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

高宮安仁[†] 山形育平[†] 青木孝文[†]
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ともない，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，グリッドリソースであるクラスタごとの管理ポリシによる制限により，ジョブの要求を満たす実行環境を得ることは難しい．既存研究では実行環境の仮想化による実行環境の提供をある程度達成しているものの，セットアップできる環境の種類が制限されていることや，セットアップが自動的でないといった問題や，システムの詳細に関する知識を必要とするといった問題があった．そこで我々は，VM 技術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として ORE（Open Resource Environment）グリッドを提案する．加えて，スクリプトや DAG などツール独自の方法で実行環境の構築手順を記述する代わりに，GUI を用いて実行環境に必要な要素を指定することで容易に実行環境を構築できる．評価では 16 台構成の実行環境を ORE グリッドを用いてセットアップしジョブを実行させた．結果，実行環境の構築に要する時間は全体で 151 秒と高速であり，一般的なグリッド上のジョブ実行時間（数時間〜数日間以上）と比較すると十分許容範囲以内であることを確認した．

ORE Grid: A Virtual-machine Based Fast Deployment Tool for Grid Execution Environment

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

With the increased variety of jobs executed in the Grid, the execution environments such as OSes, softwares, and libraries requested by such jobs have becoming increasingly diversified. However, it is difficult for grid users to acquire the necessary environment suited for each jobs because the job execution environment on the grid are strongly tied to its local administration policies. Recently proposed solutions may achieve virtualization of execution environment at certain level, but are still incomplete that construction of execution environments will again requires manual operations and/or expert knowledge of underlying systems. Instead, we propose the system called ORE (Open Resource Environment) Grid which automatically and dynamically builds exclusive execution environment for each submitted jobs. Moreover, the GUI setup front-end offers succinct methods to pick the necessary features and generate an execution environment description automatically instead of resorting to tool-dependent VM description forms such as shell scripts or DAG descriptions. Our experiences have shown that setup of 16 VM nodes itself will only take 151 seconds, and the setup cost is certainly within an allowable range compared to accumulated running time of general Grid jobs (several hours to several days).

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をもとに，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]．応用分野での利用用途として，大規模な解析を必要とする高エネルギー物理学や天文学，および生物学などでの利用が増えつつある．こう

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

したグリッドの浸透にともないグリッド上で実行されるジョブは急激に多様化している．この結果，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．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に強く依存するため，ジョブ実行に必要な環境を得ることは難しい．

そこで，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手法が考えられる．この方法では，ユーザはジョブの実行に必要なジョブ仕様に加えて，ジョブを実行するのに必要な実行環境仕様を計算リソースの管理者へ渡す．管理者はこの仕様に従って実行環境をインストールしユーザへ提供する．しかし，この手法には以下の 2 つの問題がある．

(1) ジョブの実行ごとに OS やソフトウェアを入れ替えることは，ローカルユーザなどのほかに計算リソースを共有しているユーザにとって影響が大きい．また，こうした利用形態は管理ポリシや他のユーザの利用状況に強く依存するため，多くの場合不可能である．

(2) ジョブの実行要求ごとに OS やソフトウェアを入れ替えることは管理コストが大きい．とくに相同性検索など Embarrassingly Parallel 型のジョブでは計算ノードを多数要求されることが多いため，多数のノードへのセットアップ作業はコストが高く，作業上のミスを犯しやすい．

このため，ジョブ実行に必要な実行環境を管理ポリシに依存せず，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．

問題 (1) を解決する手段として，仮想マシン[3] を用いてグリッドを構築する手法が提案されている．仮想マシンは物理マシンを抽象化しジョブ実行専用環境のイメージをユーザに提供する．この抽象化はグリッドコンピューティングにとって非常に強力である．なぜならば，ユーザは下層に存在するリソースやリソースを共有している他のユーザからはっきりと分離されるためである．このため，ユーザは物理計算機上の OS とは独立した VM イメージをジョブ実行専用環境として占有し，VM 内の OS すべてを設定することができる．

問題 (2) を解決する手段として，実行環境を自動インストールおよび設定するためのツールがいくつか提案されている[4),5]．これらのツールでは，実行環境の仕様をあらかじめ設定ファイルとして与えることによって，（複数台の）実行環境構築を自動的に並列化して行う．

本研究では，こうした仮想マシンによるグリッド構築技術と，システムの自動インストール・設定技術を組み合わせることにより，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（**O**pen **R**esource **E**nvironment）グリッドツールを構築した．ORE グリッドでは実行したいジョブとジョブ実行に必要な実行環境記述を ORE サービスに渡すことによって，動的に専用の仮想環境が構築されその上でジョブが自動実行される．仮想環境の自動構築技術として我々が開発している自動インストール・設定ツール Lucie を用い，GUI によるジョブ実行環境の構成指定や仮想計算機の自動インストールを実現した．またグリッドでのリモートジョブ機構である GRAM を拡張し，仮想環境構築ジョブの投入を可能にした．

また，既存の類似システム[6)~8)] との機能面での比較を行った．既存システムでは構築する実行環境をユーザが指定できないという制限[6)] や，サポートする実行環境として単一の仮想計算機実装や OS，ディストリビューションのみをサポートしており，実行可能なジョブの種類に制限があった[7),8)]．また，設定機構が独自の方式をとっておりシステムの詳細に関する知識を要したり[7)]，ソフトウェアの追加的なインストールや設定を手動で行う必要があったりするなど[8)] 煩雑であった．一方，我々のシステムでは使用する仮想計算機の実装やセットアップする OS，ディストリビューション，インストールするソフトウェアなど，幅広い環境をサポートしている．また，これらを設定するための GUI を提供しており，ユーザは容易に実行環境をセットアップすることができる．

