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

2008－HPC－116 (6)
2008／8／5

仮想クラスタのステートレス化のためのRocks 5ディスクレス化機構

小川宏高[†] 中田秀基[†] 広渕崇宏[†]
伊藤智[†] 関口智嗣[†]

計算機資源の効率的な運用の方法として仮想化が注目されており，仮想的なクラスタを管理するさまざまなシステムが提案されている．我々も2005年度よりユーザからの依頼に応じて予約ベースで仮想クラスタを構築・提供するサービスを実現する仮想クラスタ管理システムGriVonを研究・開発している．このような仮想クラスタ構築技術でとりわけ重要なのは，仮想クラスタで利用するストレージに求められるさまざまな性質（性能、スケールアウトの容易さ、管理の容易さ、計算機資源の集約を動的に行うためのライブマイグレーション、耐故障性など）をいかに実現するか、である。我々はこうした課題を解決することを目的として、本年5月にリリースされたNPACI Rocks[1] version 5にディスクレスブート化機構を実現した。本稿ではその詳細を述べる。GriVonの次期バージョンでは、この機構を用いてより柔軟な仮想ストレージ管理を実現する予定である。

Rocks 5 Diskless Boot Mechanism for making Virtual Clusters stateless

Hirotaka Ogawa,[†] Hidemoto Nakada,[†] Takahiro Hirofuchi,[†]
Satoshi Itoh[†] and Satoshi Sekiguchi[†]

Virtualization techniques are gaining acceptance as a core technology for utilizing computing resources in computer centers, and several systems are proposed for contructing ang managing "virtual clusters". Since 2005, we also have been developping "GriVon: Virtual Cluster Management System" which allows users to reserve a virtual cluster through an easy-to-use Web interface. For each reservation, the system automatically allocates and configures a bunch of virtualized computing resources including VMware-based virtual machines, iSCSI storages, and networks for a virtual cluster. However, most systems have serious problems in managability and performance in *virtual storage*. For example, storages for virtual machines strictly binds to a hard disk owned by their host machine, therefore, it could be difficult to realize fault-tolerance and/or load-balancing and to accomplish enough parallel read-write performance. To resolve these problems, in this paper, we propose *stateless virtual cluster*, which concentrates all of storages for virtual and physical machines to virtual storage nodes, and enables diskless boot from them. And we also describe the design and implementation of core component of stateless virtual cluster.

1. はじめに

計算機資源を集約的に保有するデータセンタでは，資源の稼働率の向上や設備コスト・運用コストの削減が強く求められており，近年こうした要請を満たす方法として仮想化技術が注目されている．仮想化技術とは，計算機システムの構成要素であるCPU，ストレージ，ネットワークなどを論理的に複数に分割し，複数のCPU，複数のストレージ，複数のネットワークとして利用可能にする技術である．この技術を用いることで，物理的に保有している計算機資源をより多くの仮想的な計算機資源として利用することができ，運用の自由度や資源の稼働率の向上，設備コストの圧縮に大きな効果がある．

こうした仮想化技術自体はすでに成熟の域にあるが、その恩恵を広くまた簡便に享受するためには，運用に伴う人的コストの削減や多拠点からなるデータセンタの資源の活用が不可欠である。こうした課題をクリアするために、多拠点からなるデータセンタの計算機資源の中から、仮想的な計算機資源の集合（仮想クラスタ）を「ユーティリティ的」に，つまりユーザの要求に応じてオンデマンドかつフレキシブルに，アロケートして利用できるようにする技術が強く求められている．

こうした要求を実現するために我々は2005年度よりユーザからの依頼に応じて予約ベースで仮想クラスタを構築・提供するサービスを実現する仮想クラスタ管理システムGriVon[2]を研究開発している。

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

このような仮想クラスタ構築技術でとりわけ重要なのは，仮想クラスタで利用するストレージに求められるさまざまな性質（性能、スケールアウトの容易さ、管理の容易さ、計算機資源の集約を動的に行うためのライブマイグレーション、耐故障性など）をいかに実現するか、である。

現状の仮想計算機のファイルシステムは，通常ホスト計算機のファイルシステム上のイメージファイル，もしくはパーティションを用いて実現されるが，この方法ではファイルシステムがホスト計算機に保有される物理ストレージ装置に強く束縛されてしまう．このため，ファイルシステムの容量設定の自由度が低くなり，動的な負荷分散や障害対策を目的としてファイルシステムを含んだ仮想計算機イメージをホスト計算機間でマイグレーションすることも難しくなる．また，管理対象となる物理ストレージ装置がホスト計算機群に分散して配置されるということは，管理自体を困難にする上，耐故障性を実現したければホスト計算機の責任で実現しなくてはならない．

