キャッシュを用いた仮想クラスタ高速構築手法の性能評価

西村豪生[†] 丸山直也[†] 松岡聡[†,††]

近年グリッド上の大規模資源を仮想計算機を用いて仮想化し，仮想クラスタを構築して効率的に共有する手法が提案されている．そのような大規模資源共有のための仮想クラスタは，環境を柔軟にカスタマイズ可能でありながら，数百台から千台規模でも高速に構築できなくてはならない．そこで我々は，キャッシュイメージを過去の要求履歴から自動生成する高速でスケーラブルな仮想クラスタ構築機構を提案している．本稿では，提案高速化手法についてより多くの条件下でキャッシュイメージ作成前後の構築時間推移を評価し，作成後の平均構築時間が作成前に対して最大 66.7%程度減少したことを確認した．また，構築時間のスケーラビリティについての評価を行い，204 ノードが 40 秒以内で構築可能であったことから，千台規模の仮想クラスタでも本手法によって数十秒以内で構築可能であるという知見を得た．

Performance Evaluation of a Cache-Based Virtual Cluster Installation Method

Hideo NISHIMURA ,[†] Naoya MARUYAMA [†] and Satoshi MATSUOKA [†,††]

Recently, clusters of virtual machines called virtual clusters are proposed as a means to share Grid resources efficiently. Such virtual cluster construction should be not only fine-grained customizable but also fast and scalable. However, existing ways have not fullfilled these requrirements. We have been proposing a novel virtual cluster installation system which is fast, scalable and fully-customizable in corporation with existing cluster installer tools. To achieve efficiency in the presence of such full customization, it automatically caches frequently-constructed virtual disk images to save software installation time in common cases. On broader environments, our experimental studies show that the average installation time could be reduced by approximately 66.7% after creation of cache images and 204-node virtual cluster can be done in 40 seconds with our prototype implementation. From the result along with a scalability study, we estimate that installation of a 1000-node virtual cluster could be done in several tens of seconds.

1. はじめに

近年グリッドによる広域分散計算が注目されており，従来の科学技術計算などの専門分野だけでなく様々な応用分野に普及しつつある．グリッド資源を大多数で効率的に共有する手法としては，実資源上に仮想クラスタを構築する手法が提案されている[1)]．仮想クラスタとはグリッドを構成する実資源上で動作する仮想計算機 (VM) を仮想ネットワークで接続したものであり，資源の仮想化によってグリッド環境の不均質性を隠蔽し，ユーザに通常のクラスタのユーザビリティを提供する．

† 東京工業大学
Tokyo Institute of Technology
†† 国立情報学研究所
National Institute of Informatics

仮想クラスタ内の環境は，ユーザが実行したいジョブに必要な環境を備えている必要がある．しかし，グリッドの利用分野の拡大に伴ってユーザが必要とするOS やソフトウェアなども急速に多様化しているため，全てのユーザの要求を満たす VM イメージを管理者が事前に用意することは非現実的である．よって，ユーザは仮想クラスタ構築システムを用いて任意の環境を柔軟かつ容易にカスタマイズできることが必須である．また，ユーザがグリッドのような大規模資源の分散を意識せずに単一クラスタかのように利用するためには，数百から千台規模のスケールで仮想クラスタ構築が数十秒以内に高速に行われなくてはならない．しかし，既存の仮想クラスタ構築システム[2),3)] は必ずしも以上の要件を満たしていない．

そこで我々は，クラスタインストーラツールとパイプライン転送を用いた高速でスケーラブルな仮想クラスタ構築システムを提案している[4)]．本システムは過

```
[Hardware]
NumberOfNodes: 32
CPUArch: x86
CPUSpeed: >= 2GHz
Disk: 2GB
Memory: 256MB

[Software]
# Specify packages
Package: ruby gcc g++ make mpich ...
Hostname: vpc%02d
Address: 192.168.1.128/32
```

図 1 仮想クラスタ構築要求の例

去のパッケージ要求履歴を解析して頻出するパッケージ要求の組合せを抽出し，それらをあらかじめインストールしたキャッシュイメージを自動生成することによって構築を高速化している．