性能評価では実際に本システムを用いて，グリッド上で実行される代表的なアプリケーションであり相同性検索プログラムの一種である BLAST[9)] の実行環境を構築し，BLAST の一般的なジョブ実行時間と比較しジョブ実行時間全体に与える影響を考察した．また，類似システムとの性能比較を行った．加えて，インストールのボトルネック分析し構築時間短縮のための指針を得た．結果，構築のためのコストは 16 ノード 151 秒と低コストであり，ジョブ実行時間（数時間〜数日）と比較しても許容範囲内であることを確認した．また，類似システム[6)] と比較すると，とくに仮想マシンのシステムイメージが大きい場合にはより高速であることを確認した．

2. グリッドジョブ実行環境の要件

グリッドでは複数の分散した計算リソースを管理ドメインをまたがってネットワークを通じて動的に共有する．そのため，共有されるソフトウェアリソースの不均質性の問題が起こる．これは，共有される計算リソースは複数の管理ドメインから提供されるため，各計算機の OS やアプリケーションなど様々な点で設定が大幅に異なるためである．しかし，通常グリッドユーザは権限が大幅に限定されているため，OS の入れ替えや新たなアプリケーションのシステム全体へのインストールなど，計算機上のソフトウェア環境の大幅な変更を許されていない．このため，ジョブ実行に必要な環境を自らセットアップすることができない．

このような限定された環境上で任意のジョブを実行可能にするための要件として，以下の 3 つをあげる．

ジョブ実行環境の仮想化 仮想化によってジョブ実行に使用される計算リソースと物理リソースを分離することにより，ジョブ実行環境の構築によるローカルユーザや他のジョブへの影響を排除する必要がある．

ジョブ実行環境の自動的な構築 ユーザからの実行環境の構築要求を受け取り，環境を動的に自動構築する仕組みが必要である．これによって，ユーザにとっては必要な実行環境をつねに利用できることになり，また管理者にとってはユーザの要求に応じてジョブ専用の実行環境を新たに構築する手間がなくなる．

ジョブ実行環境仕様の容易な指定 OS のセットアップ手順やソフトウェア設定の詳細を意識しなくても，ジョブ実行に必要な環境を GUI などで指定することによって環境を容易にセットアップできる仕組みが必要である．

我々は以上の要件を満たすシステムとして ORE（**O**pen **R**esource **E**nvironment）グリッドツールを提案する．これは仮想マシン技術と自動設定・インストール技術を組み合わせることによって実現されている．以下に ORE グリッドの構成技術である仮想マシン技術と自動設定・インストール技術，およびリモートジョブ機構を紹介する．

2.1 仮想マシン

仮想マシン（VM）とは 1 台の物理マシンを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環境を実現できる．このため，ソフトウェアの開発環境やテスト環境として仮想マシンを使用することにより，物理マシン上の環境を破壊することなくクリーンで制御された環境を構築できるといった利点がある．

PC 向けの仮想マシン技術として VMware[10)~12)]，Xen[13),14)]，coLinux[15),16)]，UML（User Mode Linux）[17),18)] などがある．一般に，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のトレードオフが存在する．ジョブを効率的に実行するにはジョブの性質に応じた最適な仮想マシンを選択することが必要である．

2.2 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設定をあらかじめ環境構成として入力しておくことよって，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．仮想マシンの構築機構としてこうしたツールを用いることによって，ジョブ実行に必要な任意の環境を複数の仮想マシン上に自動的に構築することができる．

代表的な自動設定・インストールツールとして Lucie[4)]，RedHat KickStart[19)]，NPACI Rocks[5)] がある．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（**表 1**）．様々なジョブに対応するために，複数の OS のインストールに対応している必要がある．また，インストール設定を簡単に作成・カスタマイズするための設定機構（GUI によるウィザードなど）を提供することが望ましい．

我々は大規模なクラスタ環境を容易に高速に構成することを目的とした高速セットアップ・管理ツール Lucie を開発している．クラスタのセットアップに Lucie を用いることにより，ノード 1 台ごとに手作業でインストールする場合と比べてセットアップを短時間に，かつ正確に行うことができる．

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ソフトウェア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できる．インストー

表 1 自動設定・インストールツールの比較
Table 1 Automatic configuration and installation tools.