こうした問題を解決することを目的として、我々は本年5月にリリースされたばかりの NPACI Rocks[1] version 5 にディスクレスブート化機構を実現することで、仮想クラスタのステートレス化を行った。本稿では，その詳細を紹介する。

2. 仮想クラスタ向けストレージ

今日 CPU やネットワークの仮想化技術自体はほぼ成熟したと言ってよいのに対して，ストレージに関してはそうではない．仮想計算機のファイルシステムは，通常ホスト計算機のファイルシステム上のイメージファイル，もしくはパーティションを用いて実現されるが，この方法ではファイルシステムがホスト計算機に保有される物理ディスク装置に強く束縛される．

このため、ファイルシステムの総量がホスト計算機の物理ディスク装置に制約されるため，容量設定の自由度が低い。また、動的な負荷分散や障害対策を目的としてファイルシステムを含んだ仮想計算機イメージをホスト計算機間・サイト間で（ライブ）マイグレーションすることが困難になる．

さらに、物理ディスク装置がホスト計算機群に分散して配置されるため，管理対象が分散してしまう．特に，耐故障性を実現したければホスト計算機の責任で実現しなくてはならない．Amazon EC2 において、ファイルシステムイメージと仮想計算機の生存期間が一致しているのは、分散したホスト計算機の管理を大幅に簡略化するための設計上の重要な選択の一つだと言える。

これらの問題を解決する方法の一つは，仮想計算機のファイルシステムを専用のストレージノードに集約することである。が、別の問題もある。

一つは、ストレージノードに低価格で汎用に利用可能な NFS や iSCSI[3] を仮定すると，ストレージノード上のファイルシステムイメージを仮想計算機からいかにトランスペアレントに利用可能にするか、という問題である。言い換えると，ファイルシステムイメージを自動的に仮想計算機にルートファイルシステムとして認識させ，利用可能にする機構の実現が必要不可欠である。

もう一つは、コモディティとして入手可能な CPU のコア数の増加などによって，単体のホスト計算機でサービス可能な仮想計算機の個数は増加しており，ストレージノードに要求されるストレージ容量やスループットをいかに達成するかという問題である。

3. 仮想クラスタ管理システム GriVon

3.1 GriVon の概要

仮想クラスタ管理システム GriVon[2] は、ユーザからの依頼に応じて予約ベースで仮想クラスタを構築・提供するサービスを実現するシステムで、我々が 2005 年度から研究・開発している（**図 1**）。

具体的には，このシステムを用いると，

(1) ユーザはまず Web インタフェースを用いて利用する計算機資源の量（CPU の個数，メモリー容量，ディスク容量など）と構成情報，その占有時間を指定して予約し，

(2) 予約開始時間が近づくと，システムは予約内容に基づいて，VMware Server 1.0 による仮想計算機，VLAN，VPN，iSCSI ストレージなど必要な計算機資源を生成・確保し，

(3) その仮想クラスタに対して San Diego Supercomputing Center を中心に開発されているクラスタ自動構築ツールである NPACI Rocks version 4[1] を用いて OS・アプリケーションなどのインストールと設定を行ってユーザに提供する

ことができる．

また、GriVon システム自体も Rocks を用いてインストールするように設計されている。したがって、物理計算機資源に余裕が生じた場合などにオンデマンドで物理クラスタを追加しておけば、それに応じてより多くの仮想クラスタノードをシステムで提供できるようになる。

その他、実装の詳細については，2) を参照のこと．

3.2 GriVon での仮想ストレージの実現

GriVon では，仮想計算機の root ファイルシステムを iSCSI ターゲット（サーバ）上の仮想化されたディスクボリュームを利用する。具体的には，まずホスト計算機が iSCSI イニシエータ（クライアント）となって，iSCSI ターゲットに接続する．すると，そのストレージはホスト計算機上ではブロックデバイスとして