本稿では，より様々な条件下で提案高速化手法の評価を行い，その有効性を示した．プロトタイプ実装を用いて，パッケージ要求の傾向が均質である場合と変化する場合のサンプル要求について，それぞれ構築時間推移をキャッシュイメージ作成前後で比較した．その結果，作成後の平均構築時間が作成前に対して最大 66.7%減少したことを確認した．また，構築時間のスケーラビリティについて，204 ノードが 40 秒以内で構築可能であったことから，千台規模の仮想クラスタでも本手法によって数十秒以内で構築可能であるという知見を得た．

2. キャッシュを用いた仮想クラスタ高速構築機構

本稿で評価した，我々が開発している仮想クラスタ構築システム[4] について紹介する．なお，提案手法はキャッシュイメージを用いた構築高速化に重点を置いているため，以下では本システムを用いるためのインタフェースの詳細については省略する．

2.1 仮想クラスタ構築システムの概要

本システムは，従来のクラスタインストーラツール[5),6)] と連携して VM 内の環境をユーザの要求に応じて動的に自動インストールする．ユーザは図 1 で示すような抽象化された形式で CPU 性能や RAM 量などのハードウェアスペックと，インストールしたいソフトウェアパッケージなどを記述する．インストール機構として既存のクラスタインストーラツールを用いているため，ユーザは従来のクラスタ管理手法と同等の負担で仮想クラスタ環境のカスタマイズを行うことが出来る．

システムの概要を図 2 に示す．本システムはユーザ

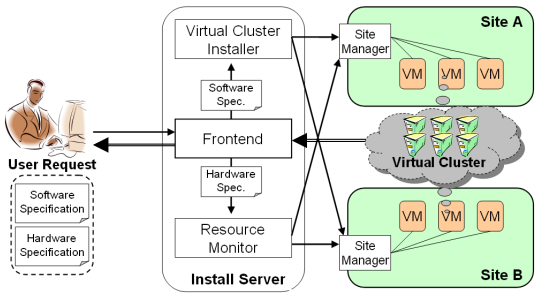

図 2 システムの概要図

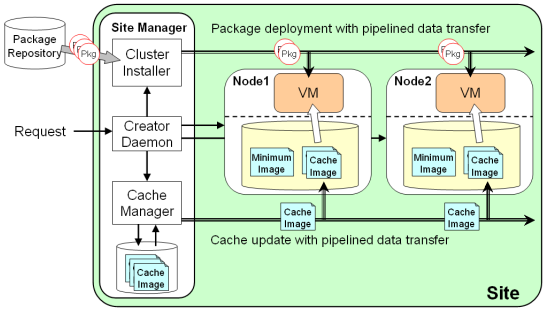

図 3 サイト内インストールの概要図

からの構築要求を受け付ける仮想クラスタ構築サーバと，実際に仮想クラスタを構成する計算資源となる複数のサイトから構成される．システムのフロントエンドはユーザからの要求を受け取ると，ハードウェアスペックの要求を満たす計算ノードを有するサイトを選択し，構築を要求する．また，抽象化されたユーザのソフトウェアスペック要求を各サイトのクラスタインストーラが利用できる形式へと変換して転送する．構築要求を受けた各サイトの Site Manager はサイト内の計算ノード上で VM を起動し，ローカルのクラスタインストーラを用いて環境のセットアップを行う．

各サイトでの VM 内環境のセットアップは，頻繁に用いられるパッケージが事前に含まれたキャッシュイメージからの差分インストールによって行われる．図 3 にサイト内インストールの概要を示す．Site Manager はクラスタインストーラとキャッシュマネージャから成り立っている．デーモンプロセスが仮想クラスタ構築サーバから構築要求を受け取ると，キャッシュマネージャは最適なキャッシュイメージを選択し，計算ノードへとパイプライン転送する．なお，一度計算ノードへ転送されたキャッシュイメージはローカルのイメージレポジトリに保管され，次回以降は転送する必要はない．選択されたキャッシュイメージから VM が起動されると，クラスタインストーラは必要なソフトウェアパッケージを外部のパッケージレポジトリからダウンロードし，各 VM に対してパイプライン転送して差分インストールを行う．