	対応 OS	設定方法
RedHat Kickstart	RedHat	スクリプト
NPACI Rocks	RedHat	スクリプト
Lucie	RedHat, Suse, Debian	スクリプト，GUI

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，管理者はまず 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 サーバ上に Lucie インストーラの構築を行う．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 サーバ上に自動的に設定される．インストール対象マシンを（再）起動するとネットワークブートによりインストーラ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れる．

またインストール設定を簡易化するための仕組みとして，Lucie はメタパッケージ[20] を提供している．メタパッケージは MPI 環境や Condor 環境など機能単位でのインストール設定のテンプレートと，テンプレートカスタマイズ用の GUI（**図 1**）を提供する．Lucie ユーザはインストールしたい必要な機能を提供するメタパッケージを公開サーバからダウンロードし，メタパッケージに付属するウィザードを操作するだけで必要なインストール設定を生成することができる．

2.3 リモートジョブ機構

リモートジョブ機構では，グリッドユーザからのジョブ実行要求および実行環境要求を受け取り，ローカルリソース（クラスタ）上への VM の起動や自動設定・インストールの実行の指示，および実際のジョブ実行を行う．

グリッド上で標準的に用いられている Globus Toolkit[21] のリモートジョブ実行機構として GRAM がある．GRAM はユーザからジョブ実行依頼を受け取ると，バックエンドとしてジョブに応じたジョブマネージャを起動し，実際のジョブ実行を行う．ジョブマネージャとしては単純に fork システムコールによってジョブを実行するものや，Condor[22] などのジョブキューイングシステムにジョブを投入するものなどがある．GRAM の特徴としてジョブマネージャがモジュール化されており，新しいモジュールの追加による拡張が可能である．

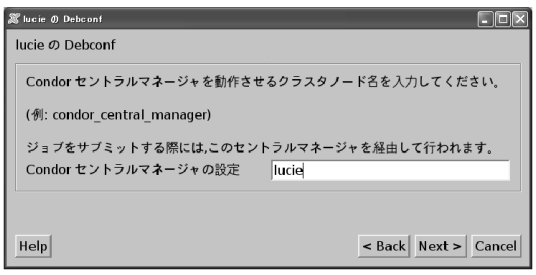


図 1 Condor メタパッケージのカスタマイズ GUI
Fig. 1 A customization GUI with Condor metapackage.

3. 設計と実装

グリッド上に動的かつ自動的に仮想ジョブ実行環境を構築するシステムとして，我々は ORE グリッドを開発している．ORE グリッドシステムは以下のコンポーネントから構成される（**図 2**）．

ジョブ実行サービス ユーザからのジョブ起動要求と実行環境構築要求を受け取る．実行環境構築要求は VM 構築サービスに転送し VM 構築を依頼する．構築された VM 上にジョブを起動し，結果をユーザへ返す．

VM 構築サービス ジョブ実行サービスから送られた実行環境構築要求に従って VM を起動し，指定された仕様の VM を自動的にセットアップする．

クラスタノード VM が起動される実計算機．

3.1 ORE グリッドによるジョブの投入

ORE グリッドを用いてジョブを投入する場合，ユーザは実行環境の構成やジョブの内容などジョブ実行に関する情報をすべて GUI を用いて指定可能である．具体的には，仮想マシンのハードウェア構成（台数，メモリ容量，HDD のパーティション構成など）およびセットアップする環境に対応したメタパッケージ（MPI 環境，Condor 環境，Java 環境など）を指定するための ORE グリッドウィザード☆（**図 3**）をダウンロードし，

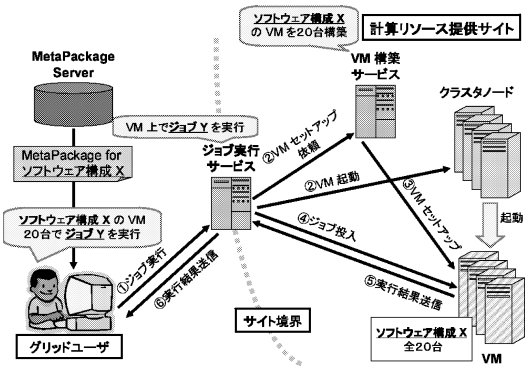


図 2 ORE グリッドアーキテクチャ
Fig. 2 ORE Grid architecture.

☆ 実際には ORE グリッドウィザード自体も 1 つのメタパッケージとして提供されている．

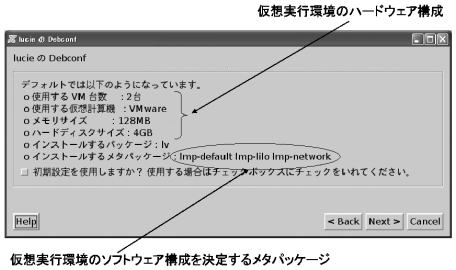


図 3 ORE グリッドウィザードによる VM 環境の設定
Fig. 3 VM configuration with ORE Grid wizard.

GUI を通じてカスタマイズする．続いて指定されたメタパッケージ（たとえば，図 1 の Condor メタパッケージなど）がダウンロードされ，ユーザはメタパッケージごとに GUI によるカスタマイズを行う．このように本システムでは設定がすべて GUI 化されており，類似システム[6)~8)] に見られる欠点である，ユーザによるカスタマイズ機構が提供されていない点や，独自の設定方式やシステムの詳細に関する知識を要求する点を改善している．

GUI による設定後，ジョブは指定された実行環境で自動実行され実行結果がユーザへ返却される（図 2）．

3.2 ORE グリッドの設計と動作

ORE グリッドの実際の動作の流れは次のようになる．

(1) ユーザは仮想マシンのハードウェア構成，ソフトウェア構成，およびジョブの実行内容をダウンロードした GUI を通じて入力する．GUI で生成された項目はジョブ実行サービスへ送信される．
(2) ジョブ起動サービスはユーザから送られてきたジョブ実行環境設定を VM 構築サービスに送信し，VM の構築を依頼する．また，要求された台数の VM をクラスタノード上に起動する．
(3) VM 構築サービスはこの VM 上に要求されたジョブ実行環境を自動構築する．
(4) ジョブ起動サービスはユーザから送られてきたジョブを自動構築された VM 上に投入する．
(5) VM は送られてきたジョブを実行し，得られた結果をジョブ起動サービスに返送する．
(6) ジョブ起動サービスはこの実行結果をユーザに返送する．