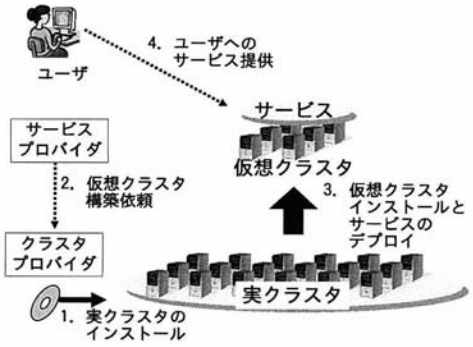

図 1 クラスタプロバイダ, サービスプロバイダとユーザ

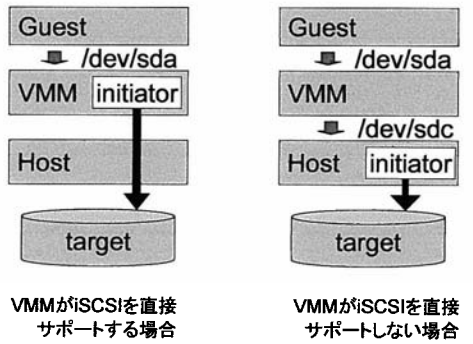

図 2 iSCSI の構成

認識される。このブロックデバイスを物理ディスクとして指定し，仮想計算機を起動する．つまり，仮想計算機としては通常の物理的なデバイスをマウントしている場合と同じ動作をしているにもかかわらず，ホスト計算機が仲介することで，実際には iSCSI 経由でターゲット上のストレージを使用できる．

図 2 にこの様子を示す．左図が，仮想計算機システムが iSCSI を直接サポートしている場合を示している．ゲスト OS のディスクアクセスは仮想計算機システムによって iSCSI プロトコルに変換される．これに対し，右図が今回とった構造である．ゲスト OS のディスクアクセスは，仮想計算機によってホスト計算機にアタッチされたディスクへのアクセスに変換される．これが，ホスト計算機にインストールされた iSCSI イニシエータによって，さらに iSCSI プロトコルに変換される．

この方法は 2 で述べた問題の多くを解決する．仮想計算機のファイルシステムがホスト計算機と分離され，iSCSI ストレージ上に格納されるため，容量設定の自由度が確保でき，負荷分散や障害対策を目的としてファイルシステムを含んだ仮想計算機イメージをホスト計算機間でマイグレーションすることも容易になる．また，管理対象がストレージノードに集中するため，ホスト計算機の責任で仮想計算機のディスクイメージの耐故障性を保証する必要がなくなる．

iSCSI は仮想計算機から見ると，ストレージの IP レベルインタフェースを規定しているだけなので，並列読み書き性能や耐故障性の実現を下位レイヤーに分離・隠蔽できる。例えば，複数の RAID コントローラを装着したストレージノードを複数用意して，冗長化と分散化を行ったストレージクラスタとして利用することで耐故障性と並列読み書き性能を両立することも理論的には可能である．

また、遠隔サイト間でのライブマイグレーションを実現するために、遠隔ホスト計算機間でのストレージの動的再配置[4] を容易に実現する場合にも、iSCSI や NBD[5] のような IP レベルインタフェースを利用するのは合理的な選択だと言える☆。

しかし一方で，ホスト計算機の物理ディスクの使用を前提としているため，ホスト計算機の設定内容がそのホスト計算機の物理ディスク装置に束縛されるという問題が残る．したがって，そのディスク装置に障害が起きた場合，仮想計算機イメージは失われないものの、ホスト計算機の設定内容は失われる可能性がある．また，ホスト計算機を他の業務のために明け渡す必要に迫られた場合にもマイグレーションすることは困難である．

また，ホスト計算機が仮想クラスタのために専用に利用可能でない場合（例えばホスト計算機が通常時は業務に使用されていて仮想クラスタにはストレージを資源として貸し与えることができない場合）や，省電力のためディスクレス構成のホスト計算機を利用したい場合，そもそも「ホスト計算機の物理ディスク」の存在が仮定できないため，GriVon のアプローチは適用できない．

4. 仮想クラスタのステートレス化の実現

4.1 概　　要

3.2 で述べた問題を解決するには、ホスト計算機と仮想計算機の双方をディスクレスブートできるようにすることで，仮想クラスタシステム全体の，物理ディスク装置への依存を一掃する、言い換えると物理クラスタへの永続的なステートの保持を排除する、ことが必要である。

具体的には，**図 3** に示すように，まずホスト計算機群をディスクレスブートするようにセットアップし，次にそのホスト計算機上で動作する仮想化ソフトウェアを使って仮想計算機群を生成する．さらにその仮想計算機群もディスクレスブートするようにセットアップする．

