Vol.2010-HPC-126 No.39
2010/8/5

“Bare Metal” Cloud: 実マシンを提供するクラウドサービス

高宮安仁[†1]　田浦健次朗[†2]
安井雄一郎[†3]　藤澤克樹[†3]

オンデマンドで仮想マシン (VM) を提供する Infrastructure as a Service (IaaS) は VM のオーバーヘッドや他のユーザが実行するジョブの影響により、大部分の HPC アプリケーションで本来の性能を出すことができない。VM のかわりに実マシンを提供する IaaS を実現すればこうした問題は解決できるが、VM イメージを元に何台でも複製できるという VM の高い運用性を実マシン上で実現するのは難しい。我々のシステムは 1) IaaS のデファクトスタンダードである Amazon EC2 の VM イメージを実マシンにインストールする仕組みと、2) インストールエラーの検出およびフェイルオーバー機能を備えることで、VM イメージを元に実マシンを安定してセットアップできる IaaS を実現した。これによって、HPC アプリケーションの性能を損ねない多様な実行環境を実マシン上に簡単にセットアップできる。

“Bare Metal” Cloud: A cloud service delivering real machines

Yasuhito Takamiya,[†1] Kenjiro Taura,[†2]
Yuichiro Yasui[†3] and Katsuki Fujisawa[†3]

Infrastructure as a Service (IaaS) that enables on-demand deployment of virtual machines (VM) cannot bring out the real performance of most of HPC applications because of the overhead of underlying VMs and the effect of other co-located users’ activities. While real-machine based IaaS instead of VMs may settle these problems, it is still difficult to achieve high managability of VMs on real machines. Our novel mechanism achieved an IaaS that can deploy dedicated real machines on demand by providing follwoing mechanisms that: 1) install the VM image of Amazon EC2, the de-facto standard of IaaS, into real machines 2) failover boot errors by investigating boot-sequence syslogs. With these mechanisms, one can build a variety of execution environments that never lower the performance of HPC applications on real machines.

1. はじめに

クラウドコンピューティング[3),12)] の一種である Infrastructure as a Service (IaaS)[2),14),16)] は起動したいマシンイメージと台数を指定するだけで使い始められるという利便性や、使った分だけ払う “pay-as-you-go” 課金モデルにより Web サービス業者を中心に急激にユーザを拡大しつつある。ユーザからのリクエストに応じて安価かつ動的に仮想マシン (VM) を生成するという IaaS の仕組みは、その構成要素である VM の次の性質に依存している:

- VM 単位の多重化によって一台の実マシン上にたくさんのユーザを共存させることができる。このため、仮想マシンモニタ (VMM) を実行する実マシンの台数を圧縮できコストを下げられる。
- VM のセットアップはすべてソフトウェアレベルで行うため、物理的な電源 ON/OFF を伴う実マシンのセットアップに比べ安定してマシンを制御できる。このため、リソースの足りる限りユーザが要求したマシン台数を確実に提供できる。
- VM は実マシンのハードウェアを仮想化することでまったく同じハードウェア構成のマシン群を提供する。このため、ひとつの VM イメージファイルを元にまったく同じ VM を複製し起動できる。

しかし、IaaS の提供する VM 環境 上で High Performance Computing (HPC) アプリケーションを実行する場合、次の性能的な問題がある (第 2.2 節):

性能のばらつき　同じ実マシンを共有する他のユーザのリソース利用状況によって利用可能なリソース量が時々刻々と変化する。Web アプリケーションと異なり、多くの HPC アプリケーションは CPU キャッシュやメモリバスを占有する前提で最適化されているため、こうした変動の影響を受けやすい。このため、時間帯によって性能がばらつきやすく性能予測が難しい。

性能の低下　VM はハードウェア仮想化のオーバーヘッドが大きく、またページサイズやファイルシステム、およびパーティションサイズの変更といった HPC アプリケーションの性能に大きく影響のあるカスタマイズをサポートしていない。このため、実マシンに比べて大きく性能が落ちる。

このように HPC 実行環境として性能的に問題のある IaaS であるが、”VM イメージを

†1 NEC システムプラットフォーム研究所
†2 東京大学大学院 情報理工学研究科 電子情報学専攻
†3 中央大学 理工学部 経営システム工学科

選ぶだけで、好きな台数のマシンをいつでも起動できる” という利便性は Web サービスだけでなく HPC にも有用である。我々の目的はこうした IaaS の利便性を保ちつつ、実マシンと同じ性能を達成できるような “HPC 向け IaaS” を実現することである。

そこで我々は実マシンをユーザに提供する IaaS である Bare metal cloud を提案する。主な特徴は次のとおりである:

