論　文

インタークラウドオートスケールにおける仮想計算機展開機能の高速化

前野　洋史† 神屋　郁子†† 下川　俊彦††

Study about High-Performance Virtual Machine Deployment on Inter-Cloud Autoscale

Hiroshi MAENO†, Yuko KAMIYA††, and Toshihiko SHIMOKAWA††

あらまし　我々は、複数のクラウドを仮想的に 1 つのクラウドとして利用可能にする、複数クラウドを利用したサーバ広域分散配置システム:Soarin を開発している。一般的なクラウドではオートスケール機能により必要に応じて仮想計算機の台数を増減できる。Soarin では、複数のクラウドを跨って仮想計算機の台数を増減するインタークラウドオートスケールを実現している。インタークラウドオートスケールを実現するためには、同一サービスを提供可能な仮想計算機を複数のクラウドに展開する必要がある。従来の Soarin では、サービスを提供するために必要なアプリケーションを組み込んだ仮想計算機を予め構築し、その仮想計算機のディスクイメージを必要に応じて各クラウドに転送・登録する時間が必要であった。本研究では、仮想計算機の展開を高速化する手法を提案する。各クラウドで準備されている最小構成のディスクイメージを使用して仮想計算機を起動した後、構成管理ツールを利用して、サービス提供可能な状態に設定する。提案手法を実装し、有効性を評価するため、仮想計算機の展開時間を従来手法と提案手法で比較した。その結果、提案手法の有効性が確認できた。

1. はじめに

近年のクラウドコンピューティングの普及により、オートスケールの利用が容易になった。オートスケールにより、同一の機能を持った計算機を動的に増減することができる。すなわち必要に応じて計算機の台数を増減し、計算機システムの処理能力が増減できる。しかし、一般的なオートスケールでは、ネットワーク性能の向上は困難である。ここでネットワーク性能とは、サーバシステムとクライアントシステム間の帯域幅や RTT のことである。ネットワーク性能を向上させる方法として、サーバシステムを広域に分散配置する方法がある。これにより様々な場所でインターネットに接続することになり、提供する帯域幅が向上する。また、クライアントを近傍のサーバに誘導することで RTT も改善できる。

広域分散配置型のサーバシステム構築を支援するために、我々は、複数クラウドを利用したサーバ広域分散配置システム:Soarin [1] を開発している。Soarin は、複数のクラウドを仮想的な一つのクラウドとして利用可能にする。一般的なオートスケールでは単一のクラウド内で計算機を増減する。Soarin では複数のクラウドを跨った仮想的なクラウド内で計算機を増減する。この機能のことをインタークラウドオートスケールと呼ぶ。インタークラウドオートスケールを実現するためには、各クラウドに同一のサービスを提供可能な仮想計算機を展開しなければいけない。従来、Soarin ではサービス提供をするために必要なアプリケーションを組み込んだ仮想計算機を予め構築し、その仮想計算機のディスクイメージを必要に応じて各クラウドに転送・登録していた。その後、各クラウドで仮想計算機を起動していた。しかしこれには転送時間および登録時間が必要で、仮想計算機の展開に時間がかかるという問題があった。

本研究の目的は、Soarin における仮想計算機展開時

† 九州産業大学大学院情報科学研究科, 福岡市
Guraduate School of Information Science, Kyushu Sangyo University, Fukuoka
†† 九州産業大学情報科学部, 福岡市
Faculty of Information Science, Kyushu Sangyo University,Fukuoka

間を短縮することである。

本論文は全 6 章から成り、構成は以下のとおりである。2. では複数クラウドを利用したサーバ広域分散配置システム Soarin について説明する。3. では本研究で提案する高速化した仮想計算機展開機能の設計について述べ、 4. でその実装について述べる。5. で本研究を評価し、まとめと今後の課題について 6. で述べる。

2. 複数クラウドを利用したサーバ広域分散配置システム Soarin

2.1 Soarin の概要

我々は、インターネット上で提供されているサービスにおけるサーバ・クライアント間のネットワーク性能の向上を目的として、複数クラウドを利用したサーバ広域分散配置システム Soarin を開発している。