2.2 キャッシュイメージを用いた高速化

ユーザが仮想クラスタに対して要求するパッケージ構成が多種多様であっても，その中には頻繁に用いられるパッケージの組合せがあると考えられる．本システムは，過去の仮想クラスタ構築のパッケージ要求履歴から，そのような頻出するパッケージの組合せを抽出してキャッシュイメージを自動生成する．適切なキャッシュイメージを用いることにより，仮想クラスタ構築時間の中で支配的であるパッケージインストール時間を大幅に削減することが可能である．

2.3 キャッシュイメージ生成アルゴリズム

過去のパッケージ要求集合履歴を階層的クラスタ解析によってグルーピングし，各グループを優先度によって順位付けしてキャッシュイメージを生成する．優先度は今後の構築時間がそのキャッシュイメージによってどれだけ削減されるかの期待値であり，そのキャッシュイメージが今後用いられる頻度と用いられた際に削減される構築時間の積としてモデル化した．削減される構築時間はそのキャッシュイメージが含んでいるパッケージの容量に比例する．将来どれだけ用いられるかの頻度は，今後のパッケージ要求も分析対象の要求履歴と同様の傾向を持つとの仮定から，要求履歴の中での出現頻度とした．

本システムでは，パッケージ集合間の距離関数をそれらの共通パッケージ容量の逆数とした．Fortran, Python パッケージの容量をそれぞれ 1.6 メガバイト，3.4 メガバイトとして，パッケージ要求集合 $r_1\{Fortran, Python\}$，$r_2\{Python\}$，$r_3\{Fortran\}$ をクラスタ解析する例を示す．図 4 がクラスタ解析によって生成される樹状図である．共通パッケージ容量が最も多い r_1 と r_2 の距離が最小の 0.36 となり，最初にグルーピングされる．その後残った r_3 とグルーピングされるが，r_1, r_2 と r_3 の共通パッケージは存在しないため，距離は ∞ となる．図 4 の例では $\{r_1\}, \{r_2\}, \{r_3\}, \{r_1, r_2\}, \{r_1, r_2, r_3\}$ の優先度を計算し，$\{r_1, r_2\}$ の $3.4 \times (2/3) = 2.27$ が最大となるのでキャッシュイメージとして採用される．一つのグループがキャッシュイメージとして選ばれると，その下位にあたるグループの共通パッケージ容量と上位にあたるグループの出現頻度を修正して優先度を再び計算し，容量が許す限りのキャッシュイメージを生成する．アルゴリズムの詳細については文献[4] を参照する．

2.4 最適なキャッシュイメージの選択

キャッシュイメージを用いた場合に削減される構築時間はパッケージ容量に比例するため，構築時には要求パッケージ集合との共通パッケージ容量が最も大きいものを最適なキャッシュイメージとしてとして選択する．なお，キャッシュイメージが含むパッケージ集合は要求パッケージ集合の部分集合でなくてはならない．

次にその最適とされたキャッシュイメージを用いるか，最小構成イメージからインストールするかの判断を行う．選択されたキャッシュイメージについて，削減される構築時間とキャッシュイメージ自身を各計算ノードへ転送するのに必要な時間を過去の履歴から予測する．予測は過去の履歴をノード数，パッケージ容量をパラメータとして重回帰分析することによって行う．各ノードへの転送時間に対して削減される構築時間の方が大きかった場合のみ，そのキャッシュイメージを用いて差分インストールを行う．それ以外の場合は計算ノードへ配備済みの最小構成イメージから構築する．

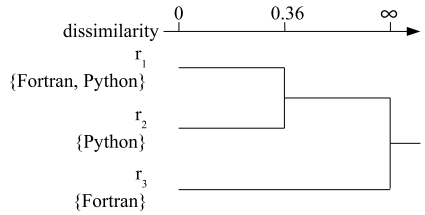

図 4 階層的クラスタ解析によって生成される樹状図

表 1 PrestoIII クラスタのノード構成

	CPU	RAM	HDD
構成 0	Athlon2000+	1GB	IDE
構成 1	Opteron242	2GB	IDE
構成 2	Opteron280	4GB	SATA
構成 3	Opteron250	2GB	SCSI

