複数サイトにまたがる仮想クラスタの構築手法

広渕崇宏[†] 横井威[†] 江原忠志[†,††]
谷村勇輔[†] 小川宏高[†] 中田秀基[†]
田中良夫[†] 関口智嗣[†]

大規模計算機資源の効率的かつ柔軟な運用を可能にする手法として, 仮想クラスタ技術が注目されている. しかし単一拠点の資源のみを対象とする既存システムでは, サイト内に存在する資源規模に制限されて, 仮想クラスタのスケーラビリティや柔軟性が最大限享受できるとはいいがたい. そこで我々は複数サイトにわたって横断的に仮想クラスタを構築可能なシステムの実現に取り組んでいる. 単一実行環境としての透過性を実現し, WAN のネットワーク遅延や帯域の制限のもとで, 拠点ごとのハードウェア差異に対応する必要がある. そこで本稿で提案する仮想クラスタの構築手法においては, WAN を介して拠点横断的に仮想クラスタの内部ネットワークを構築することで単一実行環境を実現し, パッケージベースのクラスタ構築機構によってハードウェア差異を吸収し決め細やかなカスタマイズによる実行環境の最適化を可能にする. さらに拠点ごとの透過的なパッケージキャッシュが迅速なクラスタ構築を可能にする. 評価実験を通して, 提案手法によって WAN を介する通信量を低減させながら大規模な仮想クラスタを迅速に構築可能であることを確認した.

A Multi-Site Virtual Cluster System over WAN

Takahiro Hirofuchi,[†] Takeshi Yokoi,[†] Tadashi Ebara,[†,††]
Yusuke Tanimura,[†] Hirotaka Ogawa,[†] Hidetomo Nakada,[†]
Yoshio Tanaka[†] and Satoshi Sekiguchi [†]

A virtual cluster is a promising technology for reducing management cost and improving capacity utilization in datacenters and computer centers. However, recent cluster virtualization systems do not have the maximum scalability and flexibility due to limited hardware resources in one physical location. In this paper, we propose a multi-site virtual cluster system over a WAN, which is based on recent broad backbone networks and end-to-end network resource reservation. To address heterogeneity of physical clusters and interconnect networks, the proposed system exploits a package-based cluster deployment method for full customizability of virtual nodes. It extends private networks of virtual clusters over a WAN for a monolithic system view. In addition, its transparent package caching mechanism allows rapid deployment of large-scale virtual clusters. Our experiments showed that virtual clusters were successfully installed through the proposed mechanism with small network traffic over a WAN.

1. はじめに

データセンタや計算機センタでは絶え間なく変化する計算機需要の下でも計算機資源を最大限効率的かつ柔軟に運用しなければならない. そこで近年では仮想化技術の導入が進みつつある. 物理的な計算機資源を論理的に分割して共有することで資源の利用効率を高め, また計算機資源の柔軟な再配置を可能にする. なかでも大規模な計算機群を取り扱う HPC 分野においては, 仮想クラスタシステムとよばれる計算機クラスタ自体を仮想化できる仕組みに注目が集まっている. 多様なアプリケーションに対する実行環境全体をオペレーティングシステム自体を含めて柔軟に構築できる. またその実行環境を大規模ノードを対象として迅速に配備できる.

しかし, 既存の仮想クラスタシステムは単一のデータセンタや計算機センタなどにおいて閉じて運用されるのみという問題がある. その運用の柔軟性は単一拠点に存在する物理資源規模によって制限されてしまい, 最大限の効率性を追求できるとはいいがたい. どのような計算機需要に対してもスケーラブルに仮想クラスタを構築できることが求められており, また単一拠点の大部分に影響する停電やハードウェアメンテナンス

† 産業技術総合研究所
Advanced Institute of Science and Technology
†† 数理先端技術研究所
Mathematical Science Advanced Technology Laboratory Co., Ltd.

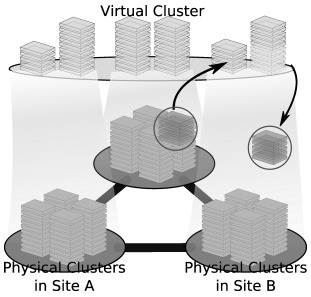

図 1 マルチサイト仮想クラスタ概念図

の際にも仮想クラスタの運用を継続できる必要がある.

そこで, 我々は WAN 環境上複数拠点に存在する計算機資源をもとにした単一システムイメージの仮想クラスタ (マルチサイト仮想クラスタ) 実現に取り組んでいる. 動的かつ迅速に複数サイトにまたがった仮想クラスタを構築し, 一旦構築した仮想クラスタの構成ノードおよび構成サイトを, 動的に増やしたりあるいは減らしたりすることができる. またマルチサイト仮想クラスタは手元の物理クラスタとなんら変わらないユーザビリティを提供する.

しかし, その構築は必ずしも容易ではない. 複数拠点に散らばる計算機資源をもとに, 透過的にひとつのクラスタ仮想化環境を提供し続けなければならない. また, LAN とは遅延や帯域に大きな開きがある WAN 環境をその中に抱えても, 迅速に実行環境をアプリケーションに対して最適な形で構築できる必要がある. しかも, 実際には拠点ごとに差異のあるハードウェア資源から単一の実行環境を提供しなければならないという課題もある.