- 実マシンを 1 ユーザが占有できるため、他のユーザの影響による性能のばらつきが無い。このためユーザ間で同じ実マシンを共有する IaaS と比較すると基本課金は高くなるが、性能予測しやすいため課金額を実行前に見積もりやすい。
- 実マシンで実行できるため、VM による性能低下が無い。このため、実機向けに最適化された HPC アプリケーションの本来の性能を引き出せる。
- 代表的な IaaS である Amazon EC2[2] (以下 EC2) 用に公開された 6,000 近く[⋆1] の VM イメージから実マシンを起動できる (第 4.2 節, 第 4.4 節)。このため、あらかじめ MPI など主要なアプリケーションがセットアップされた、好きな Linux ディストリビューション環境をすぐに使いはじめることができる。
- また、一つの VM イメージから実マシンと EC2 にまったく同じ実行環境を作ることができる。これによって EC2 へのベンダーロックインが防げる。加えて、実マシンと VM のシームレスな混合クラウド環境を構築できる (第 2.2 節)。
- 指定された台数の実マシンを確実に起動する仕組みを実現した (第 4.3.1 節)。もし起動に失敗した場合でも、自動的に予備マシンを補完することで指定された台数を確保する。

我々のアイデアはシンプルである: IaaS の性能のばらつきや低下を改善するために VM に複雑な仕組みを導入する[4),17)] のではなく、IaaS が実マシンを提供してしまえばよい。以降では VM と実マシンのセットアップ手段の違い (第 2.1 節) に着目し、” 実マシンを提供する IaaS” を実現するための各種ビルディングブロックを説明する。

2. 背景: IaaS

IaaS の典型的な利用イメージをつかむために、代表的な IaaS である Amazon EC2 の使用例を見てみよう (図 1 の上段)。基本的には次の 2 ステップだけで利用できる。

ステップ 1 (VM イメージの検索) ユーザは起動したい VM イメージ (Amazon Machine Image、AMI と呼ばれる) をクラウドストレージサービス Amazon S3 (以下 S3) 上で検索する。たとえば次のコマンドはすべての AMI とその ID (`ami-*`) をリストアップする:

```
% ec2-describe-images --all
                                ...
IMAGE ami-f51aff9c ec2-public-images/fedora-8-i386-base-v1.06.manife ...
IMAGE ami-2b5fba42 ec2-public-images/fedora-8-i386-base-v1.07.manife ...
IMAGE ami-5647a33f ec2-public-images/fedora-8-i386-base-v1.08.manife ...
                                ...
```

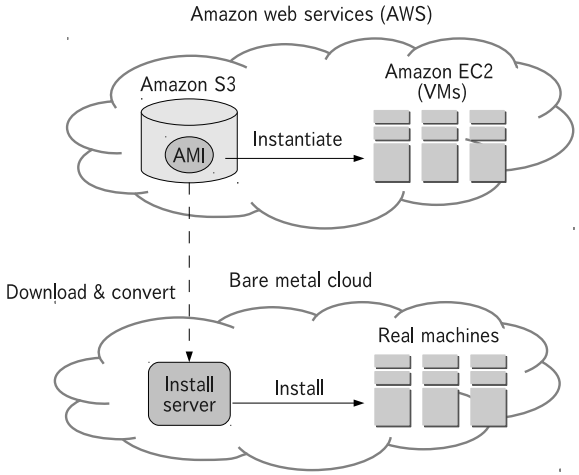

図 1 Amazon EC2 と Bare metal cloud の概要

ステップ 2 (VM の起動) ユーザは起動したい AMI の ID とマシン台数を指定することで VM を起動する。例えば、ID が `ami-48aa4921` の AMI から 16 台の VM を 起動する場合のコマンドは次のようになる:

```
% ec2-run-instances ami-48aa4921 --instance-count 16 --key gsg-keypair
% ec2-describe-instances   # 起動した VM の一覧を表示
INSTANCE i-a09969c7 ami-48aa4921 ec2-72-44-50-86.foobar.com ...
INSTANCE i-a09969c8 ami-48aa4921 ec2-72-44-50-87.foobar.com ...
INSTANCE i-a09969c9 ami-48aa4921 ec2-72-44-50-88.foobar.com ...
                                ...
```

このように IaaS は “オンデマンドで利用可能な無限のリソース” という抽象化をユーザに与える[3)]。こうした抽象化が可能な理由は、VM がマシンのセットアップを次のように抽象化するためである。

⋆1 2010 年 6 月現在

2.1 セットアップの抽象化

セットアップ方式に注目すると、VM と実マシンの大きな違いは次の 3 点である:

マシンイメージ VM の実体は IaaS 固有な仮想ハードディスクイメージであり、OS やアプリケーションなどがすべて 1 つの VM イメージファイルとして収められている。このため、VM イメージをコピーするだけでまったく同じ VM を複製できる。一方で実マシンの実体は物理ハードディスクに書き込まれたファイルの集合であり、インストール方法も Linux ディストリビューションごとに大きく異なる。