(1) でのユーザによるジョブ投入後，(2)〜(6) の動作はすべて自動的に実行される．このため，ジョブ実行に用いる VM の台数が増加した場合でも，ジョブの投入から実行結果確認までを完全に自動的に行うことができる．類似システム[6),8)] では実行環境の構築およびジョブの投入が自動化されておらずすべて手動で行う必要がある．結果として全体のジョブ実行時間が長くなる．一方，本システムでは自動化により全体の実行時間への影響を最小限にしている．

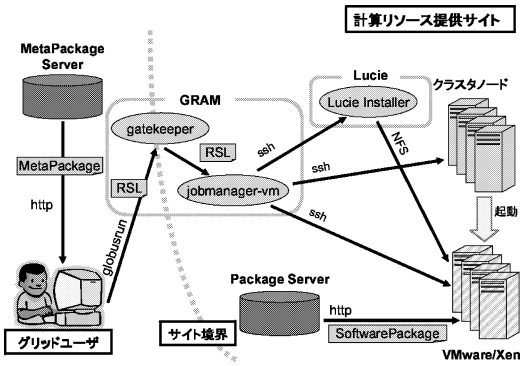


図 4 ORE グリッドの実装
Fig. 4 ORE Grid implementation.

3.3 ORE グリッドの実装

ORE グリッドの各コンポーネントの実装を詳しく説明する（**図 4**）．

メタパッケージサーバ メタパッケージサーバは VM の構築に必要な様々なメタパッケージ（MPI 環境，Condor 環境，Globus 環境など）を http で提供するインターネット上の公開サーバである☆．ユーザはジョブ実行に必要な機能を提供するメタパッケージをメタパッケージサーバから検索・ダウンロードし，実行するジョブや実行環境を指定する RSL（Resource Specification Language）記述を GUI ウィザードを通じて自動生成する．生成された RSL は globusrun コマンドによってジョブ実行サービスへ送信される．

ジョブ実行サービス ジョブ実行サービスとして，独自に拡張した GRAM を使用した．この拡張では，ユーザからの実行環境構築要求に応えるモジュールである jobmanager-vm を GRAM へ追加した．GRAM gatekeeper はユーザからの RSL を受け取ると，これを仮想マシンの起動や Lucie の起動を行うジョブマネージャモジュールである jobmanager-vm へ転送する．jobmanager-vm は ssh 経由で Lucie へのインストーラ構築依頼，クラスタノードでの VM の起動，VM インストール後の VM へのジョブ投入をそれぞれ行う．

VM 構築サービス VM 構築サービスとして，Lucie サーバによって作成される Lucie インストーラを

☆ たとえば http://lucie-dev.is.titech.ac.jp/にメタパッケージサーバが公開されており，様々なメタパッケージが入手できる．

用いている．これはインストーラ機能を持つ最小構成の Linux イメージであり，NFS 経由ですべての VM ノードに起動イメージとしてネットワークマウントされる．インストーライメージをマウントした VM はインストーラを実行し，ローカルディスクへのインストールを実行する．

VM ジョブ実行環境に用いる VM として，現在は VMware および Xen をサポートしている．ユーザは ORE グリッドメタパッケージに付属する GUI（図 3）を用いることによって，実際に使用する VM の種類を選択することができる．類似システム[6)~8)] に見られるような，単一の VM のみしかサポートしていないといった制限はなく，ジョブの性質に応じて適切な VM を選択することができる．

パッケージサーバ パッケージサーバは VM にインストールされるソフトウェアパッケージを http 経由で提供するサーバである．VM 上で起動する Lucie インストーラはインストール処理の一部としてパッケージサーバへ接続し，指定されたパッケージをローカルディスクへダウンロード・インストールする．

上記コンポーネントのうち，ジョブ実行サービス，VM 構築サービス，パッケージサーバはすべてクラスタノードと同じ管理ドメイン内（同一ネットワーク内）にあることを前提としている．これは，外部サーバによるインストールでは性能が大幅に低下することが容易に予想でき，実用的でないためである．また，現在の Lucie は ORE グリッドの動作に必要なコンポーネントを自動的にセットアップする仕組みを提供しているため，少ない労力で ORE グリッドを構築し外部ユーザへ提供することができる．

3.4 ORE グリッドの構築

ORE グリッドを構築し，仮想計算リソースを提供するための手順を説明する．管理者が構築しなければならないサーバは GRAM サーバ，Lucie サーバ，パッケージサーバの 3 種類である．Lucie サーバが自動インストールのために提供するサービスはインストーライメージをクラスタノードへ提供するための NFS と，インストール中のネットワーク情報を配布する DHCP である．これらの設定は Lucie の管理コマンドによって自動生成することができるため，自動生成された設定を既存の DHCP や NFS の設定に追加するか，もしくはこれを用いて新たなサービスを開始することで容易に構築することができる．また，Lucie は GRAM サーバ構築ツール，パッケージ全体のミラーイメージ，およびパッケージサーバ構築ツールを提供しており，Lucie サーバマシン上に自動的に GRAM サーバとパッケージサーバを構築することができる．もしパッケージ全体をミラーリングするためのハードディスク容量が不足している場合には，HTTP プロキシサーバを構築しクラスタノードへ提供することで解決できる．