そこで, 本論文ではマルチサイト仮想クラスタシステムの構築手法を提案する. 提案手法では, 仮想クラスタの内部ネットワークを拠点間で横断的に構築して複数拠点からなる資源をクラスタとして単一の実行環境として取り扱えるようにする. このとき複数サイトにまたがる大規模な仮想クラスタを使いこなすためには, 本質的に隠蔽することができないノード間のネットワーク特性やハードウェア資源の差異や変化に対応して, 常に最適なシステムを構築していく必要がある. そこで, パッケージベースの仮想クラスタ構築機構を採用し, 拠点ごとに動作するパッケージキャッシュ機構を設けることで, ユーザが各ノードの決め細やかなカスタマイズを何度でも迅速に試行することを可能にする.

本稿では我々が目指すマルチサイト仮想クラスタについて述べた後, その構築手法を論じる. そして設計および実装とその評価実験を通して, WAN 環境下における提案手法の妥当性を検証する.

2. マルチサイト仮想クラスタ

我々はマルチサイト仮想クラスタによって, 大規模計算クラスタ運用環境における高度な拡張性と柔軟性を, 高いユーザビリティのもとで実現することを目指している (**図 1**). 仮想化技術によって各拠点に存在するヘテロジーニアスな計算機資源の差異を吸収でき, それらを WAN を介して統合することで大規模な単一システムイメージの仮想クラスタを提供できる. データセンタや計算機センタ間を結ぶバックボーンネットワークの帯域は 1Gbps を超えつつあり, また end-to-end のネットワーク資源予約も利用可能になりつつある. ゆえに WAN を介して大規模仮想クラスタを構築しスケーラブルに運用していくことは十分可能であると考えている.

また物理資源の差異を吸収した仮想化資源からなるクラスタであるがゆえに, その構成を柔軟に変更することが可能になる. 新たな拠点に接続し既存の仮想クラスタを動的に増強したり, あるいは需要の動向に合わせて適切な規模にまで縮小したりできる. ある拠点におけるハードウェアメンテナンス予定や計算資源需要の逼迫をふまえて, 仮想クラスタを構成する一部の仮想ノードを動的に他の拠点のもので置き換えることもできる.

たとえ遠隔拠点に分散する仮想計算機からなるマルチサイト仮想クラスタであっても, ユーザに対しては単一のクラスタとしてのシステムイメージを提供する. ユーザが仮想クラスタの構成要素の物理的な配置を極力意識することなく, 物理クラスタを扱う場合と同様に利用できることを目指す. 従来のクラスタ向けプログラムができるかぎりシームレスに最小限の手間で, 分散計算機資源からなるマルチサイト仮想クラスタ上でも実行できることが望ましい.

3. 仮想クラスタシステムの提案

マルチサイト仮想クラスタの実現手法について議論する.

3.1 要求事項の整理

複数サイトにまたがった仮想クラスタを構築するために, 単一拠点のみに対応した仮想クラスタとは異なった要求事項が構築システムに対して存在する.

クラスタ仮想化機構は, 単一拠点であれ多拠点であれ, その基本的な仕組みを維持したまま両者に対応できなければならず, どのような拠点構成をとっても単一実行環境としての透過性が常に維持される必要がある. 運用拠点の追加や離脱に対しても実行環境を維持したまま柔軟に対応できなければならない. クラスタ実行環境におけるソフトウェア資産の多くがマルチサイト環境においてもシームレスに実行可能でなければならない.

また，ネットワーク遅延や可用帯域に制限のあるWANを介しても，大規模な仮想ノード群からなる単一の実行環境を迅速に構築できなければならない．WAN環境におけるネットワーク帯域や遅延はLAN環境におけるものとは大きな開きがあり，既存の仮想クラスタ管理の仕組みをそのまま複数拠点をまたいで適用できるとはいいがたい．マルチサイト仮想クラスタとなることで，単一拠点を対象とするよりも大規模な仮想クラスタを構築できなければならない．仮想クラスタ内部のOSやアプリケーションのインストールや設定が，広域環境においても依然として容易かつ迅速に完了する必要がある．

さらに，複数拠点の計算機資源を元に構築するゆえに，クラスタを構成するノードやノード間ネットワークの差異に対して柔軟に対応できる必要がある．必ずしもマルチサイト仮想クラスタをホストする物理クラスタすべてが同一性能であるとは限らないし，拠点ごとに異なった計算資源量を割り当てる可能性もある．またCPUアーキテクチャのように仮想計算機で吸収できないハードウェア的差異もある．マルチサイト仮想クラスタを構成する拠点間のネットワーク特性もそれぞれ異なりうる．マルチサイト仮想クラスタにおいて，以上のような差異がある中でも最適な実行環境を構築できる必要がある．そのためには，ユーザが単一実行環境の中に存在する仮想ノードやノード間ネットワークの違いを意識して，仮想ノードごとに異なる計算処理を割り当てられるようにインストールするアプリケーションやその設定を柔軟に何度でも試行できなければならない．仮想ノードごとに異なる計算処理を割り当てられるなど決め細やかにカスタマイズ可能でなければならない．複数サイトに分散する大規模な計算機資源を使いこなすためには，各ノードの役割をユーザ自身の手によって容易に設定できなければならない．