起動エラー VM の起動や停止はすべてソフトウェア的に行われるため、メモリ不足や過負荷などの特殊な状態で無い限り起動プロセスは成功する。一方で実マシンは電源ユニットの故障や他の物理要因により非決定的であり、しばしば失敗する。しかも、起動中はエラーメッセージがコンソールにのみ出力されるため、外部からの診断がむずかしい。

ハードウェア構成 VM は CPU やハードディスクを仮想化し均一化された仮想マシンを OS に見せる。このためホストマシンのハードウェア構成が異なっても同じ VM イメージを起動できる。一方で実マシンはマシンごとにハードウェア構成が異なる可能性が高い。

このようにセットアップの抽象化は IaaS の実現に不可欠である。しかし VM による抽象化は次に示すように性能面での副作用が大きい。

2.2 性能への影響

ここで実際の HPC アプリケーション 2 種類を例に挙げ、VM が原因による性能のばらつきや性能低下を見てみよう。我々は、実マシンおよび EC2 から成る混合クラウド環境上で、マシン資源の動的な増減に対応した最適化系 HPC アプリケーションを開発している。図 2 は、EC2 および同等な CPU を搭載した実マシン上で全米道路データに対し 1 対全最短路ソルバ[18] を 24 時間、10 分間隔で実行した場合の実行時間である。グラフによると EC2 上での性能のばらつきが大きく、最悪の実行時間は最良の値と比較して 1.69 倍になる。また、ページサイズを増やすためのカーネルオプション (HugeTLB[10]) を有効にした実マシンよりも性能は低い。図 3 は半正定値最適化問題に対するソルバ SDPA[7] を EC2 および同等な CPU を搭載した実マシン上で 4 種類の BLAS[6] 実装を使って実行した結果である。EC2 と実機を比較すると、他のユーザによる外乱や VM のオーバーヘッドによって実行時間が最大で 3.44 倍になる。またその傾向は ACML[1], MKL[9], GotoBLAS2[8] など実マシン上での実行を前提として高度な最適化が施されたライブラリほど顕著である。

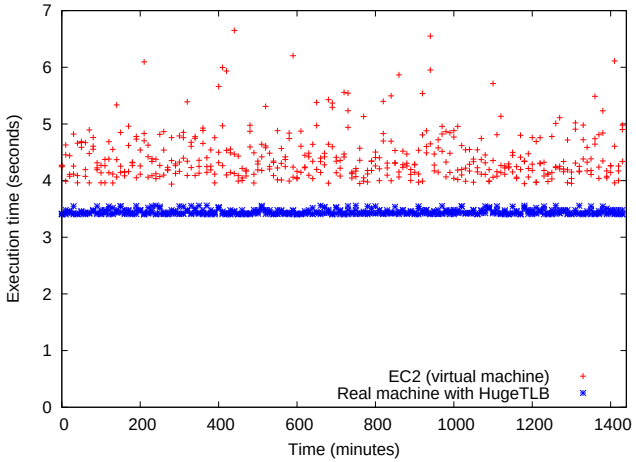

図 2 EC2 の m1.large タイプマシン (7.5GB メモリ、仮想デュアルコア CPU x2、1 コアは AMD Opteron または Intel Xeon の 1.0-1.2GHz に相当)、および同等の CPU (Intel Xeon E5345) を搭載した実マシン上で全米道路データの 1 対全最短路問題の所要時間の比較。10 分間隔で 24 時間計測。

3. Bare metal cloud

“実マシンを提供する IaaS” を実現する Bare metal cloud の全体像を図 1 の下段に示す。Bare metal cloud は実マシンのセットアップを次のように抽象化することで IaaS を実現している:

マシンイメージの変換 実マシンを VM と同様にイメージベースでインストールするために、AMI を EC2 インスタンスを通じて実マシン用に変換する (第 4.2 節)。また、VM イメージは一般に巨大なので (たとえば AMI は標準で 10GB)、一度変換したマシンイメージを圧縮してキャッシュし、さらに実マシン間で効率的にコピーする (第 4.4 節)。

起動エラー処理 エラーが起こった場合にも指定されたマシン台数を確実に確保するために、実マシンの起動失敗をサーバ側のネットワークブートログから外部診断する。またあらかじめ予備マシンを余分にセットアップしておくことで、エラーが起こった場合には指定された台数を確保する (第 4.3.1 節)。

ハードウェア構成の違いの吸収 マシンのハードウェア構成に応じてカーネルを自動的に選択し、またドライバを含む initrd を自動的に再構築することでハードウェアの違いを