また，計算リソースとして提供するすべてのクラスタノードには Xen もしくは VMware いずれか仮想マシンソフトウェアをあらかじめインストールしておく必要がある．管理者は自動インストーラを用いてクラスタのセットアップ時にこれらを追加インストールすることが期待される．Lucie では Xen 環境（Xen および Xen 対応カーネル）をセットアップするためのメタパッケージを提供している．このため，Lucie を用いてクラスタをセットアップすることによって ORE グリッドのためのクラスタ環境を自動的にセットアップすることができる．

本章で説明したツール類およびドキュメント類はすべて Lucie ホームページ☆からダウンロード可能である．

4. 評　　価

本システムの有効性を評価するために，BLAST（Basic Local Alignment Search Tool）[9)] 実行環境の構築を行い，実際に実行環境が構築され，実行結果がユーザへ返されることを確認した．また，構築時間を計測することで構築時間が BLAST の実行時間（数時間程度）に対して許容範囲内であるかどうかを確認した．なお，BLAST は相同性検索ソフトウェアの一種であり，問合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．加えて，インストール性能のボトルネックを分析し構築時間短縮のための指針を得た．

4.1 評 価 環 境

本システムの評価環境として東京工業大学松岡研究室の PrestoIII クラスタノードの一部を使用した．各種サーバおよびクラスタノードとして用いるマシンはすべて同一のものを使用し，スペックは**表 2** のとおりである．また構築する仮想マシンのスペックは**表 3** のとおりである．GRAM サーバおよび Lucie サーバは 1 台のノード上で動作させた．また，すべてのクラスタノードおよび各種サーバは Gigabit Ethernet でスイッチングハブ（DELL Power Connect 5224）に接続されており，同一管理ドメイン内にある．

☆ http://lucie-dev.titech.hpcc.jp:2500/

表 2 評価環境
Table 2 Experimentation environment.

CPU	AMD Opteron250 × 2
メモリ	PC2700 2 GB
OS	Debian GNU/Linux sarge
カーネル	Linux 2.4.27
ネットワーク	Gigabit Ethernet
VM	VMware 5.0（linux 版）

表 3 仮想マシンのスペック
Table 3 VM specification.

HDD	20 GB（SCSI buslogic）
メモリ	256 MB
OS	Debian GNU/Linux woody
カーネル	Linux 2.4.28
追加パッケージ	blast2

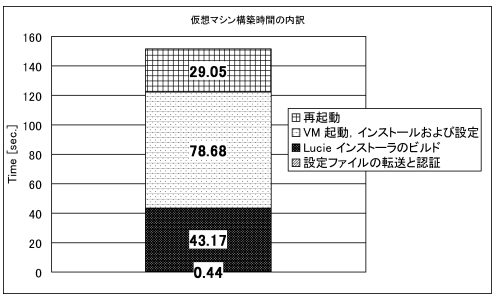


図 5 仮想マシン構築時間の内訳
Fig. 5 VM build performance.

4.2 ジョブ実行環境構築時間の評価

表 3 の構成で仮想マシンを 1 台構築し，構築の各ステップに要した時間を調べた．環境構築時間は**図 5** のようになり，仮想マシンの構築に全体で 151 秒要した．通常 BLAST のジョブ実行時間全体は数時間以上に及ぶため，これと比較すると構築時間は十分許容範囲であると考えられる．また，グリッド上のジョブの大部分も実行時間が数時間〜数日に及ぶため，本システムは十分適用可能だといえる．

さらに構築時間を短縮するために，図 5 でのインストールの各ステップの高速化を考える．インストール時間全体の 29%を占める Lucie インストーラのビルドステップでは，処理の大部分が Linux ベースシステムの展開に割かれており大幅な高速化は難しい．そこで再起動時間（全体の 13%）および VM のインストール・設定時間（全体の 52%）の削減方法を考える．

インストール後の再起動は，Lucie インストーラとして動作している Linux を終了し，ローカルディスクへインストールされた Linux を起動するために行われる．再起動の高速化機構の一種である Linux 2.6

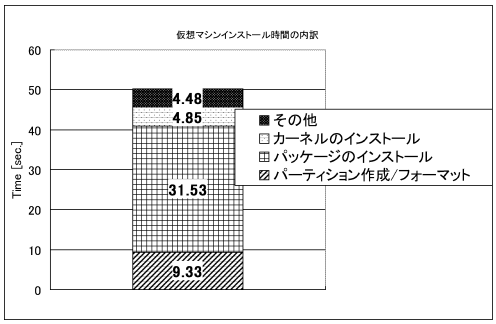


図 6 仮想マシンインストール時間の内訳
Fig. 6 VM installation performance.

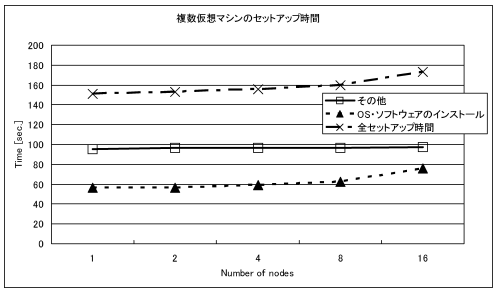


図 7 複数仮想マシンのセットアップ時間
Fig. 7 Multiple VM setup performance.