3. 性 能 評 価

3.1 評 価 項 目

本稿では以下の項目について評価を行い，提案仮想クラスタ高速構築手法の有効性を示す．

(1) 作成前後の構築時間推移
ある傾向を持つパッケージ要求に対し，一定数の要求が蓄積されるたびに提案手法によってキャッシュイメージを作成する．キャッシュイメージ作成前後の構築時間を比較することにより，提案手法による構築時間削減効果を評価する．

(2) 要求傾向が変化した場合
一定数毎にパッケージ要求傾向が変化した場合の提案手法による構築時間削減効果を評価する．

(3) スケーラビリティ
ノード数を変化させた場合の提案システムによる構築時間変化を計測することにより，提案システムのスケーラビリティを評価する．

3.2 評 価 環 境

評価実験は仮想計算機モニタとして Xen[7] バージョン 3.0.2-2，クラスタインストーラツール Lucie[5] バージョン 0.0.5 を用いたプロトタイプ実装によって行った．プロトタイプ実装は，パイプライン転送として Dolly+[8] バージョン 0.93-1 と MPICH[9] バージョン 1.2.7-p1 のブロードキャストを用途に応じて使い分

表 2　コアパッケージの分類と含まれるパッケージサイズ

	ミドルウェア		アプリケーション	
	Roll	容量 (MB)	Roll	容量 (MB)
a	Grid	46.0	Bio	61.0
	HPC	9.09	APBS	17.3
b	Condor	64.5	Numerics	29.5
	PBS	2.17	Intel	43.6

ける．

評価環境としては本研究室の PrestoIII クラスタを用いた．PrestoIII クラスタのスペックは表 1 の通りである．Site Manager として構成 0 を用い，計算ノードして構成 1〜3 を用いた．Site Manager は Linux-2.6.12.6，計算ノードは Xen パッチ済みの Linux-2.6.16 でそれぞれ稼働しており，全てのノードは Gigabit Ethernet ネットワークで接続されている．

3.3　サンプル要求

評価実験に用いるサンプルパッケージ要求集合としては，NPACI Rocks[6] における機能別のソフトウェアパッケージ集合である Roll[10] を参考にした．8 つの Roll を表 2 に示すようにミドルウェアとアプリケーションの 2 種類に分類し，それぞれから一つずつ選んだ $4 \times 4 = 16$ 通りの組合せをコアパッケージとした．それに加え，クラスタ環境にインストールされることが想定される 10 種のエキストラパッケージの中から 2 種を選択した $4 \times 4 \times {}_{10}C_2 = 720$ 通りをサンプル要求とした．なお，エキストラパッケージは 13.8 キロバイトから 17.4 メガバイトの範囲である．

3.4　キャッシュイメージ作成前後の構築時間推移

3.4.1　実 験 概 要

720 のサンプル要求からランダムに選択した 200 の要求に対し，50 要求毎に提案手法によってキャッシュイメージを作成し，構築時間推移を計測した．キャッシュイメージ，パッケージの転送については共に Dolly+ を用いた．計算ノードとしては表 1 の構成 2, 3 による計 50 ノードを用い，VM スペックとしてメモリ 256 メガバイト，ディスク 2 ギガバイトを要求した．また，作成するキャッシュイメージ容量の総計を 5 ギガバイト以内とした．

3.4.2　実 験 結 果

図 5 に構築時間の推移を示す．横軸の補助線がキャッシュイメージが生成された位置を示す．キャッシュイメージ生成後の平均構築時間は，作成前に対して 42.3%である 30.4 秒減少した．また，生成前は構築に最小 43 秒，最大 148 秒程度要していたのに対し，キャッシュイメージを用いてインストール (キャッシュヒット) した場合は最小で 27 秒，最大でも 63 秒程度で構築された．なお，本実験における生成後のキャッシュヒット率は 94.7%であった．