☆ 実際、我々は 4) の実装を GriVon にマージしていくことを予定している。

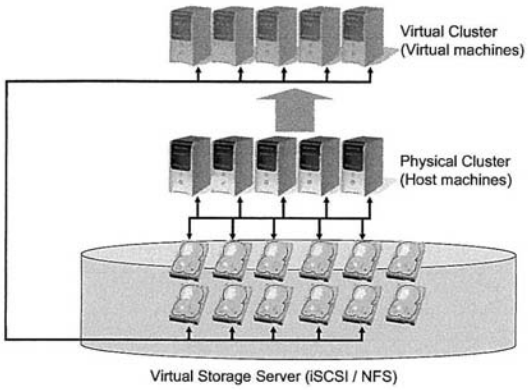

図 3 仮想クラスタのステートレス化

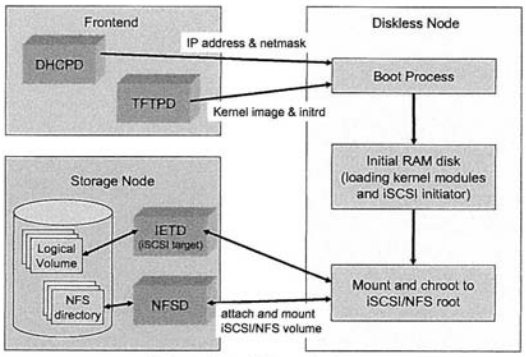

図 4 ディスクレスブート

このような構成を採ることで，ホスト計算機と仮想計算機のファイルシステムがホスト計算機の物理ディスクから分離され，iSCSI ストレージ上に格納されるため，容量設定の自由度が確保できる．また，負荷分散のためにファイルシステムを含んだ仮想計算機イメージをホスト計算機間でマイグレートしたり，ホスト計算機の障害時には代替のノードにまるごとマイグレートしたりすることができる．

3.2 の前半で触れたように，GriVon の実装では，仮想計算機が利用する仮想ディスクをアタッチする目的でホスト計算機が iSCSI イニシエータとなっていた．これに対して，ステートレス仮想クラスタではホスト計算機・仮想計算機がそれぞれ自分の利用するストレージをアタッチする目的で iSCSI イニシエータとなる．したがって，両者の独立性が高まり，管理の容易性も増す．

システムへの要件は，ホスト計算機と仮想計算機の双方が PXE compliant であるということだけである．現代のほとんどの PC・サーバ製品に付属している NIC も PXE compliant であり，また，GriVon が仮想計算機実装として利用している VMware Server の仮想 NIC もそうである．GriVon が利用している Rocks もまた PXE compliant な NIC を要求することからこの要件はそれほど強い制約ではない。ただし，Xen の準仮想化環境では DomainU を PXE ブートできない．この場合には，3.2 で述べたのと同様に Domain0 に一旦リモートストレージをアタッチする必要がある．

また，ホスト計算機の物理ディスクは，原理的にはホスト計算機および仮想計算機のスワップファイル，ディスクキャッシュ，あるいは一時的なワークスペースとして利用することも可能なため，必ずしも無駄とはならない．

以下では，ディスクレスブートの実現方法について説明した後，仮想クラスタのステートレス化を実現するコアコンポーネントとなる，iSCSI/NFS Diskless Roll の実装について述べる．

4.2 ディスクレスブート

ディスクレスブートは，PXE ブートしたカーネルから NFS root や iSCSI デバイスを root mount することで，ディスクレスの状態でマシンを動作させるものであり，特に NFS root を用いる方法は比較的枯れた技術として知られている．

図 4 に示すように，ディスクレスブートは，Linux がサポートしている 2 段階のブートプロセスを利用して実現される．

まず，フロントエンドノードで動作する DHCP サーバと TFTP サーバから，それぞれ IP アドレスとネットマスク，kernel イメージと RAM ディスクイメージ（initrd）を取得する．次に kernel イメージをメモリ上に読み込み，initrd を初期 root ファイルシステムとしてマウントし，初期ブートプロセスを開始する．

初期ブートプロセスは、（RedHat 系のディストリビューションでは）initrd に含まれる/init スクリプトに書かれている通りに実行される．具体的には，必要なデバイスドライバなどを実現する kernel モジュールを読み込み，IP アドレスなどを設定し，NFS root を利用する場合には NFS のクライアントモジュール，iSCSI root を利用する場合には iSCSI イニシエータをそれぞれ起動する．続いて，ストレージノードが提供する NFS ディレクトリや，iSCSI ディスクを/tmproot にマウントし，そのディレクトリに chroot する．最後に chroot した root ファイルシステムから 2 段階目のブートプロセスを実行する．

