

前述したとおり VMWare では通常の PXE ブートを利用した起動が可能なので，通常の Rocks の起動シーケンスでインストールすることができる．

また，起動の際に仮想計算機が使用する MAC Address を外部から指定することができる．これを用いて，Rocks でインストールされるアプライアンスタイプを制御することができる．

5.2 クラスタマネージャ・プロトタイプ

現在実装されているクラスタマネージャは非常にプリミティブな段階であり，実装されているのは以下の機能のみである．

- 各 p-node 上の xen 環境の状態監視
- 各 v-node の制御
- 簡単な Web インターフェイスの提供

5.2.1 Xen 環境の状態監視

状態監視は，3.1.3 で述べた Ganglia を利用して行った．具体的には，Rocks の提供する Ganglia センサ統合の枠組みである greceptor を利用し，xen monitor から得られる S 式で表現される情報を Ganglia の情報として配布するための，プラグインを作成した．

5.2.2 v-node の制御

v-node の制御として，すでに起動している仮想マシンのサスペンドを実装した．これは p-node に対して ssh で接続し，xen の提供する制御コマンドを使用することで実現されている．

5.2.3 Web インターフェイスの提供

Python による CGI によって Xen 環境の監視と制御を実装した．前述の機構によって Ganglia 上に提供された情報を，Python の Ganglia クライアント API を用いて取り出し，HTML 化して表示する．同様に Python CGI を経由して簡単な v-node の制御が可能になっている．

6. 関 連 研 究

6.1 ORE Grid

ORE Grid[9] は，Globus Toolkit のジョブ起動機構である GRAM と連動したシステムで，ユーザに指定された環境を仮想計算機上に構築し，ユーザの指定したジョブを実行するシステムである．仮想計算機の動的構成には，インストールツール Lucie[6] を用いている．

本システムがクラスタを数日-数週間の単位でリースすることを前提としているのに対し，ORE Grid では個々のジョブを対象としている点が大きく異なる．

6.2 Virtual Workspace

Virtual Workspace[10),11)] は，Globus Project[12] の一環として行われているプロジェクトで，ジョブの実行環境を仮想計算機上に構築することを目的としている．WSRF[13] を用いたインターフェイスで仮想計算機が構成する仮想的なワークスペースを作成し，そこでユーザのジョブを実行する．

Virtual Workspace も，基本的に個々のジョブの実行を指向している点がわれわれのシステムと異なる．

☆ Xen Roll は Rocks チームによって開発されたもので，本研究の貢献ではない．

7. おわりに

仮想計算機と OS インストールシステムを用いて，OS とアプリケーションを一体として動的に配備するシステムに関して，設計とプロトタイプ実装に関して述べた．プロトタイプ実装では，Rocks を利用した仮想計算機クラスタの構築と Web インターフェイスからの簡単な管理を実現できた．

本システムの設計と実装はまだ端緒についたばかりであり，多くの課題が存在する．

- ストレージの提供
  現在の実装では，クラスタに提供されるストレージは，各仮想計算機に与えられた仮想ディスクのみである．しかしこれは，低速で低容量であるため，データインテンシブなアプリケーションに対しては不足である．
  これに対する対策としては，実資源としてのストレージを確保しネットワーク経由で，仮想計算機に提供する方法が考えられる．この場合には，ストレージの動的確保の実現，複数ユーザのストレージの相互分離，などの問題を解決しなければならない．
- 他の仮想クラスタ間ネットワーク機能
  現在の設計では，仮想クラスタ間のネットワーク分離を VLAN で実現しているが，仮想クラスタのセキュリティ要請によっては，VLAN では十分でない場合も想定される．より高度な安全性を提供する手段としてソフトウェア VPN の使用に関して，検討を進める．
- 複数クラスタの統合的運用
  単一のクラスタでは，提供できる資源には限界がある．複数のクラスタ上に，仮想的な単一クラスタを形成し，運用することも検討する必要がある．
- 他のノードインストール機構への対応
  現在の設計は，Rocks を実クラスタと仮想クラスタ双方のインストールに使用する．このため，仮想クラスタで使用できる OS が CentOS に限定されてしまう．これは，アプリケーションによっては問題になることが予想される．他のディストリビューションもサポートできるよう，実クラスタのインストール環境と仮想クラスタのインストール環境を分離し，仮想クラスタ構築を他のインストール機構で行うことを検討する必要がある．
  特に，今後の Windows CCS の普及や，アプリケーションテストセンターとしての運用を考慮すると Windows への対応は必須である．これを実現するためには，Windows をインストール可能な機構の統合が必要となる．
- Web サービスインターフェイスの検討
  クラスタマネージャに対するクラスタ予約のインターフェイスとして Web サービスインターフェイスを検討している．このプロトコルとして，Global Grid Forum の CDDLM-WG[14] の成果を採用することを検討する．

謝　辞

Rocks に関してご教示いただき，Xen Roll を提供していただいた SDSC Rocks チームの Mason Katz, Greg Bruno, Anoop Rajendra に感謝する．

参考文献

1) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauery, R., Pratt, I. and Warfield, A.: Xen and the Art of Virtualization, *SOSP 2003* (2003).
2) VMWare. http://www.vmware.com.
3) Papadopoulos, P. M., Katz, M. J. and Bruno, G.: NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters, *Cluster 2001: IEEE International Conference on Cluster Computing* (2001).
4) Rocks. http://rocks.npaci.edu/.
5) Condor. http://www.cs.wisc.edu/condor/.
6) 高宮安仁, 真鍋篤, 松岡聡: Lucie: 大規模クラスタに適した高速セットアップ・管理ツール, 先進的計算基盤システムシンポジウム SACSIS2003 論文集, pp. 365–372 (2003).
7) Ganglia. http://ganglia.sourceforge.net/.
8) Massie, M. L., Chun, B. N. and Culler, D. E.: The Ganglia Distributed Monitoring System: Design, Implementation, and Experience, *Parallel Computing*, Vol. 30, No. 7 (2004).
9) 高宮安仁, 山形育平, 青木孝文, 中田秀基, 松岡聡: ORE Grid: 仮想計算機を用いたグリッド実行環境の高速な配置ツール, *SACSIS2006 (toappear)* (2006).
10) Virtual WorkSpace. http://workspace.globus.org/.
11) Keahey, K., Foster, I., Freeman, T. and Zhang, X.: Virtual Workspaces: Achieving Quality of Service and Quality of Life in the Grid, *Scientific Programming Journal* (2006).
12) Globus Project. http://www.globus.org.
13) WSRF. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsrf.
14) CDDLM-WG. https://forge.gridforum.org/projects/cddlm-wg/.