3.2 マルチサイト仮想クラスタの実現手法

そこで提案システムにおいては，仮想クラスタ内部ネットワークをWANを介して遠隔拠点間との間で透過的に拡張することで，単一実行環境の仮想クラスタを作り出す．さらにパッケージベースのクラスタ構築手法を採用した上で拠点ごとに透過的なインストールパッケージのキャッシュ機構を設けて，不可避な不均一性への対応として決め細やかなノードカスタマイズを可能にし，同時にシステムの迅速な配備を実現する．マルチサイト仮想クラスタのシステム内部では単一クラスタとしての実行環境が維持され，そのクラスタ管理フレームワークの中で実行環境を維持した柔軟なノード管理を可能にする．

広域イーサネットサービスやIP網を経由したL2レベルでのVPNをもちいて，仮想クラスタ内部ネットワークを多拠点横断的に構築できるよう拡張する．ひとつの仮想クラスタを構成する仮想計算機やストレージサービスを接続するネットワークは，複数サイトにまたがって仮想クラスタが構築された場合にも，依然としてひとつのネットワークセグメントとして維持される．これにより，マルチキャストやブロードキャストを用いるアプリケーションを変更せずとも複数サイトにまたがった仮想クラスタに用いることができる．既存の成熟しつつあるクラスタ管理ツールを複数サイトにまたがる仮想クラスタ構築に際しても適用できる．複数サイトにまたがって作成された仮想クラスタが引き続きユーザにとってもアプリケーションにとっても透過性を有する．つまり，仮想クラスタを構成する仮想計算機等の物理的な配置にかかわらず，ひとつの仮想クラスタとして単一のシステムイメージとして見え続ける．

パッケージベースのクラスタ構築機構を採用することで柔軟なノードカスタマイズによるクラスタシステム構成の最適化を可能にする．さらに，透過的なインストールパッケージキャッシュ機構の組み合わせにより，WANをまたいで大規模な仮想クラスタを構築する際に発生する非常に大きなデータ転送を劇的に低減し，ネットワーク遅延が存在する環境下においても実用可能なクラスタ構築速度を実現する．仮想クラスタシステムの実現手法においては大きく分けて2種類あり，あらかじめ生成された仮想マシンイメージをホストノードに対して転送する方式と，インストールするソフトウェアパッケージのリストとそれらの個別の設定を仮想ノードごとに生成して仮想計算機に渡し，仮想計算機内部でインストーラを実行する方式とがある．ここで我々は，ユーザによる決め細やかなカスタマイズが仮想ノードごとに可能であるという点とWAN環境における親和性とを考慮し，後者による方式をマルチサイト仮想クラスタにおいて採用する．各拠点に存在する物理クラスタのCPUアーキテクチャなど仮想計算機においても見られるハードウェア的な差異をインストール時に吸収でき，また仮想計算機の性能やノード間ネットワーク性能を意識してカスタマイズが可能になる．さらに仮想クラスタを構築する際に，ソフトウェアパッケージ単位で拠点ごとにキャッシュすることで，仮想マシンイメージそのものをキャッシュするよりも効率的に容易にデータ転送の極小化がはかれる．

またマルチサイト仮想クラスタのシステム内部では単一クラスタとしての実行環境が維持され既存のクラスタ向けプログラムの移行を容易にしている．さらに，そのノード管理機能自体も，マルチサイト化にともなって特別な変更をともなうことなく，単一クラスタとしてのフレームワークの中で取り扱う．仮想ノードの動的な追加や削除あるいは置き換えなどの処理は，透過的に拡張された仮想クラスタ内部ネットワークを介することにより異なる拠点に存在しても区別無く扱われ，どのような拠点構成をとってもクラスタとして

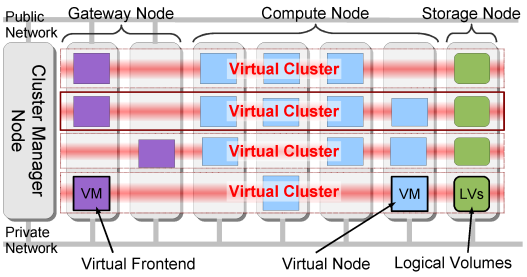

図 2 各拠点のクラスタ管理システム構成

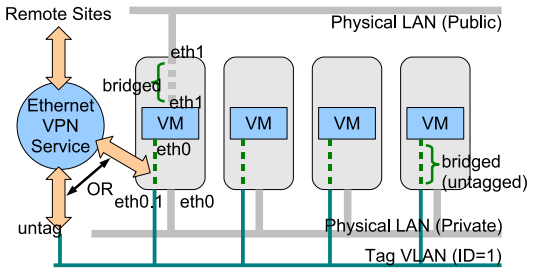

図 3 VPN による仮想クラスタ内部ネットワーク拡張

の実行環境を維持したまま柔軟に対応できる.

4. 設計と実装

本節では, 提案システムにおけるクラスタ仮想化機構について述べた後, その設計と実装について述べる.