4.3 実　　装

実装の基本的な方針は、NFS と iSCSI によるディスクレス化機構を Rocks version 5 の Roll として実装することである。なぜなら、GriVon の実装では、ホスト計算機・仮想計算機の双方とも Rocks を用いて管理・インストールされるので、共通のディスクレス化 Roll を用いてホスト計算機でも仮想計算機でもディスクレスインストール・ディスクレスブートが可能になるからである。また、既存の実装では iSCSI イニシ

エータのセットアップはGriVonの管理下で行っていたが、独立したRollとして実現することで、GriVonを使わない環境でも利用できるようになるからである。

実装において重要なのは，Rocksではインストール時にPXEブートを利用するのに加え，ディスクレスブート時もPXEブートを行うということである．それゆえ，フロントエンドで動作するDHCPサーバとTFTPサーバは，インストール時とディスクレスブート時にそれぞれ適切なkernelイメージやinitrdなどを選択してPXEクライアントに渡すように設定されなければならない．

また、Rocks 5では、Rocks 4と異なり、ベースになるLinuxディストリビューションがCentOS 4からCentOS 5に変更されている。CentOS 5ではkernelが大幅に更新されており、Open-iSCSI[6]イニシエータが利用するカーネルモジュールがあらかじめ組み込まれるなど、Rocks 4での実装は再利用できないが、より容易になっている。

これに留意して行った実装の概要を**図5**に示す☆．

まず，インストール時は以下のように実行される．

(1) フロントエンドノードでinsert-ethersを実行する．このとき，insert-ethersのプラグインモジュールがストレージを確保する．iSCSIの場合にはiSCSIターゲット上でディスク領域を確保し，ディスクレスノード用に公開し，NFSの場合にはNFSサーバ上でディスクレスノード用のディレクトリを作成・公開する．

(2) 続いてディスクレスノードの電源を入れると，フロントエンドのDHCPサーバとTFTPサーバから，それぞれIPアドレスとネットマスク，kernelイメージとinitrdを取得し，初期ブートプロセスを開始する．

(3) ディスクレスノードはinitrdに書かれたスクリプトに従い，フロントエンド上のkickstart.cgiが生成するディスクレスノード用のカスタマイズされたkickstartファイル（ks.cfg）をダウンロードし，anacondaインストーラでインストールを開始する．

(4) anacondaでは，iSCSIターゲットをアタッチ，またはNFSマウントしてインストールプロセスを実行し，実行が完了したら，フロントエンドのXMLRPCサーバを呼び出す．

(5) XMLRPCサーバは続くディスクレスブートに必要なkernelオプションをDHCPサーバに設定する．

(6) ディスクレスノードをシャットダウンする。

(7) フロントエンドノードでiSCSIターゲットをアタッチし、kernelイメージとinitrdを取り出し、デタッチする。initrdにディスクレスブートに必要なパッチを適用する。具体的にはスタティックリンクされたOpen-iSCSI[6]イニシエータ（/sbin/iscsistart）の追加とinitスクリプトの書き換えを行う。

(8) kernelイメージとinitrdをTFTPサーバにそれぞれ設定・格納する．

ディスクレスブート時は，インストール時に設定されたkernelオプションなどにしたがって，図4と同様の手続きにしたがってブートする．

4.4 Rocksによる実装の拡張

Rocksでのインストールは，インストール対象のホスト計算機・仮想計算機にリモートストレージをアタッチした状態で行なわざるを得ない．しかし，実質的にはディスクレスブートのインストールはどのマシンにアタッチして実施しても構わない．インストール専用マシンにリモートストレージをアタッチし，適当なディレクトリにマウントした状態でコマンドラインでanacondaを実行してインストールを実施した後，そのリモートストレージをディスクレスノードに割り当てることもできる．

この方法を採った場合，インストール専用ノード，ディスクレスノードとも再起動する必要はないのでインストール時間の大幅な短縮が見込める．反面，インストール時にデバイスのprobe，デバイスドライバの組み込みなどを正常に行なえない危険性はある．

仮想計算機のように均質であると考えてよいターゲットマシンに対しては，上記のようなインストール方法が極めて有効だと考えられる．

4.5 基本性能の評価

スペースの都合で詳細は割愛するが，基本性能の評価を行った。

1000Base-Tのスイッチに直結されたCore 2 Duo 1.86Ghz，2GBメモリのマシンを2台用意し，それぞれRocksのフロントエンドノード、コンピュートノードとしてセットアップし、フロントエンドノードにiSCSIターゲット、コンピュートノード上にVMware Server 1.06をそれぞれインストールしておく。この状態で、VMware Serverの仮想計算機に対してフロントエンドノードから約500パッケージ(約1GB)を含むCentOS 5のインストールを行う。