の kexec 機構[23] では，ブートローダを介さずに起動中の Linux カーネルとは別の Linux カーネルの起動を可能にする．そこで，kexec を用いて Lucie インストーラの Linux からインストールされた Linux へ高速に切り替えることによって再起動に要する時間を大幅に削減できると考えられる．

また，VM のインストール・設定時間を削減するために，このステップの内訳を調べると**図 6** のようになった☆．パッケージの取得やインストール，kernel のインストールといったインストール処理のみで全体の 63%と VM 構築時間の大部分を占めることが分かる．これを削減する方法として，一度構築した VM のイメージを保存しておき再利用する方法がある．また，これによる副次的な効果としてパーティションの作成やフォーマットに要する時間も短縮することが期待できる．そこで，構築済みのイメージの管理や再利用のための仕組みが今後の課題の 1 つである．

4.3 複数の仮想マシンの構築時間

図 7 に 1 台から 16 台までの仮想マシンを構築した場合の全セットアップ時間，および全セットアップ時間中の OS およびソフトウェアのインストール時間と，

☆ VM 自体のブート時間 21.64 秒は除いてある．

その他の処理（ハードディスクのパーティション作成やフォーマット，ソフトウェア設定など）に要した時間を示す．仮想マシンが増加するにつれ，その他の処理に要した時間は増加しないものの，OS およびソフトウェアのインストール時間が増加している．結果として全体のセットアップ時間が増加している．そのほかの処理に要する時間がほぼフラットである原因として，そのほかで実行される処理は各仮想マシンで独立して並列に実行可能な処理であるため，仮想マシン台数が増加しても全仮想マシンが一定時間内に完了できるためである．また，OS およびソフトウェアのインストールの処理時間が増加する原因として，処理中のソフトウェアパッケージ取得の際にパッケージサーバに全仮想マシンからの接続による負荷が集中し，これがボトルネックとなっていることが考えられる．

このことは計算機の台数が少ないときはあまり問題にならないが，仮想マシンの構築が数百台といった大規模になったときには大きな性能低下が予想される．このために，先に述べた構築済み VM イメージのキャッシュやパッケージサーバのレプリケーション，および Dolly+ [24),25)] などの高速ファイル転送システムを使用することが必要である．

5. 関 連 研 究

仮想計算機を用いグリッド上にジョブ実行環境を動的に構築する技術が数々提案されている．

Virtual Cluster Workspace [6)] は ORE グリッドと同じく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．あらかじめ作成しておいた仮想計算機のディスクイメージをノード全体へ GridFTP [26)] を用いて配布することで仮想実行環境を構築する．このシステムでは ORE グリッドのような外部ユーザが仮想環境を自由に設定するための機構を備えておらず，Xen のディスクイメージを事前に管理者が作成しなければならない．このため，管理者のコストが高くジョブによっては起動できないものが存在する．一方，我々のシステムでは仮想実行環境をグリッド上のサービスとして提供しており，ジョブの実行要求に応じて適切な実行環境を動的かつ自動的にユーザへ提供することができる．また，Virtual Cluster Workspace では Xen のシステムイメージ全体（仮想ディスクおよびスワップパーティション，メモリイメージ）をコピーしているため無駄なデータのコピーが生じ，システムイメージサイズが増加するに従ってセットアップ時間も増加する．一方，我々のシステムではインストール時間はシステムイメージサイズによらず，インストールするソフトウェアのサイズに影響される．これは通常，システムイメージ全体をコピーする方式と比較して高速であり，大容量のディスクを用いた場合でもインストール性能に影響を受けないという利点がある．

同様のシステムとして VMPlants [7)] がある．仮想計算機の構築に用いる VM の仕様として，ユーザは 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．よく使われる最小公倍数的な仮想計算機のディスクイメージを golden VM image として保存し，これを DAG による構築手順中で再利用することにより，golden image と類似した構成の仮想計算機の起動や構築の高速化をはかっている．しかし，VMPlants では仮想計算機の仕様を DAG で手続き的に記述しなければならないため，OS やソフトウェアのセットアップ手順や設定方法の詳細を熟知している必要があり煩雑である．一方，我々のシステムではセットアップする環境をメタパッケージを選択し GUI を通じてカスタマイズすることにより，機能単位でのより抽象化された設定が可能である．

The Virtuoso Model [8)] は仮想計算機グリッドのための仮想ネットワークを提供する．計算リソース提供者の仮想計算機を仮想ネットワークを通じてユーザのローカルネットワークに接続することにより，ユーザはグリッド上の計算リソースをローカルネットワーク上の計算機と同様に使用することができる．仮想計算機のカスタマイズでは，ユーザは仮想計算機のメモリ容量や HDD のパーティション構成などの基本的な項目のみを指定することができる．仮想計算機には基本的なオペレーティングシステム構成とアプリケーションのみがプリインストールされるため，独自に使用したいアプリケーションやライブラリをユーザ自らがインストールする必要がある．そのため，ユーザへの負担は大きくとくに数十台〜数百台の大規模な仮想計算機群を構築する際には非常に大きいコストとなる．一方，ORE グリッドでは OS やソフトウェアはすべて自動的にインストールされるためこうした問題は起こらない．また，仮想計算機のインストールやカスタマイズが通常のシェルを通じて対話的に行われるため，仮想計算リソース提供者にとってはユーザがどのようなアプリケーションをインストールしたかを把握できないという問題がある．一方，我々のシステムでは仮想計算機のインストール処理を Lucie とメタパッケージ経由に限定しているため，サイトポリシにそぐわない環境の構築を必要に応じて制限することが可能である．

