様々な計算リソースに対応した プロビジョニングシステムの開発

湯原　基貴† 　佐藤　賢一‡ 　林　秀樹‡

†NTT 情報流通プラットフォーム研究所　〒180-8585 東京都武蔵野市緑町 3-9-11

E-mail: {yuhara.motoki,s.kenichi,hayashi.hideki}@lab.ntt.co.jp

あらまし　クラウドの特徴である「On-demand self-service」を実現するため、クラウド基盤には、ユーザからの要求に応じて計算リソースを構築するプロビジョニングシステムが付随する。クラウドには IaaS、PaaS、SaaS などの様々なレイヤでのサービスが存在し、それぞれ扱う計算リソースが異なるため個別にプロビジョニングシステムを開発しなければならなかった。そこで我々は、計算リソースのライフサイクルは限定された定型的な動作で表現することが可能であるとの解説を立て、この仮説に基づいたプロビジョニングシステムの開発を行った。本プロビジョニングシステムでは計算リソースを 4 つの定型動作をもつ資源という抽象的な存在に変換して取り扱うことで、システムのコア部分を変更することなく異なる計算リソースをプロビジョニングする。本稿では資源という概念と開発したプロビジョニングシステムの効果について報告する。

キーワード　クラウドコンピューティング、プロビジョニングシステム、資源

Keyword　Clould Computing, Provisioning System, Resource

1. クラウドコンピューティング

クラウドコンピューティング（以下、クラウド）とはネットワーク経由で計算リソースを使用するサービスのことを示す。クラウドの発展は著しく、2010 年の市場規模は約 450 億円に達している[1]。

クラウドコンピューティングの特徴として「On-demand self-service」がある。これはサービスプロバイダに依存することなく、ユーザが必要に応じて自分自身が利用するための計算リソースをクラウドから借り出せることを示す[2]。サーバやストレージを自前で用意する従来の方式と比較して、ユーザは設備投資のリスクを軽減（CAPEX から OPEX へ）できるメリットがある。

この「On-demand self-service」を実現するため、クラウド基盤では計算リソースの元となる具体的な実機器やソフトウェアを制御する機能群だけでなく、ユーザに対して適切に計算リソースを提供できるよう運転管理する機能群も合わせて提供されている。我々の研究チームでは前者の機能群を実行系、後者の機能群を管理系と呼称している。本稿においてもこれらの用語を用いる（図 1）。

クラウド管理系には 3 つの大きな機能があり、プロビジョニングとはユーザのリクエストに応じて貸し出す計算リソースを構築する機能である。モニタリングは貸し出した計算リソースの利用状況や死活を監視する機能である。キャパシティ・プランニングは計算リソースの利用状況などからユーザに必要な計算リソースを見積もる機能である。本研究では「On-demand self-service」においてもっとも重要な機能であることからプロビジョニングシステムに着目する。

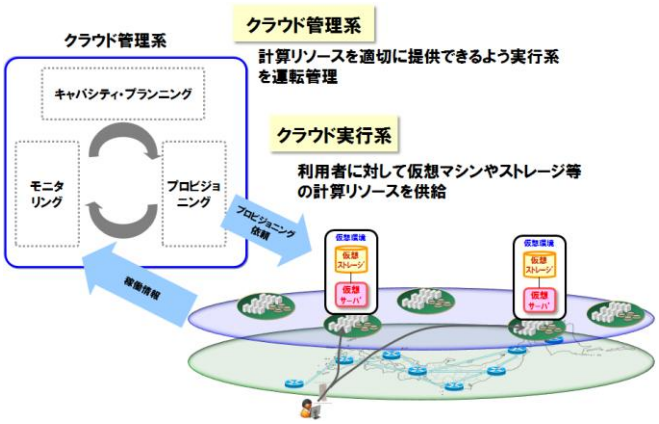


図 1　クラウドのシステム構成

クラウド実行系には利用者に対して計算リソースを提供するための実機器（サーバやストレージなど）やソフトウェア（仮想化ソフトウェア）などが存在する。

1.1. クラウドのサービス形態

クラウドには様々な種別が存在する。ここではサービス形態とサービスレイヤの観点からクラウドコンピューティングの種別を整理する。

1.2. サービス形態の観点からみたクラウドの種別

クラウドのサービス形態はパブリッククラウドとプライベートクラウドに分けられる。パブリッククラウドは不特定の利用者に対してクラウドのサービスを提供する形態である。主に一般消費者に対してクラウ