GriVon実装（コンピュートノード上のiSCSIイニシエータにアタッチする）、本稿のディスクレス実装（ディスクレスブートした仮想計算機がiSCSIイニシエータになり、直接アタッチする）でもインストールには約2000秒を要した。後者は初期セットアップ時にDHCPサーバやTFTPサーバとの通信が必要になるが、インストール処理本体に要する時間に比べてほとんど無視できた（むしろインストール時間の分散の方が大きい）。

☆ 図はiSCSIの場合．NFSを用いる場合にもほぼ構造は同じであるため，割愛した．

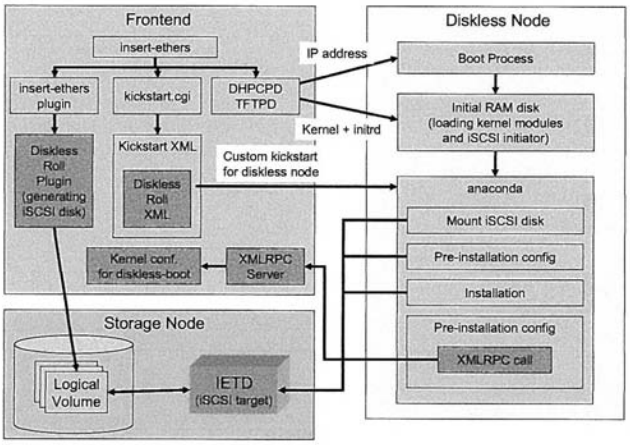

図 5 iSCSI Diskless Roll

5. 関連研究

Nopparat Nopkuat らの Diskless Roll[7] は NPACI Rocks で，Thin-OSCAR[8] は OSCAR クラスタインストーラで，それぞれディスクレスブートを実現するものである．両者の実装は，NFS root での利用に限られており，仮想クラスタへの利用は考えられていない．また，後者はディスクレスブート用の initrd を手動で生成する必要があるなど，実装が不十分な点もある．

6. まとめ

計算機資源の効率的な運用の方法として仮想化が注目されており，仮想的なクラスタを管理するさまざまなシステムが提案されている．しかし，多くのシステムは，仮想ストレージの管理性や性能などに問題を抱えている．例えば，仮想計算機のディスクイメージがホスト計算機に束縛されているために，耐故障性の実現、負荷分散、資源集約が困難になったり，不十分な並列読み書き性能を改善する手立てがなかったりといった問題がある。

こうした問題を解決するため，ホスト計算機・仮想計算機の全てのストレージを仮想ストレージサーバに集約し，ディスクレスシステムとして運用するために必要なディスクレスブート化機構を NPACI Rocks[1] version 5 上に実現した。

今後の課題は、Rocks 5 に組み込んだディスクレスブート化機構を利用した、GriVon の次期バージョンの開発、サイト間マイグレーション[4] との組み合わせなどである。もちろん、実装の完成度を上げて実用的な規模での性能評価を行うことやスケールアウトに必要なストレージサーバの構成方法なども必要である。

謝辞

本研究の一部は科研費（20700038）の助成を受けたものである。

参考文献

1) 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).
2) 中田秀基，　横井威，江原忠士，谷村勇輔，小川宏高，関口智嗣：仮想クラスタ管理システムの設計と実装，情報処理学会論文誌 ACS，Vol.48, No.SIG13, pp.13–24 (2007).
3) : iSCSI Specification. `http://www.ietf.org/rfc/rfc3720.txt`.
4) 広渕崇宏，小川宏高，中田秀基，　伊藤智，関口智嗣：仮想クラスタ遠隔ライブマイグレーションにおけるストレージアクセス最適化機構，情報処理学会研究報告 HPC (to appear) (2008).
5) Breuer, P.T., Lopez, A.M. and Ares, A.G.: The Enhanced Network Block Device, *Linux Journal* (2000).
6) : Open-iSCSI: RFC3720 architecture and implementation, `http://www.open-iscsi.org/`.
7) Nopkuat, N. et al.: Diskless Roll, `http://research.thaigrid.or.th/en/Diskless_Roll`.
8) Ligneris, B. and Giraldeau, F.: Thin-OSCAR : Design and future implementation, *Proc. of 17th Intl.Symp. on High Performance Computing Systems and Applications*, pp.261–265 (2003).