Soarin では、上述のように複数のクラウドを仮想的な一つのクラウドとして利用可能にする。その機能の 1 つに、クラウド間を跨ったオートスケールであるインタークラウドオートスケールがある。これを実現するためには、複数のクラウド上で、同一のサービスを提供可能な仮想計算機を起動する必要がある。このために、サービスやコンテンツを準備した状態の仮想計算機を構築し、そのディスクイメージを予めそれぞれのクラウド上に転送する。そして、サービス提供時に、サービス提供者の要求に応じて選択されたクラウド上で仮想計算機を起動する。これを本研究では、「仮想計算機の展開」と呼び、Soarin ではこの機能を仮想計算機展開機能として実装している。

サービス提供開始後、仮想計算機の負荷の変動などにより、処理能力や帯域が必要になった場合には、Soarin が次の仮想計算機をどのクラウド上で起動するか、サービス提供者の要求や現在の負荷の状況から判断し、選択されたクラウド上で仮想計算機を増設する。このとき、サービス提供者は、クラウドの選択指針である「ポリシー」を記述するだけで、複数のクラウドを意識することなく利用することができる。

2.2 Soarin の問題点

Soarin を用いて、複数クラウド上で同一のサービスを提供するためには、予め作成しておいたディスクイメージを各クラウドへ転送して登録する必要がある。そのため、ディスクイメージの転送時間と登録時間が発生し、仮想計算機の展開に時間がかかるという問題が生じる。これは、動的な仮想計算機の増設を行う上で大きな問題である。

3. 仮想計算機展開機能

2.2 で述べた問題を解決するため、本研究では Soarin の仮想計算機展開機能を高速化した。これには構成管理ツールを利用した。

3.1 構成管理ツール

構成管理ツールとは、ミドルウェアやアプリケーションなどの環境を自動構築するツールである。構成管理ツールでは、設定ファイルに記述した内容に従って、計算機の状態を変更する。代表的なものとして、Chef [2], Puppet [3], Ansible [4] がある。

3.2 提案する仮想計算機展開機能の高速化手法

従来の手法では、2.1 で述べたように、サービスやコンテンツを準備した状態の仮想計算機を予め構築し、その仮想計算機のディスクイメージを作成していた。その後、作成したディスクイメージを各クラウドに転送・登録し、仮想計算機を起動することで、複数クラウド上への仮想計算機の展開を可能とした。

一方、本研究での提案手法では、ディスクイメージの作成を必要としない。クラウドには予め最小構成の OS がインストールされたディスクイメージが準備されている。提案手法では、このディスクイメージを利用して仮想計算機を起動する。その後、構成管理ツールにより、サービスの提供に必要なソフトウェアやコンテンツを用意し、サービス提供可能な状態に設定する。これにより、予め作成したディスクイメージの転送と登録をすることなく、複数クラウド上への仮想計算機の展開が可能となる。

4. 高速化した仮想計算機展開機能の実装

仮想計算機展開機能の実装には、構成管理ツールとして Ansible、言語として Ruby [5]、設定ファイルの記述形式として YAML [6] を使用した。Ansible を使用した理由は、代表的な構成管理ツール 3 つを比較した結果、Soarin と最も相性が良いと判断したためである。比較した 3 つの構成管理ツールを表 1 に示す。

表 1 構成管理ツールの比較

	Chef	Puppet	Ansible
プログラミング言語	Ruby	Ruby	Python
設定ファイル	Ruby	独自 DSL	YAML
エージェントのインストール	必要	必要	不要

Chef と Puppet は管理対象マシンに管理サーバから指示を受け取るエージェントをインストールする必

要がある。一方、Ansible では管理対象マシンにエージェントが不要であり、管理対象マシン上で ssh サーバが動いていれば動作する。一般的に、仮想計算機には ssh サーバがインストールされている。したがって、Ansible を仮想計算機に対して実行する際には、管理サーバのみ用意すればよい。また、Ansible の設定ファイルは YAML で記述する。YAML は記述が容易であるため、利用者にとって分かりやすいという利点がある。Soarin でも設定ファイルの記述に YAML を使用している。これらの理由から、本研究では仮想計算機展開機能の基盤として Ansible を採用した。

