グリッド上における仮想計算機を用いたジョブ実行環境構築システムの高速化

山形育平[†] 高宮安仁[†]
中田秀基[††,†] 松岡聡[†,†††]

グリッド上で実行されるジョブの多様化に伴い、ジョブが要求する実行環境も多様化しつつある．そこで我々は仮想計算機と自動設定・インストールツールを用いて投入されるジョブごとに専用の仮想実行環境を動的に提供するシステム ORE Grid を開発している．このシステムを使用することにより約 173 秒で 16 台の BLAST 実行環境を動的に構築することが可能であるが，今後のグリッドの応用領域の広がりを考慮すると構築時間を短縮することも必要である．この構築時間短縮のため一度構築した仮想計算機環境を保存，再利用するキャッシュサーバーを作成し，ORE Grid に組み込むことを提案する．このキャッシュを使用することによりパッケージインストール時に起こるパッケージサーバーへの負荷集中を回避できる．このシステムを実装し，従来のシステムとの比較を行ったところ約 12%の構築時間削減効果が見られた．

A virtual-machine based fast deployment tool for Grid execution environment

Ikuhei Yamagata,[†] Yasuhito Takamiya,[†] Hidemoto Nakada[††,†] and Satoshi Matsuoka[†,†††]

With the increased variety of jobs executed in the Grid, the execution environments requested by such jobs have becoming increasingly diversified. So, we implemented system that supply exclusive virtual execution environment every submitted job by virtual machine and installer. So, we implemented system that supply exclusive virtual execution environment every submitted job by virtual machine and installer. This system enable to setup 16 machines that can execute BLAST job dynamically at about 173 seconds. This research suggest system that save and recycle virtual execution environment for shorten time to build job execution environment. We implemented this system and evaluate against old system. So our experiences have shown that the build time has been reduced by 12% than older one.

1. はじめに

グリッドコンピューティング[1] による広域分散計算が実用化されつつあり，科学技術計算などの専門分野だけでなく，応用分野でも広く認知されてきている．グリッドとは，複数の管理ドメイン上に存在する広域的に分散したリソース（計算機，ストレージ，実験装置など）を，ユーザの需要とリソース提供者のポリシーを元に，動的に構成される仮想組織で安全に共有するための技術である[2]．グリッドの浸透に伴いグリッド上で実行されるジョブは急激に多様化し，その結果ジョブが実行環境として要求する OS やアプリケーション，ソフトウェアライブラリなども多様化している．

そこで我々はグリッド上に動的にジョブ実行環境を構築するシステム[3] を開発している．これは実計算機上に仮想的な実行環境を構築できる仮想計算機[4] と，動的な実行環境の構築が可能な自動設定・インストールツールを用いることにより実現する．これによりユーザーはジョブ実行に必要な実行環境記述を提示することで，動的に専用の仮想環境が構築されユーザーに提供することができる．

このシステムを用いて相同性検索ソフト BLAST[5] ジョブ実行環境を 16 台構築するのに約 173 秒を要する．通常 BLAST のジョブ実行時間全体では数時間以上に及ぶこと，グリッド上でのジョブの大部分も実行時間が数時間から数日に及ぶことを考えるとこの時間は十分許容範囲であると考える．一方で今後のグリッ

† 東京工業大学
Tokyo Institute of Technology
†† 産業技術総合研究所
AIST(National Institute of Advanced Industrial Science and Technology)
††† 国立情報学研究所
NII(National Institute of Informatics)

ドの応用領域の広がりを考慮すると構築時間を短縮することも必要である．また急な負荷集中による緊急のサーバー増設など非常に短時間での計算機の提供を求められた場合，この構築時間では不十分である．

そこで我々は構築した仮想計算機のイメージ（キャッシュ）を保存し，このキャッシュを用いて計算機環境を構築する方法を提案する．これによりキャッシュ内に入っているアプリケーションやライブラリのインストールを行う必要が無いため，環境構築時間の短縮が期待できる．

我々はこのキャッシュを用いた環境構築システムを設計，実装し，従来のキャッシュを使用しないシステムとの比較を行った．その結果従来の手法と比較してインストール時間が 35.9%，構築時間全体でも 12.4%の削減効果が見られることを確認した．