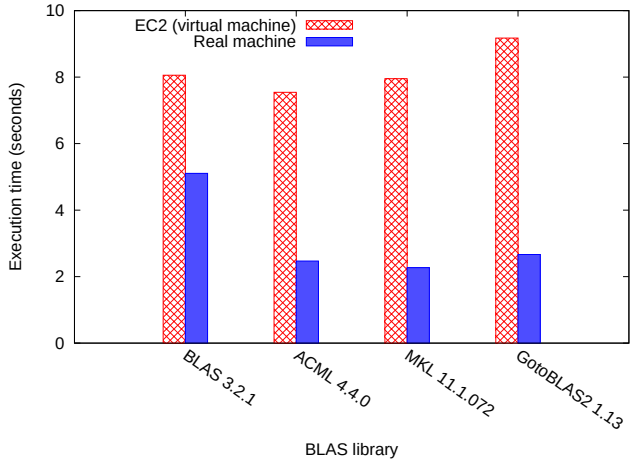

図 3 EC2 の m1.xlarge タイプマシン (15GB メモリ、仮想クアッドコア CPU x2、1 コアは AMD Opteron または Intel Xeon の 1.0-1.2GHz に相当)、および同等の CPU (Intel Xeon E5345) を搭載した実マシン上で半正定値最適化問題を 4 種類の BLAS 実装で実行したときの所要時間の比較。

吸収する (第 4.4 節)。

こうしたマシンイメージの変換やインストール、エラー処理といった処理はすべてインストールサーバが行う。

4. インストールサーバ

インストールサーバはクライアントからのリクエストを受け取り、指定された AMI を指定された台数の実マシンにネットワークインストールする。すべての処理は自動的に行われるため、実行中にキーボードやマウス入力を求められることはない。実マシンにはインストール用メディア (CD-ROM や DVD, USB メモリなど) を挿す必要はなく、インストーラはネットワーク経由で実マシンに送り込まれ実行される。インストール開始やマシン停止のための電源 ON/OFF はインストールサーバから Intelligent Platform Management Interface (IPMI) 経由で集中的に管理される。

インストール用メディアの無い実マシン上でインストールを開始するために、インストールサーバはインストーラ実行環境とインストーラ本体をネットワーク経由で実マシンに提供する。実マシンはインストーラ実行環境を起動し、この上でインストーラ本体を実行する。実際の処理は次の順で動作する:

(1) インストールサービスの開始: 実マシンへインストーラ実行環境を提供するためのインストールサービスをインストールサーバ上で起動する。
(2) AMI のダウンロードと変換: 指定された AMI をインストールサーバにダウンロードし、実マシンへインストール可能な形式に変換する。
(3) リブート (一回目): 指定された台数 + 予備の実マシンを IPMI でリブートする。リブートした実マシンはインストールサーバが提供する実行環境をマウントし起動する。
(4) インストール: インストールサーバは実マシンがマウントしたインストーラ実行環境上でインストーラ本体を起動し、実マシン用に変換済みの AMI をコピーする。インストーラはコピーされた AMI を実マシンのディスク上に展開する。
(5) ネットワークブートサービスの停止: ステップ 4 で AMI を元にインストールした Linux をブートするために、ステップ 1 で開始したインストールサービスのうちネットワークブートサービスを無効にする。
(6) リブート (二回目): 実マシンは 2 回目のリブートを行う。リブート後、ステップ 4 でインストールした Linux をローカルディスクから起動する。

続く節では、それぞれのステップを詳細に説明する。

4.1 インストールサービスの開始

物理マシンのネットワークブートとインストールに必要な次のサービスをインストールサーバ上で起動する。

ネットワークブートサービス インストーラ実行環境をネットワーク経由で実マシンに提供する。ネットワークブートサービスは Pre Execution Environment[⋆1](PXE) によるディスクレスブートを提供する DHCP, TFTP, および NFS デーモンから成る。

AMI キャッシュサービス S3 からダウンロードし実マシン用に変換・圧縮した AMI をキャッシュする。

なおネットワークブートサービスのために、あらかじめ実マシンの BIOS のブート設定画面で PXE ブートを最優先に設定しておく必要がある。この設定は初回の一度だけでよい。

4.2 AMI のダウンロードと変換

AMI は作成者の X.509 秘密鍵によって暗号化され S3 上に保存された EC2 の仮想ディスクイメージであり、公開されている AMI であればこれを元に VM を起動することができる。しかし、作成者以外が AMI ファイルをローカルにダウンロードしたり展開することはできない。そこで、AMI を実マシンにインストール可能な Linux のファイル集合に変換