3.5　要求傾向が変化した場合の構築時間推移

キャッシュイメージを作成した直後から，それまでとは異なった傾向のパッケージが要求された場合の構築時間推移について評価する．

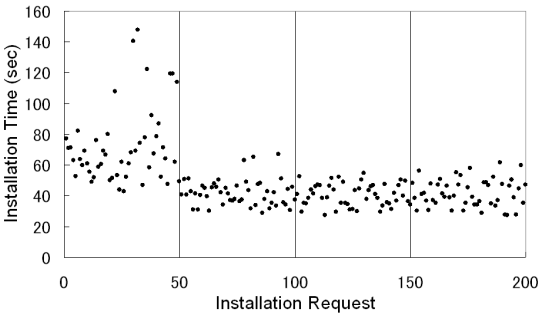

図 5　キャッシュイメージ作成前後の構築時間推移

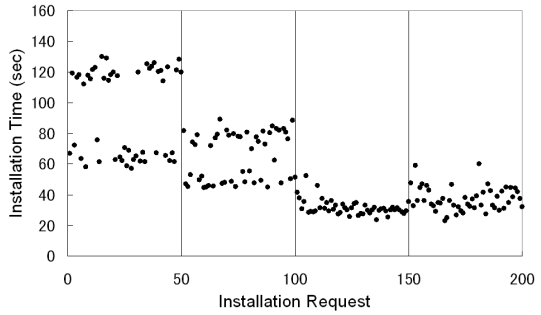

図 6　パッケージ要求傾向が変化する場合の構築時間推移

3.5.1　実 験 概 要

コアパッケージをさらに表 2 に示すように a, b 二つに分類する．720 のサンプル要求の中から，a の Roll しか含まないグループ A と，b しか含まないグループ B を抽出する．エキストラパッケージを追加することで，グループ A, B 共に $2 \times 2 \times {}_{10}C_2 = 180$ 通りのパッケージ要求を含む．

グループ A, B に属するパッケージの組合せを 50 要求毎に交互に要求し，200 通りについて構築時間を測定した．パッケージ要求以外の条件は 3.4 と同様である．

3.5.2　実 験 結 果

図 6 に構築時間の推移を示す．グループが切り替わった直後の 51-100 試行でのキャッシュヒット率は 0%であったが，二つのグループの履歴が蓄積された 101 試行以降はキャッシュヒット率が 100%であった．

グループ A については 1-50 試行と 101-150 試行，グループ B については 51-100 試行と 151-200 の平均構築時間をそれぞれ比較する．グループ A は 66.7%である 63.2 秒，グループ B は 43.0%である 28.1 秒平均構築時間が減少した．

キャッシュイメージ作成直前と直後の傾向が完全に異なっている場合は提案手法によって生成されたキャッシュイメージによって高速化は望めない．しかし，直前の傾向とは異なっていても分析対象の履歴内に同様の傾向が存在すれば，提案手法によって適切なキャッ

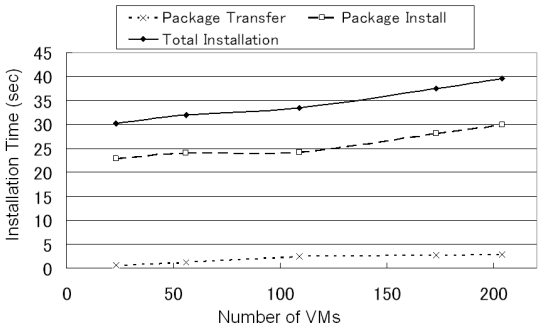

図 7 ノード数の増加による構築時間変化

シュイメージが生成され，構築時間が削減されることが確認された．

3.6 システムのスケーラビリティ

ノード数の増加に伴う構築時間増加を計測し，提案システムのスケーラビリティを評価する．

3.6.1 実験概要

表 1 の構成 1〜3 からディスク性能が良いノードを優先して選択し，23〜204 ノードについての構築時間を計測した．キャッシュイメージが各計算ノードのイメージレポジトリに保管されているとして，差分の 5.8 メガバイトをインストールする．なお，パッケージの転送については MPICH のブロードキャスト通信を用いた．VM のスペックとしてはメモリ 256 メガバイト，ディスク 2 ギガバイトを要求した．