2. ORE Grid

本章では本研究で使用した環境構築システム ORE Grid の目的と実装を説明する．

2.1 目　　的

グリッド上で実行されるジョブの多様化に伴い，OS やソフトウェア，ライブラリなどジョブが要求する実行環境も多様化しつつある．

しかし，グリッド上のジョブ実行環境が提供するソフトウェアリソースは管理ドメインごとの管理ポリシーに強く依存するため，ユーザーがジョブ実行に必要な環境が常に提供されているとは限らない．これを解決する手法として投入されるジョブごとに専用の計算リソースを確保し，そこへ OS やライブラリなど専用の実行環境を動的に構築することが考えられる．

しかしこの手法には (1) ジョブの実行ごとに OS やソフトウェアを再インストールすることはリソースを共有している他のユーザーに対して影響が大きい，(2) ジョブの実行要求ごとに OS やソフトウェアを再インストールすることは管理コストが大きい，という問題が挙げられる．このため，ジョブ実行に必要な実行環境を管理ポリシーに依存せず，既存の環境を破壊することなく自動的に構築する技術が求められている．

この問題を解決するために我々はグリッド上でユーザーが動的にジョブ実行環境を構築できるシステム，ORE Grid を開発している．ORE Grid の詳細については[6] を参照する．

2.2 実　　装

ORE Grid は仮想計算機，自動設定・インストールツール，ジョブ実行サービスを組み合わせて実装されている．各コンポーネントについて説明を行う．

仮想計算機　仮想計算機とは一台の物理マシンをソフトウェアで多重化し，仮想的で独立した複数台のマシンとして使用できるようにする技術である．ハードウェアや OS，ソフトウェアが仮想化されるため，物理マシン上の環境とは独立した計算機環境を実現できる．

PC 向けの仮想計算機技術として VMware[7),8)]，Xen[9)]，coLinux[10)]，UML (User Mode Linux)[11)] などがあり，仮想化の度合いや仮想化手法の違いにより，I/O 速度や CPU 速度の違いなど性能面でのトレードオフが存在する．

ORE Grid では使用できる仮想計算機として VMware と Xen に対応しており，環境構築の依頼が来た場合，この仮想計算機上に環境が構築されることになる．どの仮想計算機を使用するかはユーザーが決定するため実行するジョブに合った仮想計算機を選択することが出来る．

自動設定・インストールツール　自動設定・インストール技術とは，インストールする OS やカーネル，ソフトウェア，ライブラリおよび各種ソフトウェアをあらかじめ環境構成として入力しておくことによって，指定された構成を複数の計算機へ自動かつ高速に設定するシステムである．

代表的な自動設定・インストールツールとして RedHat Kickstart[12)]，NPACI Rocks[13)]，Lucie[14)] などがあり，それぞれについてセットアップできる OS や設定方法が異なる．

ORE Grid では自動設定・インストールツールとして Lucie を使用しており，これを仮想計算機の構築機構として用いることにより，ジョブ実行に必要な環境を仮想計算機上に自動的に構築することができる．

Lucie ではインストールするアプリケーションやバージョン，OS を柔軟に選択することが可能であり，HDD のパーティション情報からインストールするカーネルやアプリケーションなどハードウェアからソフトウェアレベルまで様々な設定を行うことが出来る．インストールはネットワーク経由により全ノードで自動的に開始されインストール時のキーボード入力などの対話的操作はすべて排除されている．

実際に Lucie を用いてインストールを行う場合まず Lucie の設定ファイルを記述する．設定ファイルにはインストールする対象マシンのハードウェア情報やインストールするソフトウェアパッケージやカーネル，パッケージサーバーのアドレスなどを記述する．次に Lucie に付属する管理コマンドを用いて Lucie サーバー上に Lucie インストーラーの構築を行う．ここではインストールに必要なインストーラー用 Linux イメージや各種インストールサービスの設定が Lucie サーバー上に自動的に設定される．インストール対象マシンを起動，再起動するとネットワークブートによりインストーラーイメージが自動的にインストール対象マシン上にロードされ，インストールが開始さ

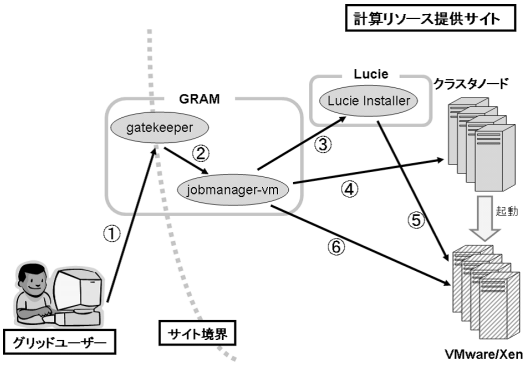

図 1 ORE Grid

れる．

インストールが開始されるとまず各計算機上のハードウェアの検索を行い，必要なモジュールのロードを行う．次に設定ファイルで指定された形式でハードディスクのパーティション設定，フォーマットを行う．次にまったくアプリケーションがインストールされていない最小のブートイメージを Lucie サーバーから NFS でコピーし，それをインストールするノード上に展開する．その後アプリケーションやカーネルのインストールを行う．次に設定ファイルの作成やスクリプトの実行を行い，それが終了するとインストールが完了する．