仮想計算機展開機能では、増設した仮想計算機に対して Ansible を実行する。実行にあたっては、実行対象となる仮想計算機の IP アドレスを指定する。また、Ansible の実行時には仮想計算機上で ssh サーバが動作している必要があるため、仮想計算機側で ssh 接続が可能になってから Ansible を実行する。

5. 評　　価

本研究の有効性を評価するために、仮想計算機展開時間を従来手法と提案手法で比較した。

5.1 評 価 環 境

評価環境として、Amazon EC2 の東京リージョンと九州産業大学内に構築した OpenStack 環境及び CloudStack 環境を利用した。今回利用した OpenStack 環境は 1 台構成で構築し、CloudStack 環境は 2 台構成で構築した。OpenStack と CloudStack の環境構築に使用したハードウェアの構成を表 2〜3 に示す。

表 2　OpenStack のハードウェア構成

OpenStack	Icehouse
OS	CentOS6.5
CPU	Intel(R) Celeron(R) CPU G530 @ 2.40GHz
メモリ	8GB
NIC	1000BASE-T

表 3　CloudStack のハードウェア構成

CloudStack	4.3.1
OS	Ubuntu 14.04 LTS
CPU	Dual-Core AMD Opteron(tm) Processor 1210
メモリ	8GB
NIC	1000BASE-T

5.2 評 価 方 法

本論文では、仮想計算機展開時間を従来手法と提案手法で比較した。本評価でクラウド上に展開する仮想計算機は、Apache2.4.7 パッケージがインストールされ、自動起動設定がされているものである。

従来手法で仮想計算機を展開する場合、ディスクイメージの転送、ディスクイメージの登録、仮想計算機の起動の 3 つの手順が必要である。ここで、ディスクイメージの転送時間を T_t、ディスクイメージの登録時間を T_r、仮想計算機の起動時間を T_e、従来手法での仮想計算機展開時間を T_c とすると、以下の式が成り立つ。

$$T_c = T_t + T_r + T_e$$

提案手法で仮想計算機を展開する場合、仮想計算機の起動、Ansible の実行の 2 つの手順が必要である。ここで、仮想計算機の起動時間を T_e、Ansible の実行時間を T_a、提案手法での仮想計算機展開時間を T_p とすると、以下の式が成り立つ。

$$T_p = T_e + T_a$$

各手順に要する時間を 5 回ずつ計測し、平均を算出した。各手順の計測方法について以下で説明する。

5.2.1 T_t の計測方法

T_t は、九州産業大学内にあるディスクイメージ転送用物理計算機から各クラウド上へディスクイメージを転送した際にかかる時間を計測した。各クラウドに転送したディスクイメージは Ubuntu 12.04 LTS に Apache2 をインストールしたものであり、サイズは約 1.5GB である。本評価のネットワーク構成を図 1 に示す。

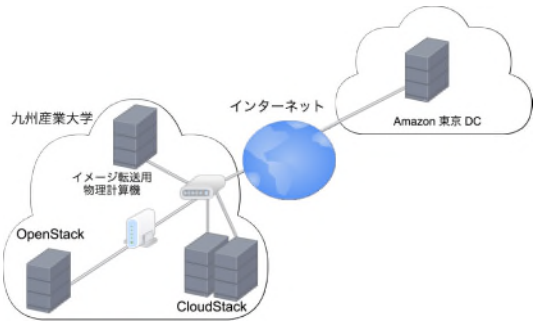

図 1　評価におけるネットワーク構成

Amazon EC2 でディスクイメージを利用可能にするには、Amazon EC2 の API を使用する必要がある。Amazon EC2 の API を使用するために Amazon EC2 API Tools というコマンドラインツールを利用した。Amazon EC2 API Tools をディスクイメージ転送用