4.1 クラスタ仮想化機構

提案システムでは, パッケージベースの仮想クラスタ構築機構にもとづくマルチサイト仮想クラスタを実現している. 一般的なクラスタ管理フレームワーク Rocks[1] のもとに, VMware Server や iSCSI, VLAN 等の技術を我々独自の管理システムの下で組み合わせている. Rocks が提供する豊富なクラスタ向けアプリケーションパッケージがそのままマルチサイト仮想クラスタ上でも導入可能である. マルチサイト機構を伴わない部分については文献[2),3)] に詳細が述べられているが, 説明の都合上ここではその概要を含めて記す.

各拠点におけるシステム構成は**図 2** に示される. 物理クラスタの各ノードは 4 種類存在する. **クラスタマネージャノード**は管理インタフェースを提供し, 要求に応じて仮想クラスタの作成や破棄などを他の物理ノードに対して命令する. 他のサイトのクラスタマネージャノードに対して仮想クラスタの構築を要請し, また逆に他のサイトからの要請も受け付けることもできる. このとき, 同時にネットワーク帯域なども必要に応じて予約する. また, Rocks におけるフロントエンドとして動作し, PXE ブートサーバ機能により, 他の物理ノードを自動的に設定する. **ゲートウェイノード**および**計算ノード**上では仮想マシンモニタが存在して複数の仮想計算機が動作し, それらはクラスタマネージャの要求に応じて作成される. 本システムでは仮想クラスタ内部のシステムインストールにおいても Rocks を用いる. そのため, ゲートウェイノード上で作成された仮想計算機は仮想クラスタ内部において Rocks におけるフロントエンド (仮想フロントエンド) として動作する. つまり生成された仮想クラスタにおける管理ノードとなり, 仮想クラスタ内部に対して PXE ブートサーバ機能や NAT 機能を提供する. **ストレージノード**ではクラスタマネージャの要求に応じて論理ストレージ領域を作成し, iSCSI ターゲットサーバから仮想クラスタに対して提供する. サイト内において各仮想クラスタに対して割り当てられたネットワークは, タグ付き VLAN によって仮想クラスタごとに完全に分離されている.

ひとつのマルチサイト仮想クラスタは, 仮想フロントエンドが存在しその仮想クラスタにおいて中心的役割を果たす親サイトと WAN を介して接続する複数の子サイトからなる. ユーザからのクラスタ割り当て要求を親サイトのクラスタマネージャが受け取ると, その拠点内のクラスタに仮想フロントエンドを含む仮想クラスタを生成する. ユーザの要求に応じて, クラスタマネージャは他拠点のクラスタマネージャに対しても, その仮想クラスタの子サイトとして仮想クラスタを割り当てるように要求する. 要求が成立した場合, 子サイトの仮想クラスタ内部ネットワークを WAN 経由で延長し, 同時に必要に応じてネットワーク資源予約も行う. 親サイトにおける仮想クラスタ内部ネットワークは透過的に子サイトの仮想クラスタ内部ネットワークと相互に接続され, 子サイトの仮想ノードに対しても仮想フロンエンドからネットワークを介したシステムインストールが開始される.

4.2 仮想クラスタ内部ネットワークの拡張

各サイトごとに作成された仮想クラスタの内部ネットワークを拡張して互いに接続することで, それらをひとつの仮想クラスタとして統合可能にする (**図 3**). 各仮想クラスタの内部ネットワークは, 拠点内においてはイーサネットフレームに付随する VLAN タグによって識別される. そこで内部ネットワークの拡張機構では, ひとつのマルチサイト仮想クラスタごとに拠点間でそれぞれ仮想クラスタ内部ネットワークを相互接続し, 対応する拠点内仮想クラスタのイーサネットフレームを VLAN タグを取り除いた上で相互に転送する.

本システムで用いる仮想クラスタの内部ネットワーク接続には複数の選択肢が存在する. 商用の広域イーサネットサービスを用いたり, あるいは IP 網で L2TPv3[4] や EtherIP[5] 等に対応した VPN 装置や OpenVPN[6] や Vtun[7], PacketiX[8] などのソフトウェア VPN を用いたりもできる. ただし, クラスタマネージャと連係動作可能でありその接続を制御できる必要

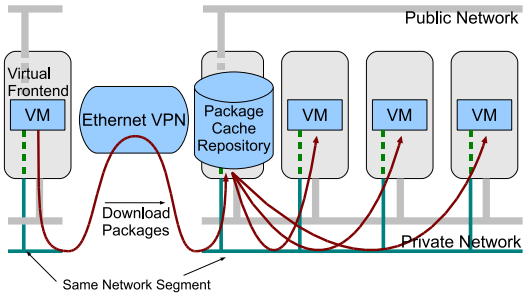

図 4 透過的パッケージキャッシング

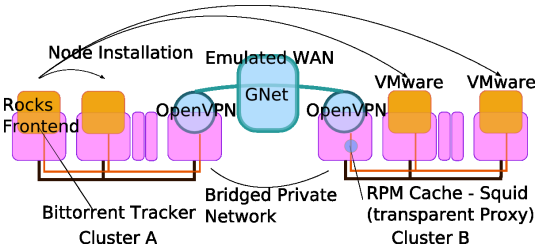

図 5 評価環境概要

表 1 実験ノード

クラスタ	ノード数	種類
A	16	AMD Opteron Processor 244 3GB memory, Gb Ethernet x2
B	134	AMD Opteron Processor 246 6GB memory, Gb Ethernet x2