ジョブ実行サービス ジョブ実行サービスではグリッドユーザーからのジョブ実行環境構築要求を受け取り，ローカルリソース（クラスタ）上への仮想計算機の起動や自動設定・インストールツールの実行の指示，および実際のジョブ実行を行う．

ORE Grid ではグリッド上で標準的に用いられている Globus Toolkit[15] のジョブ実行システム GRAM を使用している．GRAM ではユーザーからジョブ実行依頼を受け取ると，バックエンドとして JobManager を起動し，実際のジョブ実行を行う．ORE Grid ではこの JobManager として jobmanager-vm を実装した．jobmanager-vm ではグリッドユーザーからのジョブ実行環境構築要求を受け取ると，指定された台数分の仮想計算機を立ち上げ，自動設定・インストールツールの実行を指示し，仮想計算機環境の構築を行う．

環境構築を行う場合の手順は以下の通りである（図 1）．

（1） ユーザーは構築する仮想計算機のスペック（台数，メモリ容量，HDD のディスク容量，インストールするパッケージなど）と実行するジョブを GUI を用いて指定する．それで作られた設定ファイルと合わせて gatekeeper に対して環境構築の依頼を行う．

（2） ユーザーから環境構築依頼を受けた gatekeeper は環境構築用 JobManager である jobmanager-vm を起動する．

（3） jobmanager-vm はユーザーから送られてきたジョブ実行環境設定を元に Lucie installer に対して仮想計算機の構築を指示する．

（4） jobmanager-vm はユーザーから要求された台数分の仮想計算機をクラスタノード上に起動する．

（5） Lucie installer は起動した仮想計算機に対してユーザーが指定したアプリケーションをインストールする．

（6） インストール終了後 jobmanager-vm は構築した仮想計算機に対してジョブをサブミットする．

このシステムを用いることによりユーザーは任意のジョブ実行環境を仮想計算機上に構築することが出来る．

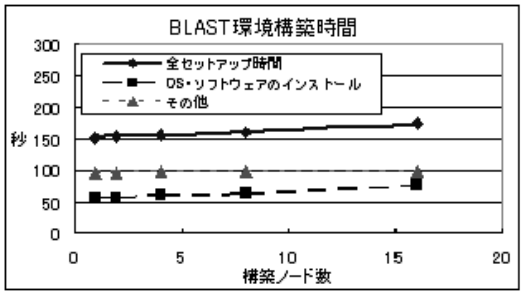

図 2 BLAST 環境構築時間

2.3 環境構築時間と考察

ORE Grid を用いた環境構築時間を示す．計測環境は章 4 と同じ環境である．

構築時間は図 2 のようになった．16 台構築に約 172 秒要している．この時間は現在のグリッドのジョブ実行時間と比較すると十分許容範囲であると言える一方，今後のグリッドの応用領域の広がりを考慮すると構築時間を短縮することも必要であると考える．

仮想計算機構築にかかる時間の主な部分は Lucie のセットアップ，仮想計算機インストール，再起動である．Lucie のセットアップ時間は大部分が Linux ベースシステムの展開に裂かれており，大幅な高速化は難しい．そこで仮想計算機インストール時間と再起動時間について考える．

インストール時間を削減する方法として一度構築した仮想計算機のイメージを保存（キャッシュ）し，再利用する方法がある．また再起動時間を削減する方法としてはブートローダーを介さずに起動中の Linux カーネルとは別のカーネルの起動が可能である Linux の kexec 機構を使用する方法がある．本研究ではキャッシュを使用するシステムに注目し，このシステムの実装を行った．