ドサービスを提供する際に見られる形態である。一方、プライベートクラウドとは特定の利用者グループが自身でクラウド基盤を保有し、グループ内利用者に限定してクラウドサービスを提供する。企業が自社の社員に対してサービスを提供する際の形態である。

1.3. サービスレイヤの観点からみたクラウドの種別

提供する計算リソースを基準にサービスレイヤからクラウドを分類すると、ハードウェアを計算リソースの対象とするのが HaaS（Hardware as a Service）である。これと同様に OS（Operating System）を対象とするのが IaaS（Infrastructure as a Service）、ミドルウェアを対象とするのが PaaS（Platform as a Service）、アプリケーションを対象とするのが SaaS（Software as a Service）である（図 2）。

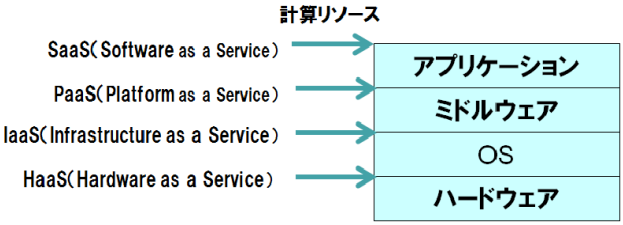


図 2　クラウドのサービスレイヤと計算リソース

2. 本研究の目的

2.1. 本研究のターゲット

パブリッククラウドでは IaaS では AmazonEC2[3]、PaaS では google app engine[4]など競合サービスが普及している。一方、プライベートクラウドの分野では現段階では主要なサービスプロバイダは存在していない。そこで我々はプライベートクラウドに対するクラウド基盤システムの提供者となることを主たる目的として、プライベートクラウドの基盤を構築するためのプロビジョニングシステムをターゲットとして研究を行うこととした。

2.2. プライベートクラウドにおけるプロビジョニングシステムの課題

プライベートクラウドではクラウドサービスの利用者自身がクラウド基盤を準備する必要がある。企業での利用を考えると、ソフトウェアの開発や試験を行うグループでは開発環境や検証環境を構築するための IaaS や PaaS を必要とするのに対して、人事を担当するグループでは勤怠管理サービスを構築するための SaaS が必要とされるなど、プライベートクラウドではニーズに応じて複数のサービスレイヤのクラウドサービスの提供が要求される。これに伴い、プロビジョニングシステムも複数のサービスレイヤを横断した計算リソースの操作を要求される。

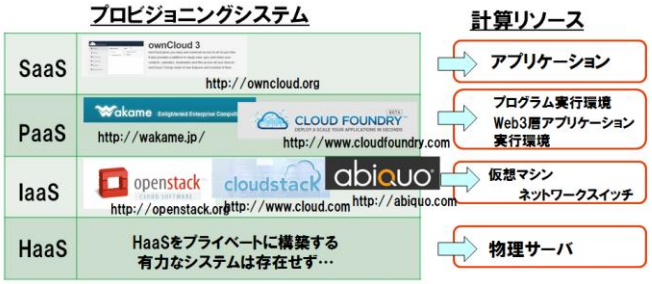


図 3　既存のプロビジョニングシステム

これに対して、プライベートクラウドで利用可能な既存のプロビジョニングシステムは各サービスレイヤの計算リソースに最適化して構築されており、1 つのプロビジョニングシステムで複数のサービスレイヤのプロビジョニングをサポートすることができない。図 3 より、OpenStack [5]や abiquo [6]は IaaS 環境を構築することを目的としたプロビジョニングシステムを持ち、サーバ（仮想マシン）、ネットワークをプロビジョニングの対象とし、ソフトウェア実行環境などは対象としていない。これとは逆に Cloud Foundry [7]は PaaS 環境を構築する OSS であり、ソフトウェア実行環境をプロビジョニングの対象とするが、サーバ、ネットワークなどはプロビジョニング対象外である。

このため、プライベートクラウドで新規サービスレイヤの提供が必要となった場合

・既存プロビジョニングシステムの改修
・新規プロビジョニングシステムの追加構築

のどちらかが必要となり、迅速なサービス展開ができない恐れがある。

そこで上記のようなプライベートクラウドでの迅速なサービス展開を行うために、本研究ではサービスレイヤを横断して様々な計算リソースに対応したプロビジョニングシステムの開発を目的とする（図 4）。