⋆1 一種のブートローダであり、ネットワークブートサービスから取得したブートイメージを起動する機能を持つ

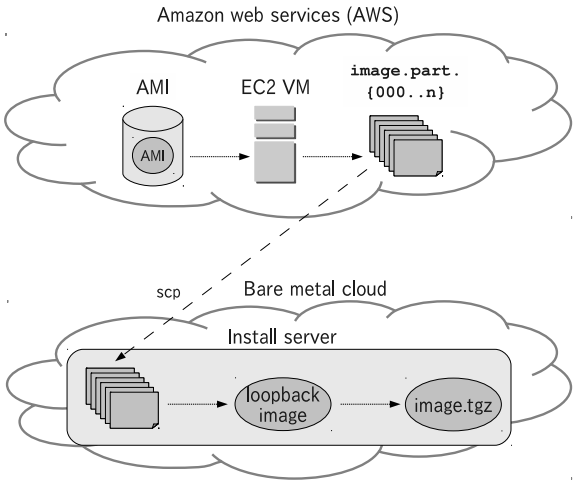

図 4 AMI を実マシンにインストールするためのダウンロードと変換

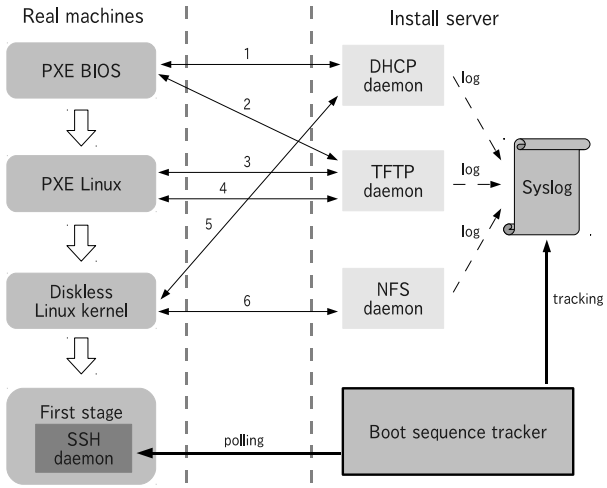

図 5 典型的なネットワークブートシーケンス。ブートシーケンストラッカーは syslog の追跡と sshd のポートへのポーリングによってネットワークブートの失敗を検知する。

するために、いったん AMI から VM を起動し、この VM から Linux のファイル集合を以下のように取り出す (図 4):

(1) AMI から EC2 インスタンスを一台起動する。
(2) 起動した EC2 インスタンスを自分の X.509 秘密鍵で AMI チャンクファイルに変換する (`ec2-bundle-instance` コマンド)。なお生成されるチャンクファイルは AMI を 10MB 単位に分割したものである。
(3) インストールサーバに scp で AMI チャンクファイルをコピーする。
(4) AMI チャンクファイルを結合したのち復号し (`ec2-unbundle` コマンド)、一時ディレクトリにループバックマウントする。
(5) ループバックマウントされたディレクトリの中身を tar + gzip 形式で 1 ファイルに圧縮し、AMI キャッシュサービスにキャッシュする。

4.3 リブート (一回目)

AMI のインストールを開始するために、インストールサーバは IPMI インタフェース経由で実マシンの電源を入れ、実マシンをネットワークブートする。ネットワークブート後、実マシンは AMI をローカルディスクに書き込むためのインストーラ実行環境を起動する。

なお、ネットワークブートの失敗を検出するため、実マシンとネットワークブートサービス間の通信ログはインストールサーバ上の syslog に保存される (図 5)。また、いくつかの実マシンがネットワークブートに失敗した場合でも台数のロスを少なくするために、余分な実マシンをあわせて起動しておく*1

実マシンがブートすると、PXE ブートシーケンスに入る。PXE ブートシーケンスでは、インストールサーバ上の DHCP サーバにリクエストを出し自分の MAC アドレスに対応する IP アドレスとネットワーク情報、および TFTP サーバ (インストールサーバ) の IP アドレスを得る。次に TFTP サーバからファーストステージ実行用の Linux カーネルイメージを受信しこれをブートする。Linux カーネルはふたたび DHCP リクエストを出し、実行環境としてマウントする NFS ディレクトリの URI を得る。そして、この URI をルートファイルシステムとしてマウント後、その中の `/etc/init.d/rcS` を起動する。この `/etc/init.d/rcS` は Linux システムが起動したときに最初に実行されるスクリプトであり、インストーラ実行環境の初期化スクリプトである。`/etc/init.d/rcS` は起動後、インストールジョブを受け入れるための SSH デーモンを起動する。

*1 余分に起動する台数はシステムごとのエラー頻度やユーザと結ぶ Service Level Agreement (SLA) の内容によって異なるため、管理者が適切に設定しておく必要がある。

4.3.1 ブートエラーの検出とフェイルオーバー

ネットワークブートは様々な原因によりしばしば失敗し、外部からの検知が難しい。代表的な原因は、NFS サーバや TFTP サーバの過負荷、およびインストール用 DHCP サーバが別の DHCP サーバと干渉している場合などである。こうしたエラーはどれもエラーメッセージが出ないか、エラーを起こした実マシンのコンソールにしかエラーメッセージが出力されないため外部からの検知が困難である。