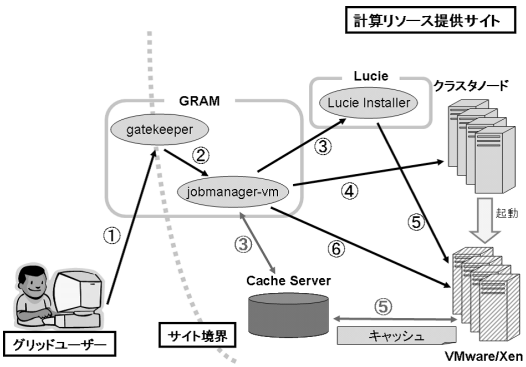

図 3 キャッシュを用いた環境構築手順

3. キャッシュを用いた環境構築システムの設計・実装

我々は構築時間を短縮するために一度構築した仮想計算機のイメージを保存しておき，再利用する方法を提案する．

キャッシュの形式として

(1) 仮想計算機のディスクイメージをそのまま利用
(2) ブートイメージ（/bin/や/usr など）を tar で圧縮したもの

の 2 通りが考えられる．

（1）のメリットとしてディスクのパーティション作成やフォーマットに要する時間も削減することが可能である．しかしながら仮想計算機ごとに仮想ディスクの形式が違うため，仮想計算機ごとにキャッシュを用意する必要がある．またキャッシュのデータが（2）と比較して大きいため，転送に時間がかかるというデメリットも存在する．

（2）のメリットとして仮想計算機ごとにキャッシュを用意する必要がない．また（1）と比較してデータも小さいため転送が高速である．一方でディスクのパーティション作成やフォーマットを行わないでよいという副次的なメリットは存在しない．

本システムでは Xen，VMware どちらにも対応するため（2）のキャッシュ形式を採用している．Lucie ではプレーンなブートイメージをコピー，展開しているため，本システムではこのブートイメージではなく，キャッシュを展開する．これによりキャッシュにインストールされているパッケージはインストールする必要がなくなる．

本システムではキャッシュサーバーを実装し，これを組み込んだ（図 3）．キャッシュを使用する場合の構築手順は以下のように変更になる．

(3)' jobmanager-vm はユーザーから送られてきたジョブ実行環境設定を元にインストールするアプリケーションのリストをキャッシュサーバーに対して転送する．キャッシュサーバーはリストを受け取り，これがインストールされているキャッシュが存在するかを判断し，その結果を jobmanager-vm に返す．また jobmanager-vm は Lucie installer に対して仮想計算機の構築を指示する．

(5)' Lucie installer は起動した仮想計算機に対してユーザーが指定したアプリケーションをインストールする．この際仮想計算機はプレーンなブートイメージを展開する代わりに，キャッシュサーバーにアクセスし適切なキャッシュを取得しそれを展開する．

またキャッシュが存在しない場合インストール時にキャッシュの作成を行うことになる．この場合構築手順は以下のように変更になる．

(5)" Lucie installer は起動した仮想計算機に対してユーザーが指定したアプリケーションをインストールする．アプリケーションインストール後仮想計算機はブートイメージを圧縮しそのデータをキャッシュサーバーへ転送する．キャッシュサーバーはキャッシュを受け取り新しいキャッシュとして登録する．

キャッシュサーバーでは現在所持しているキャッシュとそれにインストールされているアプリケーションのリストを管理している．Lucie サーバーからキャッシュに関する問い合わせがきた場合このリストを参照し返答を行う．

本研究ではキャッシュの転送方法として，各計算機が直接キャッシュサーバーへアクセスしキャッシュを取得する方法と，dolly+[16] を使用する方法の 2 つを実装した．dolly+とはサーバー上にあるファイルを多数のノードにコピーする時に発生する負荷の集中を回避するために，論理的なリング結合関係をノード間に作り，コピーするファイルをノード間で順々に送る方法であり，これにより高速な転送が可能である．

しかしながらキャッシュを使用することにより（1）NFS 経由での最小のブートイメージファイルのコピー，展開，(2) パッケージ取得，インストール時間，が削減できる一方，(1)' キャッシュサーバーからのキャッシュのコピー，(2)' キャッシュの展開、を必要とする．そのため $(1)+(2) > (1)'+(2)'$ である場合，逆に構築時間がかかってしまうことになる．そこで各々にかかる時間をモデル化し，それを元にキャッシュを使用するかどうか判断する必要がある．

4. 評　　価

本システムの評価環境は東京工業大学松岡研究室の PrestoIII クラスタを使用した．

評価環境として東京工業大学松岡研究室の PrestoIII クラスタを使用した．スペックは表 1 の通りである．また構築する仮想計算機のスペックは表 2 のとおりで

表 1　評価環境

CPU	AMD Opteron250 × 2
メモリ	PC2700 2GB
OS	Debian GNU/Linux sarge
カーネル	Linux 2.4.28
ネットワーク	Gigabit Ethernet
VM	VMware 5.0 (linux 版)