物理計算機にインストールし、ec2-import-instance コマンドによって T_t を計測した。ec2-import-instance コマンドは、ディスクイメージの転送と登録を同時に行う。そのため、転送と登録の時間を正確に測ることは難しい。しかし、ディスクイメージの転送が終了した時点の時間は標準出力に表示される。今回はその時間を Amazon EC2 の T_t とする。

CloudStack と OpenStack の評価環境はディスクイメージ転送用物理計算機と同じネットワーク内にある。LAN 内の転送では速度が速いのは当然であるため、Amazon EC2 と公平な条件ではない。よって、CloudStack と OpenStack に関しては、Amazon EC2 と同じ距離のネットワーク上に構築されていると仮定し、ディスクイメージ転送用物理計算機から Amazon EC2 上の仮想計算機に対してのディスクイメージ転送時間を、CloudStack と OpenStack の T_t とした。この計測には FTP を使用した。

5.2.2 T_r の計測方法

T_r は、各クラウドで提供されている API を利用して計測した。Amazon EC2 へのディスクイメージの登録には、転送時間の計測に使用した ec2-import-instance コマンドを使用した。ec2-import-instance コマンドでは、ディスクイメージ転送の終了後に、ディスクイメージ登録の処理が引き続き行われる。この処理の進捗状況は ec2-describe-conversion-tasks というコマンドで確認できる。T_r の計測をするために、ec2-describe-conversion-tasks コマンドを使った計測用プログラムを作成した。作成した計測用プログラムによって、Amazon EC2 への T_r を計測した。

OpenStack へのディスクイメージ登録には、OpenStack のコンポーネントであり、ディスクイメージの管理をしている Glance を操作するコマンドを利用する。glance image-create というコマンドでディスクイメージを登録できる。コマンドの実行時間を計る time コマンドにより、glance image-create コマンドの実行時間を計測した。

CloudStack へのディスクイメージの登録は、CloudStack が提供している API を実行するプログラムを作成して行った。registerTemplate という API により、ディスクイメージの登録ができる。ディスクイメージ登録の進捗状況は listTemplates という API により確認できる。これらの API を利用した計測用プログラムを作成して計測を行った。

5.2.3 T_e の計測方法

従来手法の T_e は、従来の仮想計算機展開機能により仮想計算機を起動し、Apache2.4.7 が立ち上がるまでの時間を計測した。Apache2.4.7 の起動確認をするプログラムを作成し、従来手法の T_e を計測した。

提案手法の T_e は、本研究で提案した仮想計算機展開時間により仮想計算機を起動し、sshd が立ち上がるまでの時間を計測した。従来手法の場合、仮想計算機を起動すれば Apache2.4.7 も起動する。しかし、提案手法では、仮想計算機を起動した後に Ansible を実行し、Apache2.4.7 をインストールする必要がある。Ansible を実行するためには、sshd が起動していなければならない。したがって、提案手法の T_e は、仮想計算機が起動した後、sshd が立ち上がるまでの時間とした。sshd の起動確認をするプログラムを作成し、提案手法の T_e を計測した。

5.2.4 T_a の計測方法

T_a は、本研究で提案した仮想計算機展開機能により、Ansible が実行されてから終了するまでの時間を計測した。

5.3 評価結果

従来手法で仮想計算機を展開する場合の 3 つの手順について計測した結果を表 4〜6 に示す。提案手法で仮想計算機を展開する場合の 2 つの手順について計測した結果を表 7〜8 に示す。仮想計算機展開時間を従来手法と提案手法で比較した結果を表 9 に示す。

表 9 の結果より、Amazon EC2 においては提案手法の方が速く、CloudStack と OpenStack についても提案手法の方が速いということが分かる。

従来手法と提案手法をクラウド毎に比較したグラフを図 2 に示す。従来手法と提案手法の T_e について比べると、ほぼ同じ時間であることが分かる。T_e 以外を比べると、提案手法の T_a の方が短いことが分かる。図 2 より、利用した 3 つのクラウド全てにおいて、提案手法の方が仮想計算機展開時間が短いことが分かる。

表 4 ディスクイメージ転送時間

	Amazon EC2	OpenStack	CloudStack
1 回目	54	26	26
2 回目	49	29	23
3 回目	49	28	26
4 回目	49	25	25
5 回目	50	26	26
平均	50.2	26.8	25.2