Virtual Cluster Workspace，Virtuoso および VM-

Plants に共通する欠点として，セットアップできる環境に制限があることがある．Virtual Cluster Workspace では使用可能な仮想計算機として Xen のみという制限がある．一方 ORE グリッドでは Xen と VMware をサポートしており，ジョブの性質に応じて適切な仮想計算機実装を選択可能である．また，Virtuoso および VMPlants ではインストールが可能な OS として RedHat Linux のみという制限がある．これはそれぞれで用いられている下位のインストーラが RedHat のみをサポートしていることに起因する．一方 ORE グリッドで用いている Lucie は RedHat，Suse，Debian の各バージョンをサポートしているため，ジョブに応じて幅広い構成の実行環境を構築することができる．

6. まとめと今後の課題

本研究では自動インストール・設定ツール Lucie と仮想マシン技術を用いて，グリッド環境上での動的なジョブ実行環境構築システムを実現した．特徴として，既存の計算機環境に影響を与えず独立した環境を仮想的に短時間で構築できることがある．また関連システムと比較した場合の優位点として，より広範囲な実行環境（OS，ソフトウェア，仮想マシン実装）を指定できる点や，実行環境の仕様を GUI を用いて容易に記述できる点，またジョブの投入から実行環境の構築からジョブの実行，結果の返却までがすべて自動的に行われる点，実行環境のシステムイメージサイズによらず比較的短時間で構築できる点があげられる．またプロトタイプを作成し実際に相同性検索ソフト BLAST の実行環境を構築し，BLAST ジョブ実行環境の自動構築と起動・終了を確認した．評価の結果，ジョブ実行環境の構築に要する時間は一般的なグリッドで実行されるジョブの実行時間と比較すると十分許容範囲であることを確認し，本システムのジョブ実行環境としての有効性を確認した．

今後の課題として，以下の 4 点があげられる．

ジョブ実行環境構築時間の短縮 一度構築した仮想マシンのディスクイメージをキャッシュし，構成の近い仮想マシン環境を構築する際にはこれを再利用することで，VM を動作させるマシンのネットワーク性能や CPU 使用率によっては新たにゲスト OS をインストールする場合に比べインストール時間を大幅に削減できると考えられる．こうしたインストール性能の高速化やトレードオフの議論については発表済み別論文[27] にゆずる．

より多くの仮想マシンのサポート 現在の実装ではユーザが選択できる仮想マシンとして VMware および Xen をサポートしている．今後は UML や coLinux などより多くの仮想マシンをサポートし，また実行されるジョブの性質（I/O インテンシブ，CPU インテンシブなど）に応じて各種仮想マシンのうち最適なものを選択可能にする必要がある．新たな仮想マシンに対応するためには，仮想マシンごとに異なるディスクイメージの取扱い方法（ディスクイメージとして物理パーティションを使用するものや，仮想ディスクイメージを使用するものなどがある）に Lucie インストーラを対応させる必要がある．

より多くのゲスト OS のサポート クラスタ用 OS として Linux のほかに Windows が広く利用されている．しかし，現在の Lucie では Windows の自動セットアップについて次の制限がある．現在の Linux カーネルでは Windows のファイルシステムとして FAT16，FAT32 のみをサポートしており，Windows クラスタで主に使用される NTFS への書き込みを正式にサポートしていない．Lucie は Linux 上で動作するため，NTFS を使用したクラスタを構築することができない．そこで，今後の Linux カーネルの NTFS 対応状況に合わせ Lucie の Windows への対応を強化する必要がある．

仮想マシンを立ち上げるマシンの自動選択 現在の実装では仮想マシンを立ち上げるマシンは固定となっており，これを動的に決める機構がない．本来は仮想マシンを立ち上げる候補のマシンの CPU 使用率やメモリ，HDD の使用量といったリソースやサイトポリシをもとに，空いているマシンを動的に判断するローカルスケジューラ機構が必要である．また，このためのサイトポリシの指定機構が必要である．

リソース制限 現段階ではユーザが指定する仮想マシンの構成に対して，リソース提供者はこれに制限をかけたりや拒否をすることはできない．しかし実際の環境ではユーザのリソース使用状況に対して，サイトポリシに見合った制限をかける必要がある．また構築される仮想マシン環境が複数の管理ドメインをまたがる場合には，GWiQ-P[28] のようにグリッド全体に対してユーザが使用できるリソースを制限する仕組みが必要である．