3.6.2 実験結果

図 7 に結果を示す．Total Installation が全体の構築時間であり，その内訳であるパッケージ転送時間とインストール時間がそれぞれ Package Transfer, Install を示す．23 ノードの仮想クラスタが 30 秒程度，204 ノードが 40 秒程度で構築された．

構築時間はノード数の増加に伴って増加しているが，ノード数の増加に伴って増加しているのはパッケージ転送ではなく，各計算ノードで並列に行われるパッケージインストール時間であることがわかった．本来ノード数の増加に対して独立であるインストール時間が増加している原因は，ノード数の増加によってディスク性能が悪いノードも含まれるようになり，そのノードが全体の構築時間のボトルネックとなっているからであると考えられる．よって，ノードの性能が均質である環境であれば，本システムは高度にスケーラブルであると言える．

4. 議　　論

本節では前節の評価実験の結果を考察する．

4.1 構築時間の内訳

図 8 に，3.4 の評価実験において，最小構成イメージからインストールした場合とキャッシュヒットした場合についての構築時間内訳を示す．キャッシュヒットした場合，キャッシュイメージを選択する時間がやや増加しているが，支配的であったパッケージの取得，転送，インストール時間が大きく減少していることがわかる．

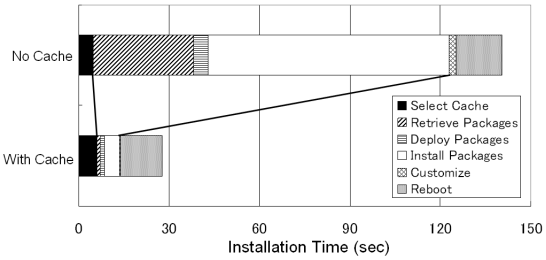

図 8 構築時間の内訳

キャッシュヒットした場合の内訳を見ると，再起動時間とキャッシュイメージの選択時間が支配的である．プロトタイプ実装では毎回過去の履歴を重回帰分析しているが，十分な履歴が存在する場合回帰式は毎回大きく異なることはない.，そこで，ある量の履歴が蓄積された時のみ回帰分析を行うことにより，キャッシュイメージの選択時間は削減することができる．また，再起動は Lucie によるカスタマイズ終了後に，インストール対象のパーティションをルートファイルシステムとするために必要となる．これは単純に VM, OS の再起動をするのではなく，`chroot`，`pivot_root` を用いることによって高速化が期待できる．

以上から，プロトタイプ実装のパッケージインストール以外の部分を高速化することにより，提案手法によって最速 20 秒以内の仮想クラスタ構築が可能であると予想される．

4.2 千台規模へのスケール

3.6 におけるスケーラビリティの評価結果から，本システムによって 1000 ノードの仮想クラスタを構築する場合を考える．図 7 のペースで単一ノードのインストール時間が増加すると仮定すると，本提案システムによる 1000 ノードの仮想クラスタ構築に 80 秒程度を要する．一方，ノード性能が均質であり各ノードにおけるインストール時間がノード数の増加に対して一定であると仮定すると，45 秒程度で構築可能である．

以上から，提案システムは千台規模の仮想クラスタを数十秒以内で構築可能という，大規模資源共有のための仮想クラスタ構築システムへの要件を満たしていると言える．

5. 関連研究

仮想クラスタをグリッド上に構築し，ジョブ実行環境として用いる手法はいくつか提案されている．

VMPlants[2] は，カスタマイズ処理と依存関係を有

向非巡回グラフを用いて定義することにより，柔軟に環境をカスタマイズ可能な仮想クラスタ構築機構である．また，事前に用意された頻繁に用いられるゴールデンイメージとの部分マッチングを行い，差分処理のみを行うことで構築の高速化をはかっている．しかし，ゴールデンイメージは事前に管理者が用意しなければならず，多種多様なユーザが存在する環境ではそのようなイメージを特定するのは現実的でない．また，VMPlants は数百メガバイトから数ギガバイトに及ぶ VM イメージの転送に NFS を用いているため，ノード数の増加にスケーラブルでない．