表 2　仮想マシンのスペック

HDD	20GB(SCSI buslogic)
メモリ	256MB
OS	Debian GNU/Linux sarge
カーネル	Linux 2.4.28

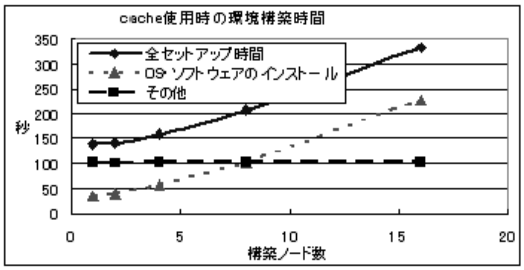

図 4　cache を用いた場合の BLAST 環境構築時間

ある．

構築する環境として BLAST（Basic Local Alignment Search Tool）実行環境を選択した．BLAST は相同性検索ソフトウェアの一種であり，問い合わせ配列を配列データベースと比較し，類似配列の検索を行うソフトウェアである．BLAST を実行するために必要なパッケージとして Debian Linux では blast2 ソフトウェアパッケージが提供されている．そこで構築する実行環境の追加パッケージとして blast2 を選択した．

キャッシュを用いた環境構築時間は図 4 である．1 台構築時にはインストール時間が 35.6%，構築時間全体では 8.2%の削減効果が見られた．一方で 16 台構築時には逆に構築時間が増加してしまっていることがわかる．これはキャッシュサーバーにアクセスが集中してしまい，転送スピードが低下したことが原因である．

次にキャッシュを dolly+を用いて転送した場合の環境構築時間は図 5 である．これでは 16 台構築時でもインストール時間が 35.9%，構築時間全体では 12.4%の削減効果が見られており，本システムの有効性が示された．

この削減効果は構築台数が増えるにつれてより大きくなるものと考えられる．これは Lucie インストーラーではパッケージインストールを行うためパッケージサーバーにアクセスが集中してしまうが，キャッシュサーバーと dolly+を使用することによりアクセス集中を回避することができるからである．

最後に仮想計算機、キャッシュサーバー間のバンド

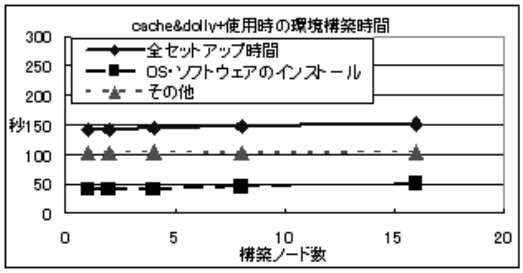

図 5　cache&dolly+を用いた場合の BLAST 環境構築時間

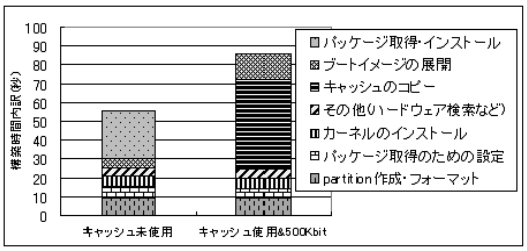

図 6　キャッシュサーバーとのネットワークバンド幅を 500kbits/sec にした環境での環境構築時間の内訳

幅を 500kbits/sec にして構築を行った。環境構築時間の内訳は図 6 のようになり、キャッシュを使用しない場合と比較して構築時間が増加していることがわかる。このようにネットワークバンド幅などの条件によっては逆に構築時間が増加することが示された。

したがってキャッシュを使用することによる時間の変化をモデル化し、これを元にキャッシュを使用するかどうかを判断する必要がある。

5. 関 連 研 究

仮想計算機を用いてグリッド上にジョブ実行環境を動的に構築する技術はいくつか提案されている．

VMPlants[17] は仮想計算機上にジョブ実行環境を動的に構築し，提供するシステムである．仮想計算機の構築に用いる VM の使用として，ユーザーは DAG（有向非巡回グラフ）を用いて構築手順を手続き的に記述する．また良く使われる仮想計算機のディスクイメージをゴールデン・イメージとして保存し，これを DAG による構築手順中に再利用することでゴールデン・イメージと類似した構成の仮想計算機の起動や構築の高速化を図っている．

6. まとめと今後の課題

本研究ではグリッド上に動的にジョブ実行環境を構築するシステムである ORE Grid での環境構築時間を短縮するため，一度構築した仮想計算機のディスクイメージ（キャッシュ）を保存，再利用するシステムを提案した．またこのシステムの設計，実装を行い，実装し