がある．またネットワーク資源予約機構との連携などにより，拠点間に確立された接続において一定の品質保証が存在することが望ましい．

現状では無償で導入可能な OpenVPN をある程度の品質保証が可能な管理下の WAN 環境で用いている．ゲートウェイノード上で仮想クラスタごとに複数の VPN セッションを動作させることで，VLAN ネットワークインタフェースから VLAN タグなしのイーサネットフレームを取り出し転送できる☆．

4.3 透過的パッケージキャッシング

透過的なパッケージキャッシング機構の概要を**図 4** に示す．仮想フロントエンドが存在しない子サイトにおいて OpenVPN が動作しているゲートウェイノード上にキャッシュサーバを設ける．

仮想ノードの構築時には，仮想フロントエンドに対するすべてのパッケージ取得要求が，一旦ローカルサイトのキャッシュサーバを経由してなされる．キャッシュサーバは，未だキャッシュされていないパッケージに関しては仮想フロントエンドに取得要求を送る．VPN 経由でパッケージをダウンロードしながら要求元仮想ノードに対して徐々に取得データを転送する．転送完了後はキャッシュレポジトリ上にパッケージを保存しておく．キャッシュ済みのパッケージに関しては即座にキャッシュレポジトリから仮想ノードに渡す．

複数の仮想ノードから同時に同じパッケージに対する取得要求が送信されたときには，キャッシュサーバはそれらの要求を束ねて単一の取得要求のみを VPN 経由で仮想フロントエンドに送信する．つまり，ある拠点において複数仮想ノードの構築を VPN 経由で行う際にも，インストールされるパッケージが同一であれば，構築にともなう VPN 経由のデータ転送量は仮想ノード一台分に要するパッケージ総量まで低減される．

以上の仕組みは仮想フロントエンドおよび仮想ノード双方に透過的に実装され，Rocks インストーラを改変することなく実装される．基本的にパッケージレポジトリはセキュリティ上の理由から仮想クラスタごとに分けて管理される．しかし，よりパッケージのキャッシュ効率を高めるために，パッケージに施された電子署名を検証したのちに配布条件が許すものであれば仮想クラスタ間で共有可能であると考えられる．

Rocks 自体は仮想フロントエンドへのパッケージダウンロードトラフィックの集中を避けるために，仮想ノード間でダウンロードしたパッケージを直接的に転送する仕組みを持つ．しかし，その機能は WAN 経由での仮想ノード構築を行う際には不十分である．インストール中のノード同士でしか動作しないほか，250MB にもおよぶ初期システムイメージの転送には適用されない．また部分的にダウンロードされたファイルについても対象外となっている．我々のパッケージキャッシング機構は，拠点内ではこの仕組みが動作する余地を残しつつも，WAN 経由でのデータ転送量を効率的に低減することができる．

現在の実装では，キャッシング機構としてウェブプロキシサーバのひとつである Squid[9] を用いている．Squid は透過プロキシモードや単一 URL に対する同時並行的な取得要求の集約機能を備えており本システムに適している．Squid サーバは OpenVPN を起動しているゲートウェイノードに設置され，ノード構築中のパッケージ取得要求のみ対して動作させる．ノード固有の設定情報等を HTTP 経由で取得する際にはキャッシュしないよう設定されている．

5. 評　　価

提案手法によって複数サイトにまたがる仮想クラスタが構築可能であることを示すため評価実験を行った．我々が考えるマルチサイト仮想クラスタを使いこなすためには，ユーザが各ノードのカスタマイズを何度も容易に試行できる必要がある．そのため，たとえネットワーク遅延が存在し帯域が十分ではない WAN 環境においても，有限時間内で迅速に新たなノード設定で環境構築でき，できる限り WAN を経由するデータ転

☆ 例えば eth0.1 等の仮想クラスタごとの VLAN ネットワークインタフェースを直接イーサネット VPN の対象とできる．

送量を低く抑えられることを評価実験を通して確認する必要がある．

実験環境を**図 5** および**表 1** に示す．2 つの物理クラスタをそれぞれ遠隔拠点に存在するとみたてて, その間のネットワーク遅延を変更可能とするネットワークエミュレータ GtrcNet-1[10] を設置する．両物理クラスタ間には 1Gbps のネットワーク帯域を設定し, GtrcNet-1 により両物理クラスタ間のネットワーク遅延を適宜変更する．クラスタ A 上にはすでに仮想クラスタが構築されており, あらたに遠隔拠点のクラスタ B を追加して, OpenVPN を経由して両拠点にまたがる仮想クラスタを構築する際の所要時間を計測した．