(単位：秒)

表 5　ディスクイメージ登録時間

	Amazon EC2	OpenStack	CloudStack
1 回目	843	63	115
2 回目	878	62	115
3 回目	815	63	115
4 回目	823	60	116
5 回目	818	60	115
平均	835.4	61.6	115.2

(単位：秒)

表 6　従来手法の仮想計算機起動時間

	Amazon EC2	OpenStack	CloudStack
1 回目	42	39	30
2 回目	44	40	30
3 回目	42	39	32
4 回目	41	38	31
5 回目	40	42	32
平均	41.8	39.6	31.0

(単位：秒)

表 7　提案手法の仮想計算機起動時間

	Amazon EC2	OpenStack	CloudStack
1 回目	50	43	30
2 回目	50	43	31
3 回目	41	44	31
4 回目	52	42	28
5 回目	52	44	30
平均	49.0	43.2	30.0

(単位：秒)

表 8　Ansible 実行時間

	Amazon EC2	OpenStack	CloudStack
1 回目	16	62	29
2 回目	11	61	29
3 回目	17	63	30
4 回目	16	59	30
5 回目	12	60	29
平均	14.4	61.0	29.4

(単位：秒)

表 9　仮想計算機展開時間の比較

	Amazon EC2	OpenStack	CloudStack
従来手法の平均時間	927.4	128.0	171.4
提案手法の平均時間	63.4	104.2	59.4
差	864.0	23.8	112.0

(単位：秒)

5.4 考　　察

評価実験の結果、今回利用した 3 つのクラウド全てにおいて、提案手法の方が仮想計算機展開時間が短いという結果が得られた。したがって、従来手法よりも提案手法の方が動的な仮想計算機の増設をする際には有効であると言える。

Amazon EC2 に仮想計算機を展開する場合、従来手法では平均 927.4 秒かかった。提案手法で仮想計算機を展開する場合と比べ、864.0 秒の差がある。これはディスクイメージの転送と登録に使用している、Amazon EC2 専用コマンドの処理時間が長いことに原因がある。Amazon EC2 で仮想計算機の起動をするためには、AMI と呼ばれる Amazon EC2 専用のディスクイメージが必要である。AMI 以外のディスクイメージを Amazon EC2 に登録するには、ec2-import-instance コマンドを使わなければならない。表 4 と表 5 の Amazon EC2 の平均値を合計すると、ec2-import-instacne コマンドの処理が終わるまでに約 15 分もの時間がかかっている。したがって、従来手法で Amazon EC2 に仮想計算機を展開する場合には、927.4 秒もの時間がかかってしまった。一方、提案手法では Amazon EC2 で初めから利用可能な AMI を利用して、仮想計算機を起動している。このため、従来手法よりも提案手法の方が仮想計算機展開時間が圧倒的に短いという結果になった。

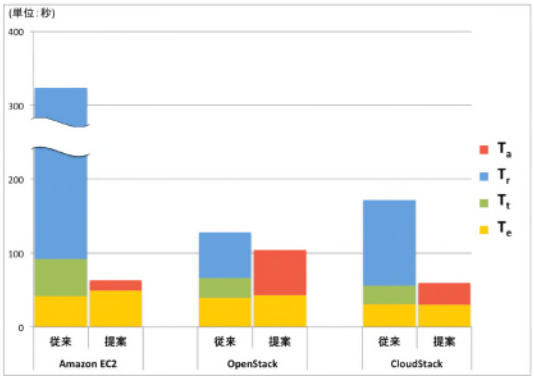

図 2　仮想計算機展開時間の比較グラフ

OpenStack に仮想計算機を展開する場合、従来手法では平均 128.0 秒かかった。従来手法と提案手法の時間を比較した結果、提案手法の方が 23.8 秒ほど速いという結果が出ている。また図 2 より、利用した 3 つのクラウドの中で、提案手法の T_a が最も長いことが分かる。これは今回利用した OpenStack 環境に原因があると考えられる。OpenStack は一般的に、2 台の物理計算機を使用した構成が推奨されている。本評価に利用した OpenStack 環境は 1 台の物理計算機を使用して構築したものである。したがって、2 台構成で構築した CloudStack と比べて負荷が多くかかっているため、T_a が長くなっていると考えられる。