こうしたエラーをインストールサーバから検出するための機構がブートシーケンストラッカーである。ブートシーケンストラッカーはインストールサーバ上で起動し、syslog の追跡と SSH ポートのポーリングによってエラーを検出する (図 5):

syslog の追跡 DHCP, TFTP, および NFS サーバの出力するログからネットワークブートの異常を検知する。たとえば、正しい DHCP リクエストでは `DHCPDISCOVER`, `DHCPOFFER`, `DHCPREQUEST`, `DHCPACK` の 4 つのメッセージが実マシンと DHCP デーモンの間に流れるはずである。ここでもし `DHCPREQUEST` が実マシンから一定時間届いていないことが syslog からわかった場合、他の DHCP デーモンが存在し `DHCPOFFER` を先に送信してしまった可能性があると推測できる。

SSH ポートのポーリング ネットワークブートに成功した場合、実マシンの SSH ポートをポーリングすることで SSH デーモンの起動を検査する。もし一定時間立ち上がらない場合にはエラーと判断する。

ブートシーケンストラッカーがリブートエラーを検知した場合、失敗したマシンを IPMI で電源オフすることにより切り離し、またあらかじめ余分に起動しておいたマシンで失敗したマシンを置き換えることでフェイルオーバーする。

4.4 インストール

インストールでは実マシン用に変換された AMI をハードディスクに書き込む。実マシンごとのハードウェア構成の違いを吸収するために、インストーラは次のようにハードディスク容量や CPU の違いに対応し、ハードウェアに合ったドライバを含む ramdisk (initrd) を自動生成する:

(1) パーティションの作成とフォーマット: ディスクドライブ上にパーティションを作成しフォーマットする。デフォルトではあらかじめ管理者が指定したパーティション設定ファイル[⋆1]に従ってハードディスクをフォーマットする。パーティション設定ファイルではパーティションのサイズを絶対値だけでなくハードディスクの全体サイズに対する割合で指定できる。このため、実マシンごとのハードディスクのサイズの違いを吸収できる。もしユーザがファイルシステムをカスタマイズしたい場合、マシンを起動する `ec2-run-instances` コマンドにパーティション指定用の拡張オプション `--storage-conf` を指定することで、パーティション設定ファイルを渡すことができる。フォーマット後、AMI を書き込むためにマウントポイントにマウントする。

(2) AMI の転送と展開: 容量が比較的大きい AMI を多数の実マシンに効率的に転送するために、Dolly+[11] と同様に AMI をリング状にパイプライン転送する。この転送時間はマシン数に対して O(1) である。そして、転送した AMI をステップ 1 でマウントしたローカルディスク上に展開する。

(3) カーネルのインストールと initrd の生成: 実マシンの CPU アーキテクチャに対応した最新のカーネルパッケージをダウンロードしインストールする。インストーラは AMI が提供するディストリビューションごとのパッケージ管理コマンドと CPU アーキテクチャごとのカーネルパッケージ名を知っており[⋆2]、これらより最新のカーネルを自動的にインストールする。加えて、インストールしたカーネル用に initrd を作成しなおす。この initrd の生成も、ディストリビューションごとの方式の違いをインストーラが吸収し自動的に実行する。

(4) ブートローダのインストール: ステップ 3 でインストールしたカーネルを起動するために、ハードディスクの Master Boot Record (MBR) にブートローダ GNU GRUB[⋆3] をインストールする。このインストールもインストーラがディストリビューション固有の方法で自動的に行う。

(5) ネットワーク情報の設定: 実マシンのホスト名や IP アドレスなどのネットワーク情報を、ディストリビューション固有の方法で自動的に設定する。

(6) SSH 公開鍵の追加: インストール後に ssh ログインできるようにするために、root アカウントの authorized_keys にユーザの ssh 公開鍵を追加する。

インストールサーバは以上の処理を行うインストーラ本体を `scp` で実マシンにコピーし、`ssh` で実行する。エラー処理とリカバリのために、インストーラの出力 (標準出力と標準エラー出力) および終了コードを ssh 経由でインストールサーバに転送する。

4.5 ネットワークブートサービスの停止

インストール後にローカルディスクから起動するために、ネットワークブートサービスを

⋆1 指定できる設定ファイルのフォーマットは `setup-storage` コマンド (`http://www.informatik.uni-koeln.de/fai/doc/man/setup-storage.html`) に準拠する

⋆2 たとえば、debian で 64bit CPU 用の最新のカーネルをインストールするコマンドは
```
% aptitude install linux-image-2.6-amd64
```
である

⋆3 `http://www.gnu.org/software/grub/`

無効にする。具体的には、TFTP サーバが実マシンに配布するブート設定をネットワークブートからローカルブートに変更する。

4.6 リブート (二回目)

SSH で実マシンをリブートする。リブートした実マシンはふたたび PXE ブートシーケンスに入り、GRUB を通じてインストールされた Linux カーネルを起動する。この二度目のリブートもブートシーケンストラッカーが追跡する。もし失敗した場合には、一回目のリブートと同様にフェイルオーバーする。