クラスタ B の各仮想ノードは仮想 CDROM ブートによる起動後☆, クラスタ A 上の仮想フロントエンドノードから kickstart ファイルを取得する．その中には自動インストールに必要なすべての設定が含まれており, ユーザのさまざまなカスタマイズを反映してノードごとに動的に生成される☆☆．次にインストーラを含んだ 250MB の初期システムイメージを仮想フロントエンドから取得し, kickstart ファイルをもとに自動インストールを開始する．以後必要なパッケージの取得と展開を繰り返し, 最後に諸設定を施し再起動して完了する．本実験では各ノードに Globus Toolkit や Condor 等を含む 495 パッケージ (約 900MB) をインストールした．

図 6 は異なるネットワーク遅延の下での所要時間であり, パッケージキャッシュ機構が有効/無効である場合と, 事前にすべてのパッケージがキャッシュされている場合を示している．また, RTT20ms の実験時における, VPN 通信量およびクラスタ B の内部向けネットワークインタフェースの通信量を, **図 7**, **図 8** および**図 9** に示している☆☆☆．

パッケージキャッシュ機構が無効な場合, 初期イメージ取得の際にタイムアウトしてしまい, 実際には常に 15 ノード程度インストールに失敗している．また図 7 が示すように, 非常に大きなトラフィックが VPN を通過してしまい, バックボーンネットワークにかける負荷も大きい．本実験では OpenVPN は暗号化を無効にしバッファサイズを高遅延ネットワーク向けに最適化しているものの, イーサネットフレームのカプセル化においてカーネル・ユーザランド間での大量のメモリコピーをともなうため CPU 負荷が非常に高くなってしまう．そのため実験に用いた計算機性能においてはおおよそ 250Mbps 程度のスループットが限界であると考えられる．

☆ PXE ブートの代わりとしてカーネルイメージと `initrd.img` のみロードする．PXE ブートは UDP を用いた TFTP によるイメージ取得を行うものの, 特に大規模ノード構築においてパケットロスにより失敗することが多い．

☆☆ ただし高速化のため提案システムではあらかじめキャッシュする．

☆☆☆ グラフ中の縦線は構築完了時刻を示す．

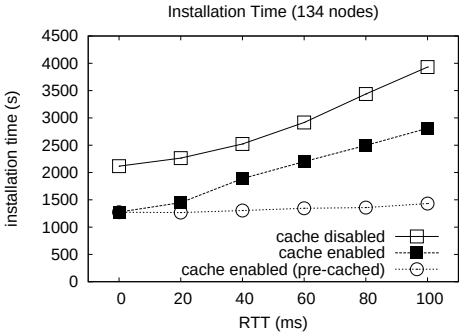

図 6 仮想クラスタ構築時間

パッケージキャッシュ機構が有効な場合, 無効な場合よりもおおよそ 800-1000 秒程度迅速に仮想クラスタ全体の構築が完了しており, 失敗したノードも存在しない．VPN スループットはおおよそ数 MB/s 程度であり, 1 ノードのインストールに必要なパッケージ分 (約 900MB) にまで転送量が低減されている．また, 高遅延環境下では仮想フロントエンドからのファイル取得に時間がかかるため, すべての仮想ノードがキャッシュサーバに接続して, 同時に同じファイルをダウンロードしている．このため, 内部向け NIC の通信量が大きい．

すべてのパッケージが事前にキャッシュされている場合は常に 1300 秒程度で構築が完了し, 高遅延下でも構築時間の増加は軽微である．また VPN スループットはピーク時でも 600KB/s 程度であり非常に小さい．初期イメージの取得以後は, ノード間パッケージ転送が働き, キャッシュサーバからはほとんど取得していない．

以上の結果より, OpenVPN による仮想クラスタ内部ネットワークの接続により WAN 環境下でも仮想クラスタのインストールが可能であることが確認できた．パッケージキャッシュ機構により VPN を経由するネットワークトラフィックを大きく低減させながら, 130 台あまりの大規模なノード数であっても確実に仮想クラスタ構築を完了させることができる．また高遅延環境下でもさらに迅速に構築を完了させるためには, パッケージの事前キャッシュが非常に有効であるとわかった．事前キャッシュせずともインストーラカーネルの TCP 送受信バッファサイズを十分大きくとればRTT の増加に対する構築時間の増加は若干緩和される．しかし, 平均パッケージサイズが小さく TCP/IP コネクションあたりの転送サイズが大きくないため効果は限定的となる．

6. 議　　論

マルチサイト仮想クラスタ中のすべての仮想ノードは単一の L3 ネットワークセグメント内に存在するゆ

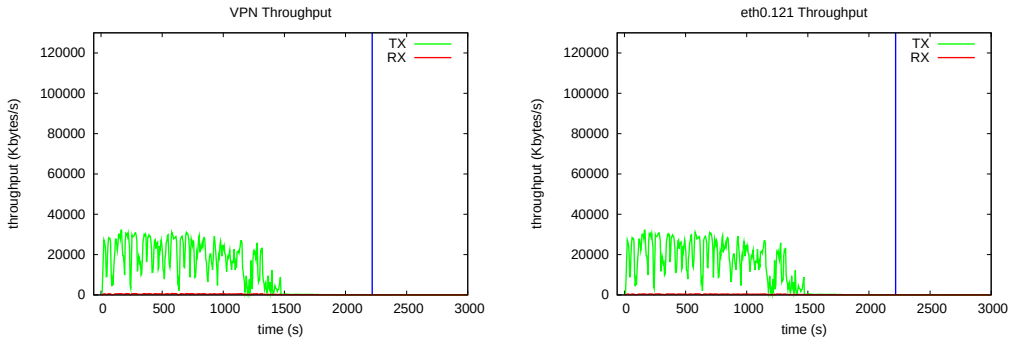

図 7 VPN 通信量およびクラスタ B の VPN ノードにおける内部向け NIC 通信量 (パッケージキャッシュ無効)