たシステムを用いて実際に相同性検索ソフト BLAST の実行環境の構築を行った．その結果キャッシュを使用しない場合と比較してインストール時間が約 36%，構築時間全体で見ると約 12%短縮され，キャッシュを用いたシステムの有効性を確認した．

今後の課題としては以下の点が挙げられる．

- 実装したシステムの拡張
  今回実装したシステムのよりいっそうの洗練が必要である．今回の実装では Lucie サーバーからインストールするパッケージのリストが送られてきた場合，すべてにマッチするキャッシュが存在したときだけキャッシュの使用が可能である．しかしながら一部マッチした場合でもキャッシュを使用し，キャッシュ内にインストールされていないパッケージは個別にパッケージサーバーからインストールするということも可能である．このように柔軟なキャッシュの使用を行っていくべきである．
- キャッシュ使用を判断する指針の作成
  3 章でも述べたようにキャッシュを使用するかどうかを判断するより精密な指針の作成が必要である。
- 別の手法での仮想計算機構築時間の短縮
  キャッシュを用いたシステムだけでなく，例えば 2.3 章で述べた kexec を用いた再起動時間短縮手法など，様々な方法を模索，実装していく必要がある．

謝辞 本研究の一部は，独立行政法人新エネルギー・産業技術開発機構 基盤技術研究促進事業（民間基盤技術研究支援制度）の一環として委託を受け実施している「大規模・高信頼サーバの研究」の成果である．

参 考 文 献

1) Foster, I. and Kesselman, C.(eds.): *The Grid: Blueprint for a New Computing Infrastructure*, Morgan-Kaufmann (1998).
2) Foster, I., Kesselman, C. and Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations, *International Journal of Supercomputing Applications*, Vol. 15, No. 3 (2002).
3) 山形育平, 青木孝文, 高宮安仁, 中田秀基, 松岡聡: カスタマイズ可能な仮想計算機上におけるグリッドでのジョブ実行, 電子情報処理通信学会研究報告 2005-CPSY(SWOPP 2005) (2005).
4) Meyer, R. A. and Seawright, L. H.: A Virtual Machine Time-Sharing System., *IBM Systems Journal*, Vol. 9, No. 3, pp. 199–218 (1970).
5) Altschul, S. F., Gish, W., Miller, W., Myers, E. W. and Lipman, D. J.: Basic local alignment search tool, *Journal of Molecular Biology*, Vol. 215, pp. 403–410 (1990).
6) 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. submitted.
7) Sugerman, J., Venkitachalam, G. and Lim, B. H.: Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor, *Proceedings of the General Track: 2002 USENIX Annual Technical Conference*, Berkeley, CA, USA, USENIX Association, pp. 1–14 (2002).
8) Waldspurger, C.A.: Memory resource management in VMware ESX server, *SIGOPS Oper. Syst. Rev.*, Vol. 36, No. SI, pp. 181–194 (2002).
9) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and the art of virtualization, *SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles*, New York, NY, USA, ACM Press, pp. 164–177 (2003).
10) Aloni, D.: Cooperative linux, *Proceedings of the Linux Symposium*, Vol. 1, pp. 23–32 (2004).
11) Dike, J.: A user-mode port of the linux kernel, *Proceedings of the USENIX Annual Linux Showcase and Conference* (2000).
12) Hamilton, M.: KickStart Document. `http://wwwcache.ja.net/dev/kickstart`.
13) Papadopoulos, P. M., Katz, M. J. and Bruno, G.: NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters, *Proceedings of 2001 IEEE International Conference on Cluster Computing*, Newport, CA (2001).
14) Takamiya, Y., Manabe, A. and Matsuoka, S.: "Lucie: A Fast Installation and Administration Tool for Large-scaled Clusters", *Proceedings of Symposium on Advanced Computing Systems and Infrastructures (SACSIS2003)*, pp.365–372 (2003).
15) Foster, I. and Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit, *The International Journal of Supercomputer Applications and High Performance Computing*, Vol. 11, No. 2, pp. 115–128 (1997).
16) Manabe, A.: Disk cloning program 'dolly+' for system management of pc linux cluster, *Computing in High Energy Pysics and Nuclear Physics* (2001).
17) Krsul, I., Ganguly, A., Zhang, J., Fortes, J. A. B. and Figueiredo, R. J.: VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing, *SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing* (2004).