5. 評　　価

本システムのインストール処理の特徴は、AMI の転送処理以外はノードごとに完全に独立で行われる点である。そこで、実マシンの台数を増やした場合のインストーラのスケーラビリティを見るために、ボトルネックとなる可能性のある AMI の転送部分と、実際のインストール性能を計測した。以下の実験で用いた実マシンおよびインストールサーバのスペックは、CPU Intel Xeon E5504、メモリ 16GB、ネットワーク Gigabit Ethernet である。転送にはあらかじめ実マシン用に変換した約 1.4 GB の AMI ファイルを用いた。

性能のボトルネックとなる可能性のある AMI の転送性能を図 6 に示す。なお比較のために、一台のマスターノードからすべてのノードにイメージを転送するスター型トポロジと、本システムで採用したリング型トポロジで転送した場合の性能を計測した。グラフからわかるように、スター型トポロジはマスターノードがボトルネックであるためノード数の増加に対して線形で転送時間が伸びる。一方でリング型にはボトルネックが存在せず、パイプライン転送によってノード数が増加した場合でも O(1) の時間で転送が完了していることがわかる。

実際のインストール性能を図 7 に示す。これによると、1 台あたりのインストール時間は約 900 秒であるが、台数が増えるにつれて 1 台あたり約 40 秒ずつインストール時間が増加する。この原因は、インストールサーバとした AMD64 版 Linux カーネルの NFS サーバのバグにより、同時に NFS のマウントリクエストが到着した場合にリクエストを取りこぼすというエラーを回避するために、ノードを再起動する間隔を通常の 1 秒から 30 秒に延長したためである。将来このバグが修正されることにより、台数が増えた場合でもほぼフラットなインストール時間になることが期待できる。

6. 関 連 研 究

NewServers 社は我々のシステムと同様に実マシンを提供する IaaS を提供している[13]。

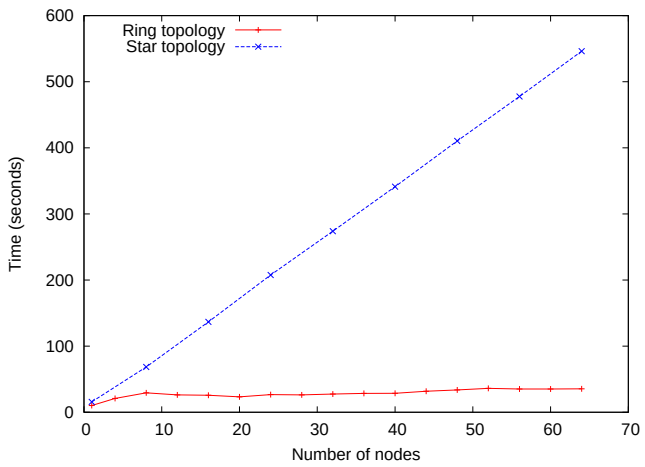

図 6 AMI ファイル (1.4GB) のスタートポロジおよびリングトポロジによる転送時間

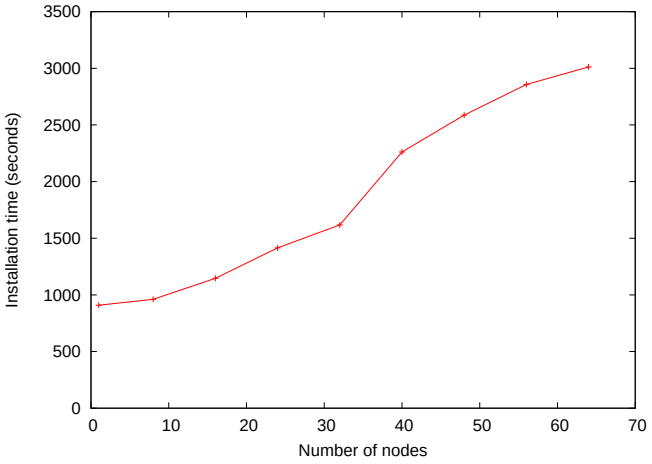

図 7 実マシン台数を増やした場合のインストール性能のスケーラビリティ

価格はたとえばスモールサーバ (Intel Xeon 2.8 GHz, 1GB メモリ, 36GB SCSI Disk) が $0.11/h であり、EC2 のスモールタイプ (Xeon 1.0-1.2GHz 相当、1.7GB メモリ、160GB ストレージ) の $0.085/h と比較するとストレージ容量は減るものの CPU 性能は上であり価格も同等である。ここからわかるように、実マシンでも仮想マシンベースの IaaS と価格競争力のあるサービスを提供できることがわかる。ただし EC2 の VM イメージをインストールできないため、我々のシステムのような EC2 とのシームレスな混合クラウドを形成することはむずかしい。Grid'5000[5] はフランスの研究プロジェクトであり、OS やミドルウェアの研究用環境を物理マシン上にセットアップできる。ただしセットアップには独自のインストーラ用設定を書く必要があり、NewServers 社の IaaS と同様に EC2 の VM イメージをインストールすることはできない。Eucalyptus[14] は EC2 互換なプライベートクラウドを構築するためのツールであり、Xen による VM 環境を EC2 と同様に構築することができる。ただし VM を対象としているため、実マシンをセットアップできない。