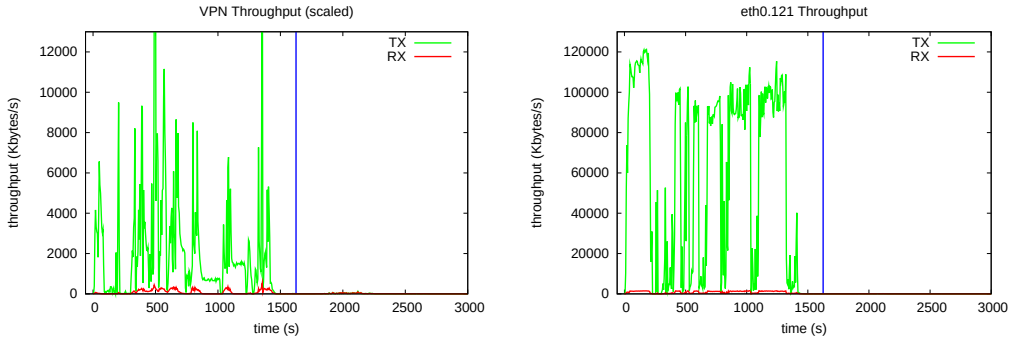

図 8 VPN 通信量およびクラスタ B の VPN ノードにおける内部向け NIC 通信量 (パッケージキャッシュ有効, 左図縦軸 12000Kbyte/s まで (scaled と表記))

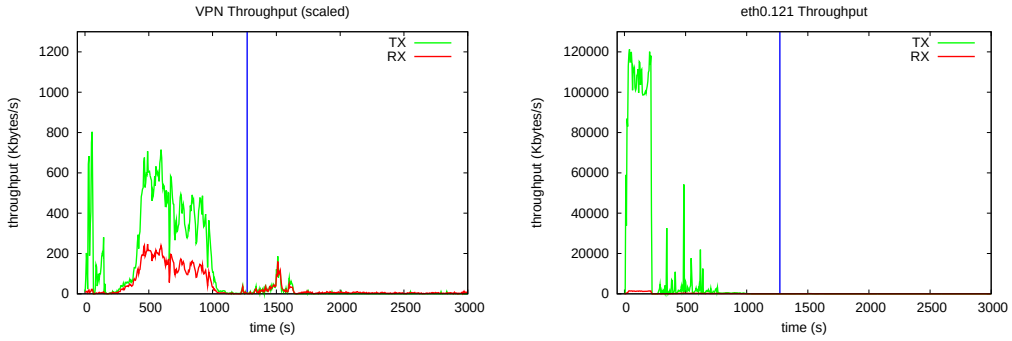

図 9 VPN 通信量およびクラスタ B の VPN ノードにおける内部向け NIC 通信量 (パッケージキャッシュ有効, キャッシュ済み, 左図縦軸 1200Kbyte/s まで (scaled と表記))

えに, 既存のクラスタ向けプログラムはほぼそのまま動作する. このことはユーザが分散する計算機資源を利用する上での敷居を低くしているといえる. しかしながら, ネットワークインテンシブなプログラムをそのまま動作させただけでは, WAN を経由することによる遅延や不十分な帯域による性能低下が生ずる恐れがある. 我々のシステムでは仮想ノードのホスト名や IP アドレス, MAC アドレスから存在する拠点を識別できるようにしており, ユーザがアプリケーション中のネットワークインテンシブ部分を各拠点内ごとに実行することを可能にしている. マルチキャストやブロードキャストを多用するプログラムのために, 新たに各拠点内に閉じた L3 ネットワークセグメントをマルチサイト仮想クラスタごとに割り当てることも可能である. 今後実アプリケーションを用いた性能評価をとおして詳細を検討していく.

マルチサイト仮想クラスタを構成する仮想ノードの構築はクラスタ構築ツールのもと常に自動で行われる

ため, あらかじめ実行プログラムを対応させておけば仮想クラスタの動的な伸縮も容易に可能になる. 例えばジョブスケジューラが仮想ノードリストの更新を検知しジョブのディスパッチや再実行を制御する. したがって, 拠点間をまたいだ仮想ノードの配置は実質的には可能であり, 運用における柔軟性はある程度有するといえる. しかし, 今後は, 仮想マシンモニタが備えるライブマイグレーション機能の適用を検討し, ユーザに対して極力透過的な仮想ノード再配置により柔軟性の向上を目指す.