CloudStack に仮想計算機を展開する場合、従来手法では平均 171.4 秒かかった。従来手法と提案手法の

時間を比較した結果、提案手法の方が 112.0 秒ほど速いという結果が出ている。また図 2 より、従来手法では T_r が長いことが分かる。これはディスクイメージの登録に使用している、CloudStack 専用 API に原因がある。使用した CloudStack 専用 API では、ディスクイメージが配置されている Web サーバの URL を引数として指定する必要がある。指定した Web サーバの URL からディスクイメージをダウンロードした後、登録処理が引き続き行われる。このため、ディスクイメージのダウンロード時間が余計にかかることにより、T_r が長くなっていると考えられる。

今回、従来手法で転送したディスクイメージのサイズは、1.5GB と小さなものである。そのため、ディスクイメージの転送時間は短かった。しかし、ディスクイメージの形式やディスクイメージのルートディスクサイズによっては、ディスクイメージのサイズが大きくなり、従来手法では転送時間が増えると考えられる。提案手法を用いると、ディスクイメージを転送する必要がない。そのため、従来手法で問題であったディスクイメージ転送時間の短縮が見込め、急な負荷にも対応可能となる。

6. ま と め

我々は、インターネット上で提供されているサービスにおけるサーバ・クライアント間のネットワーク性能の向上を目的として、複数クラウドを利用したサーバ広域分散配置システム Soarin を開発してきた。しかし Soarin には、仮想計算機の展開に時間がかかるという問題があった。本研究では、この問題に対して、仮想計算機の展開時間を短縮する手法を提案した。

従来の手法では、サービスを提供するために必要なアプリケーションを組み込んだ仮想計算機を予め構築し、その仮想計算機のディスクイメージを作成していた。その後、作成したディスクイメージを各クラウドに転送・登録し、仮想計算機を起動することで、複数クラウド上への仮想計算機の展開を可能とした。本研究の提案手法では、各クラウドで準備されている最小構成のディスクイメージを使用して仮想計算機を起動した後、構成管理ツールにより、起動した仮想計算機をサービス提供可能な状態に設定する。これにより、ディスクイメージの転送・登録をせずに、複数クラウド上への仮想計算機の展開が可能となった。

評価実験では、本研究の有効性を評価するために、仮想計算機展開時間を従来手法と提案手法で比較した。その結果、評価で利用した 3 つのクラウド全てにおいて、提案手法の仮想計算機展開時間の方が従来手法よりも短かった。評価結果より、従来手法よりも提案手法の方が有効であるという結果が得られた。

今後の課題として、設定ファイルの設計の見直しと、機能を改良することがある。本研究では、仮想計算機展開機能により、複数クラウド上に起動させた仮想計算機全てに対して同じ Playbook [7] を実行した。サービス提供者によっては、クラウド毎に使用する Playbook を分けたい場合やサービス毎に使用する Playbook を分けたい場合もあると考えられるからである。

謝辞 本研究は JSPS 科研費 26330124 の助成を受けている。

文 献

[1] 神屋郁子, 川津祐基, 下川俊彦, 吉田紀彦, “複数のクラウドを利用したサーバ広域分散配置システムの構築 ”, 電子情報通信学会論文誌, J96-B:10, 1164-1175 (October, 2013)

[2] “Chef — IT automation for speed and awesomeness — Chef”, https://www.getchef.com/chef/

[3] “Puppet Labs: IT Automation Software for System Administrators”, http://puppetlabs.com/

[4] “Ansible is Simple IT Automation”, http://www.ansible.com/

[5] “オブジェクト指向スクリプト言語 Ruby”, https://www.ruby-lang.org/ja/

[6] “The Official YAML Web Site”, http://www.yaml.org/

[7] “Playbooks — Ansible Documentation”, http://docs.ansible.com/ansible/playbooks.html