Virtual Cluster Workspaces[3] は Globus プロジェクト[11] の仮想クラスタを用いたジョブ実行環境提供機構であり，グリッドミドルウェアのデファクトスタンダードである Globus Toolkit に対して優れた相互運用性を持つ．しかし，事前に用意されたディスクイメージを単純にステージングして仮想クラスタを構築するためカスタマイズ性を持たず，またノード数に応じて構築時間も線形に増大する．我々の提案手法を用いることにより、それらの問題を解決できると考える．

6. おわりに

6.1 まとめ

本稿では，過去の履歴を統計的に分析してキャッシュイメージを自動生成する高速な仮想クラスタ構築手法の評価を行った．サンプル要求を用いた評価結果から，提案手法によって適切なキャッシュイメージが生成され，キャッシュイメージ作成後の平均構築時間が生成前に対して最大 66.7%減少することを確認した．また，23〜204 ノードの仮想クラスタを構築することによって提案システムのスケーラビリティを評価し，204 ノードが 40 秒程度で構築されることを確認した．以上から，1000 ノード規模の仮想クラスタも提案システムによって数十秒以内で構築可能であるため，提案手法が大多数のユーザによる大規模資源共有に対して有効であることを示した．

6.2 今後の課題

今後の課題として以下が挙げられる．

- 計算資源のスケジューリング
  現在，仮想クラスタ構築サーバにおけるサイト選択とサイト内の資源選択は，ハードウェアスペックを満たしているかの観点のみで行われる．各サイトに生成されているキャッシュイメージを考慮したサイト選択を行い，また各ノードの性能を監視して高性能のノードを優先的に選択することにより，ユーザの要求を満たしながら構築時間を最小にすることが出来る．
- 精緻な予測
  現在，構築時間の予測はパッケージ容量とノード数のみをパラメータとして回帰分析をしている．しかし，CPU やディスク性能などより多くのパラメータから主成分を抽出し，より精緻な予測をすることが必要である．
- インストール時間以外の構築時間削減
  より透過的にユーザにクラスタ資源を提供するために，キャッシュイメージ選択や再起動などのパッケージインストール以外の部分を高速化し，構築時間をさらに短縮することが望ましい．

謝辞 本研究の一部は科学研究費補助金特定領域研究 (18049028) の補助による．

参考文献

1) Figueiredo, R. et al.: A Case For Grid Computing On Virtual Machines, *Proc. of the 23rd IEEE Int'l Conf. on Distributed Computing Systems (ICDCS'03)* (2003).
2) Krsul, I. et al.: VMPlants: Providing and Managing Virtual Machine Execution Environments for Grid Computing, *Proc. of the 2004 ACM/IEEE Conf. on Supercomputing (SC'04)* (2004).
3) Foster, I. et al.: Virtual Clusters for Grid Communities, *Proc. of the 6th IEEE/ACM Int'l Symp. on Cluster computing and the Grid 2006 (CCGrid'06)* (2006).
4) Nishimura, H. et al.: Virtual Clusters on the Fly — Fast, Scalable, and Flexible Installation, *Proc. of the 7th IEEE/ACM Int'l Symp. on Cluster computing and the Grid 2007 (CCGrid'07)* (2007). (to appear).
5) 高宮安仁ほか: 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 (2003).
6) Papadopoulos, P. et al.: NPACI Rocks: tools and techniques for easily deploying manageable Linux clusters, *Concurrency and Computation: Practice and Experience* (2003).
7) Barham, P. et al.: Xen and the Art of Virtualization, *Proc. of the ACM Symp. on Operating Systems Principles (SOSP)* (2003).
8) Manabe, A.: Disk cloning program 'Dolly+' for system management of PC Linux cluster, *Proc. of the Computing in High Energy and Nuclear Physics 2001 (CHEP'01)* (2001).
9) MPICH Home Page. http://www-unix.mcs.anl.gov/mpi/.
10) Bruno, G. et al.: Rolls: Modifying a Standard System Installer to Support User-Customizable Cluster Frontend Appliances, *Proc. of the 2004 IEEE Int'l Conf. on Cluster Computing (CLUSTER'04)* (2004).
11) Globus Alliance. http://www.globus.org/.