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 (July 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) Meyer, R.A. and Seawright, L.H.: A virtual machine time-sharing system, *IBM Systems Journal*, Vol.9, No.3, pp.199–218 (1970).
4) Takamiya, Y., Manabe, A. and Matsuoka, S.: Lucie: A Fast Installation and Administration Tool for Large-scaled Clusters, *Proc. Symposium on Advanced Computing Systems and Infrastructures* (*SACSIS2003*), pp.365–372 (May 2003).
5) Papadopoulos, P.M., Katz, M.J. and Bruno, G.: NPACI Rocks: Tools and techniques for easily deploying manageable linux clusters, *Proc. 2001 IEEE International Conference on Cluster Computing*, Newport, CA (Oct. 2001).
6) Virtual Cluster Workspaces for Grid Applications, Technical report, Argonne National Laboratory (Apr. 2005).
7) 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: Proc. 2004 ACM/IEEE conference on Supercomputing* (2004).
8) Sundararaji, A.I. and Dinda, P.A.: Toward virtual networks for virtual machine grid computing, *Virtual Machine Research and Technology Symposium*, pp.177–190 (2004).
9) 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).
10) Sugerman, J., Venkitachalam, G. and Lim, B.: Virtualizing I/O devices on vmware workstation's hosted virtual machine monitor, *Proc. General Track: 2002 USENIX Annual Technical Conference*, Berkeley, CA, USA, pp.1–14, USENIX Association (2001).
11) Waldspurger, C.A.: Memory resource management in VMware ESX server, *SIGOPS Oper. Syst. Rev.*, Vol.36(SI), pp.181–194 (2002).
12) VMware Web Site. http://www.vmware.com/
13) 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: Proc. 19th ACM symposium on Operating systems principles*, New York, NY, USA, pp.164–177, ACM Press (2003).
14) Xen Web Site. http://www.cl.cam.ac.uk/Research/SRG/netos/xen
15) Aloni, D.: Cooperative linux, *Proc. Linux Symposium*, Vol.1, pp.23–32 (2004).
16) coLinux Web Site. http://www.colinux.org/
17) Dike, J.: A user-mode port of the linux kernel, *Proc. USENIX Annual Linux Showcase and Conference* (Oct. 2000).
18) User-mode Linux Web Site. http://user-mode-linux.sourceforge.net/
19) Hamilton, M.: Kickstart document. http://www.ibiblio.org/pub/Linux/docs/HOWTO/other-formats/html_single/KickStart-HOWTO.html
20) Takamiya, Y. and Matsuoka, S.: Design and Implementation of Configuration Packaging Methods for Cluster Installers, *IPSJ SIG Notes 2004-HPC-99*, pp.55–60 (July 2004).
21) 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 (Summer 1997).
22) Condor Web Site. http://www.cs.wisc.edu/condor/
23) Reboot Linux faster using kexec. H. nellitheertha. http://www-128.ibm.com/developerworks/linux/library/l-kexec.html
24) Dolly+ web page. http://corvus.kek.jp/~manabe/pcf/dolly/index.htm
25) Manabe, A.: Disk cloning program 'dolly+' for system management of pc linux cluster, *Computing in High Energy Physics and Nuclear Physics* (2001).
26) Allcock, W., Bester, J., Bresnahan, J., Meder, S. and Tuecke, S.: GridFTP: Protocol Extensions to FTP for the Grid, Technical report, Argonne National Laboratory (Apr. 2003).
27) 山形育平，高宮安仁，中田秀基，松岡　聡：グリッド上における仮想計算機を用いた高速なジョブ実行環境構築システム，情報処理学会研究報告 2006-HPC-105（HOKKE2006），pp.127–132 (Mar. 2006).
28) Karmon, K., Liss, L. and Schuster, A.: Gwiq-p: An efficient decentralized grid-wide quota enforcement protocol, *The 14th IEEE Intl Symposium on High Performance Distributed Computing* (*HPDC14*), Research Triangle Park, NC, pp.222–232 (2005).

(平成 18 年 1 月 27 日受付)
(平成 18 年 5 月 20 日採録)

高宮 安仁（正会員）

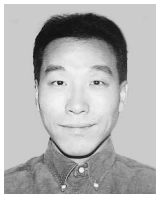
1977 年生．2003 年東京工業大学大学院情報理工学研究科数理・計算科学修士課程修了．2006 年同大学院博士課程修了．同年日本電気（株）入社．クラスタリングシステム上での耐故障ミドルウェアに興味を持つ．

山形 育平（学生会員）

2006 年東京工業大学大学院情報理工学研究科数理・計算科学専攻修了．仮想計算機技術の応用に興味を持つ．

青木 孝文（学生会員）

2005 年東京工業大学理学部情報科学科卒業．現在，同大学大学院総合理工学研究科知能システム科学専攻博士前期課程在学中．バーチャルリアリティ，ハプティックインタフェース，複合現実感に関する研究に従事．



中田 秀基（正会員）

昭和 42 年生．平成 2 年東京大学工学部精密機械工学科卒業．平成 7 年同大学大学院工学系研究科情報工学専攻博士課程修了．博士（工学）．同年電子技術総合研究所研究官．平成 13 年独立行政法人産業技術総合研究所に改組．現在同所グリッド研究センター主任研究官．平成 13 年より平成 17 年まで東京工業大学客員助教授を兼務．グローバルコンピューティング，並列実行環境に関する研究に従事．

松岡 聡（正会員）

1963 年生．1986 年東京大学理学部情報科学科卒業．1989 年同大学大学院博士課程から，学情報科学科助手に採用，同大学情報工学専攻講師を経て，1996 年に東京工業大学情報理工学研究科数理・計算科学専攻助教授．2001 年 4 月に東京工業大学学術国際情報センター教授，2002 年より国立情報学研究所の客員教授を併任．博士（理学）．