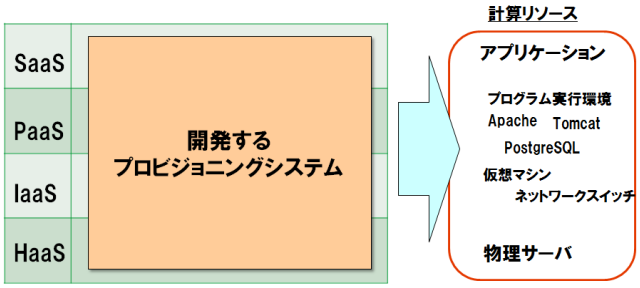


図 4　開発プロビジョニングシステム

3. 様々な計算リソースに対応したプロビジョニングシステムの開発

本研究が目的とする様々な計算リソースに対応したプロビジョニングシステムを実現するには

・任意の計算リソースを操作する仕組み

の他に、

・複数の計算リソースを協調動作させる仕組み

が必要不可欠である。例えば、仮想マシンをユーザにプロビジョニングすることを考えたとき、まずネットワークから IP アドレス（必要であれば VLAN も）をユーザに対して払い出し、当該 IP アドレスを仮想マシンに対して設定して起動しないとユーザがネットワーク経由で仮想マシンを利用することができないからである。

各項目について本研究での実現方式を説明する。

3.1. 検討、検証の対象とする計算リソース

本研究は任意のサービスレイヤの計算リソースをプロビジョニングの対象とすることが最終的な目標となるが、スケジュールの都合上、今回は需要が高い IaaS と PaaS の計算リソースに焦点を当てて検討、検証を行うこととした。

IaaS の具体的な計算リソースは仮想マシンとネットワーク（IP アドレスや VLANID）である。仮想マシンをネットワーク経由で操作する形式の IaaS を想定している。ネットワークに VLAN を使用することで異なるユーザ間で仮想マシンを操作できないよう隔離性を担保している。仮想マシン以外に LXC[5]などの Linux コンテナなどを用いて IaaS を構築する方式もあるが、プライベートクラウドにおける IaaS の主要な使い方である開発、検証環境の構築では OS のディストリビューションやバージョンも開発システムの要件として指定されていることが一般的であり、ゲスト OS がホスト OS に依存する Linux コンテナよりもゲスト OS を任意にインストール可能な仮想マシンを計算リソースと考える方が現実的である。

PaaS での具体的な計算リソースは Apache Http Server [9]、Apache Tomcat [10]、PostgreSQL [11]から構成される Web3 層アプリケーション実行環境を対象とする。この他にも分散 DB などを PaaS の計算リソースとして扱うのが一般的であるが必要最小限のアプリケーション実行環境をプロビジョニングできる能力を検証するという観点から、Web3 層アプリケーション実行環境を対象とすることとした。

3.2. 任意の計算リソースを操作する仕組み

我々の研究チームでは、計算リソースのライフサイクルは、プロビジョニングという観点から考えたとき、どのような計算リソースであっても、表 1 のような定型的な動作を繰り返すでのはないかと仮定した。

表 1 計算リソースの定型動作

動作	内容
割当	計算リソースの動作環境を確保する
起動	計算リソースをユーザが使用可能な状態にする
停止	計算リソースをユーザが使用不可能な状態にする
解放	計算リソースの動作環境を解放する

計算リソースとして仮想マシンを例にして考えると動作の内容は表 2 のようになる。表 2 から仮想マシンの作成、起動（ユーザの利用開始）、停止（ユーザの利用終了）、仮想マシンの削除までプロビジョニングで必要な一連の動作が定義できていることが分かる。

表 2 仮想マシンを対象にした定型動作

動作	内容
割当	仮想マシンに割当てるための（物理サーバの）CPU、メモリ、ディスクを確保する
起動	仮想マシンを起動する
停止	仮想マシンを停止する
解放	仮想マシンを削除し、確保していた CPU、メモリ、ディスクを解放する

この他にネットワークやアプリケーションサーバに対する定型動作とその内容を以下に示す。なお、ネットワークの実体は IP アドレスや VLANID であるが、ここではそれを保有する仮想的な計算リソースとして見做し動作を定義している。ネットワークに対して IP アドレスや VLNAID を割り当てることはユーザが占有して使用できる VLANID、IP アドレスを提供することを意味する。仮想マシンに対して割り当てられた VLANID と IP アドレスを設定して起動することで、隔離性のあるネットワークに接続された仮想マシンを起動し、利用することができる。