7. 関連研究

仮想クラスタに関する研究はすでにいくつかなされているが[11)~14)], 複数サイトにまたがって単一イメージのクラスタを構築可能なものは少ない. Virtuoso[15)] はユーザの要求に応じて遠隔サーバに仮想計算機を立ち上げ, イーサネット VPN 経由でつなくごとでユーザの既存計算機環境の一部とする. 複数の仮想計算機からなる利用環境においては VPN トポロジを動的に変化させ最適化できるとうたっている[16)]. VioCluster[17)] では, 異なる管理ドメインに存在する仮想計算機群を VPN 経由でグルーピングしてその内部で並列計算ジョブを実行する. ジョブスケジューラのワークキュー状態から判断される計算資源需要に応じて, 各ドメインの資源貸借ポリシーのもと, 動的に仮想クラスタサイズを変更できる.

一方, 我々のシステムでは, 手元の物理クラスタと同様の使い勝手をもつ仮想クラスタを, WAN 環境において大規模に構築したうえで, ユーザがその内部に目的とするシステムを柔軟に構築できることに焦点を当てている. さまざまな拠点に分散する計算機資源をもとに構築するがゆえに, 使いこなすためにはノードごとにインストールするアプリケーションやその設定を調整する必要がある. 提案手法では, そのような決め細やかなカスタマイズを迅速なシステム構築機構のもと WAN を介するデータ転送量を低く抑えたまま何度でも試行することができる. そして, これらの関連研究が提示する VPN トポロジの動的な最適化機構や計算資源需要に応じた仮想クラスタサイズの動的な伸縮機構などは, 提案システムにおいても今後適用可能な仕組みであると考えている.

8. まとめ

複数サイトにまたがる仮想クラスタの構築手法について論じた. WAN を介して拠点横断的に仮想クラスタの内部ネットワークを拡張し, 単一実行環境の仮想クラスタを構築する. パッケージベースのクラスタ構築技術と透過的なパッケージキャッシュ機構により, 不可避な不均一性への対応としての決め細やかなノードカスタマイズを可能にしつつ, システムの迅速な配備を実現する. 評価実験によって, 提案構築手法が WAN 環境に大きなネットワーク負荷をかけることなく, 大規模なマルチサイト仮想クラスタを迅速に構築できることを確認した. 今後は Xen への対応を進めるとともに, 実アプリケーションによる評価を進める. また複数の研究拠点と連携した運用により実際的な検証に取り組む.

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, *Proc. of Cluster 2001: IEEE Int'l. Conf. 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コンピューティングシステム, Vol. ACS19, pp. 13–24 (2007).
3) Nakada, H., Yokoi, T., Ebara, T., Tanimura, Y., Ogawa, H. and Sekiguchi, S.: The Design and Implementation of a Virtual Cluster Management System, *Proc. of the first IEEE/IFIP Int'l. Workshop on End-to-end Virtualization and Grid Management (EVGM2007)* (2007).
4) Lau, J., Townsley, W. M. and Goyret, I.: Layer Two Tunneling Protocol - Version 3 (L2TPv3), RFC 3931 (Proposed Standard) (2005).
5) Housley, R. and Hollenbeck, S.: EtherIP: Tunneling Ethernet Frames in IP Datagrams, RFC 3378 (2002).
6) OpenVPN: `http://openvpn.net/`.
7) Vtun: Virtual Tunnels over TCP/IP networks: `http://vtun.sourceforge.net/`.
8) PacketiX VPN: `http://www.softether.com/`.
9) Squid: Optimizing Web Delivery: `http://www.squid-cache.org/`.
10) Kodama, Y., Kudoh, T., Takano, R., Sato, H., Tatebe, O. and Sekiguchi, S.: GNET-1: Gigabit Ethernet Network Testbed, *Proc. of Cluster 2004: IEEE Int'l. Conf. on Cluster Computing* (2004).
11) Foster, I., Freeman, T., Keahy, K., Scheftner, D., Sotomayer, B. and Zhang, X.: Virtual Clusters for Grid Communities, *Proc. of the Sixth IEEE Int'l. Sympo. on Cluster Computing and the Grid (CCGRID2006)* (2006).
12) 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, *Proc. of the ACM/IEEE Supercomputing 2004 Conf.* (2004).
13) Nishimura, H., Maruyama, N. and Matsuoka, S.: Virtual Clusters on the Fly - Fast, Scalable, and Flexible Installation, *Proc. of the 7th IEEE Int'l. Sympo. on Cluster Computing and the Grid (CCGrid 2007)* (2007).
14) McNett, M., Gupta, D., Vahdat, A., and Voelker, G. M.: Usher: An Extensible Framework for Managing Clusters of Virtual Machines, *Proc. of the 21st Large Installation System Administration Conf. (LISA 2007)* (2007).
15) Shoykhet, A. I., Lange, J. and Dinda, P. A.: Virtuoso: A System For Virtual Machine Marketplaces, Technical Report NWU-CS-04-39, Northwestern University (2004).
16) Sundararaj, A. I. and Dinda, P. A.: Towards virtual networks for virtual machine grid computing, *Proc. of the 3rd Conf. on Virtual Machine Research And Technology Sympo. (VM'04)* (2004).
17) Ruth, P., McGachey, P. and Xu, D.: VioCluster: Virtualization for Dynamic Computational Domains, *Proc. of Cluster 2005: IEEE Int'l. Conf. on Cluster Computing* (2005).