7. まとめと課題

我々はオンデマンドで実マシンを提供するクラウドサービスを提案した。これによって、従来の VM ベースの IaaS では本来の性能を出せなかった種類の HPC アプリケーションが効率的に実行可能となった。また、EC2 の AMI から実マシンをセットアップ可能とすることによって、あらかじめ MPI など主要なアプリケーションがセットアップされた環境をすぐに使いはじめることができる。その上、EC2 の VM マシンを実マシンと組み合わせることで、ソフトウェア環境的にシームレスな混合クラウド環境を構築できる。

今後の課題として、このサービスを各種研究所や大学研究室のサーバに適用することで、HPC 用クラウド計算リソースとして提供することを考えている。この活動の一環として、我々は第 2.2 節で挙げた最適化系 HPC アプリケーションをブラウザから実行できる最適化問題オンラインソルバープロジェクト[15] を進めている。このシステムでは、平常時は研究室内に構築された Bare metal cloud 上でジョブを実行し、高負荷時には同じソフトウェア環境の EC2 ノードを動的に追加することでスループットを増大する。

参 考 文 献

1) Advanced Micro Devices, Inc.: AMD Core Math Library (ACML). http://developer.amd.com/cpu/libraries/acml/Pages/default.aspx.
2) Amazon Web Services LLC: Amazon Elastic Compute Cloud (Amazon EC2). http://aws.amazon.com/ec2/.
3) Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing, *EECS Department University of California Berkeley Tech Rep UCBEECS200928*, No.UCB/EECS-2009-28, pp.07–013 (2009).
4) Bjerke, H.K.: HPC Virtualization with Xen on Itanium, Master's thesis, European Organization for Nuclear Research (CERN) (2005).
5) Bolze, R., Cappello, F., Caron, E., Dayde, M., Desprez, F., Jeannot, E., Jegou, Y., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Primet, P., Quetier, B., Richard, O., Talbi, E.-G. and Touche, I.: Grid'5000: A Large Scale And Highly Reconfigurable Experimental Grid Testbed, *International Journal of High Performance Computing Applications*, Vol.20, No.4, pp.481–494 (2006).
6) Dongarra, J.: Basic Linear Algebra Subprograms Technical Forum Standard, *High Performance Applications and Supercomputing*, Vol.16, No.1–2, pp.1–199 (2002).
7) Fujisawa, K., Nakata, K., Yamashita, M. and Fukuda, M.: SDPA Project: Solving Large-scale Semidefinite Programs, *Journal of the Operations Research Society of Japan*, Vol.50, No.4, pp.278–298 (2007).
8) Goto, K. and Geijn, R. A. v.d.: Anatomy of high-performance matrix multiplication, *ACM Trans. Math. Softw.*, Vol.34, No.3, pp.1–25 (2008).
9) Intel Corporation: Intel Math Kernel Library (Intel MKL). http://software.intel.com/en-us/intel-mkl/.
10) Litke, A. and Gibson, D.: libhugetlbfs. http://libhugetlbfs.ozlabs.org/.
11) Manabe, A.: Dolly+ Home Page. http://corvus.kek.jp/~manabe/pcf/dolly/index_J.htm.
12) National Institute of Standards and Technology: The NIST definition of cloud computing. http://csrc.nist.gov/groups/SNS/cloud-computing/.
13) NewServers, Incorporated: NewServers: Bare Metal Cloud. http://www.newservers.com/.
14) Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L. and Zagorodnov, D.: The Eucalyptus Open-Source Cloud-Computing System, *IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2009)*, Los Alamitos, CA, USA, IEEE Computer Society, pp.124–131 (2009).
15) Online solver project: SDPA Online Solver. http://sdpa.indsys.chuo-u.ac.jp/portal/.
16) Rackspace US, Inc.: the rackspace cloud. http://www.rackspacecloud.com/.
17) Youseff, L., Wolski, R., Gorda, B. and Krintz, R.: Paravirtualization for HPC Systems, *Proceedings of Workshop on Xen in High-Performance Cluster and Grid Computing*, Springer, pp.474–486 (2006).
18) 安井雄一郎，藤澤克樹，笹島啓史，後藤和茂：大規模最短路問題に対するダイクストラ法の高速化，日本オペレーションズ・リサーチ学会和文論文誌 (掲載予定) (2010).