表 3 ネットワークを対象とした定型動作

動作	内容
割当	IP アドレス、VLANID の割当
起動	何もしない
停止	何もしない
解放	IP アドレス、VLANID の解放

表 4 アプリケーションサーバを対象とした定型動作

動作	内容
割当	AP サーバのインストール。AP サーバが使用する CPU、メモリ、ディスクの確保
起動	AP サーバの起動
停止	AP サーバの停止
解放	AP サーバのアンインストール

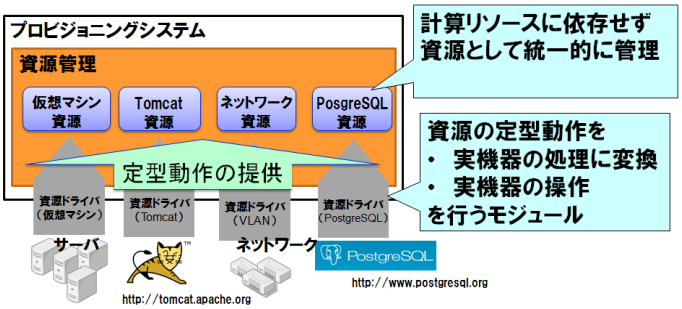


図 5 資源操作を対象とするプロビジョニングシステム

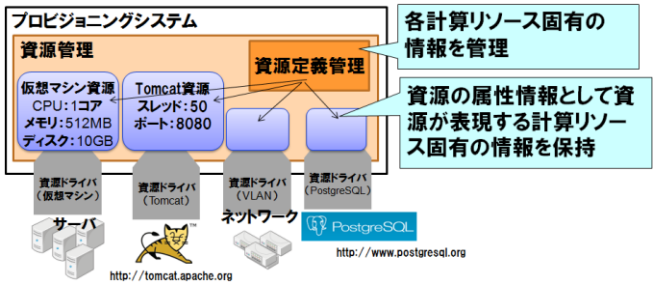
そこで我々の研究チームでは、システム上は定型動作（割当、起動、停止、解放）をもつ資源という抽象的なモデルとして計算リソースを扱い、ユーザが資源を操作すると、資源と対応する計算リソースの具体的な処理に変換して計算リソースを操作するプロビジョニングシステムのアーキテクチャを考案した（**エラー! 参照元が見つかりません。**）。

角丸四角で囲った資源（例：仮想マシン資源）はメモリ上に存在する論理的な情報であり、各計算リソースと対応するようカスタマイズされた資源である。資源は先ほど説明した 4 つの定型動作を資源が可能な操作としてもつ。この定型動作を、資源が共通して具備する操作の意味を込めて、資源 SPI と呼称する。

資源の下に位置している資源ドライバ（例：資源ドライバ（仮想マシン））は資源の定型動作を実機器の処理に変換し、実機器の操作を行うプログラムである。仮想マシン資源の起動という資源 SPI が呼び出されたとき、資源ドライバ（仮想マシン）は仮想マシンを起動するという処理に変換して実機器の操作を行う。Tomcat 資源の起動という資源 SPI が呼び出されたときは、資源ドライバ（Tomcat）が Tomcat アプリケーションサーバの起動という処理に変換して実機器の操作を行う。このように資源ドライバで実機器の処理に変換して操作を行うことで、資源は計算リソースに依存しない資源 SPI の提供を可能にする。

これまでに資源 SPI として提供される「資源の操作」について説明してきた。各資源の操作においてどのような情報を与えるべきかは計算リソースによって異なる。仮想マシンでは割当時に仮想マシンに与える CPU のコア数、メモリ・ディスク量をを必要とし、Tomcat においては worker スレッド数やポート番号を必要とする。そこで、資源ごとに資源属性という形で情報を持つことができるようにし、各計算リソース固有の情報を資源属性という形で保持できるようにした（**エラー! 参照元が見つかりません。**）。

資源定義管理機能とは資源の雛形となる情報を管理する機能である。資源がどの資源ドライバと対応しているか、資源属性としてどのような情報を持っている



かを管理している。資源の作成において、資源定義管

図 6 資源属性の設定

理の情報を用いて資源属性を設定することで、計算リソース固有の情報を資源に設定する。また、資源の操作時には資源ドライバに資源属性情報が渡される。資源ドライバは資源属性の情報を用いて実機器を操作することができる。また、資源ドライバによって資源属性の情報を更新することが可能であるため、実機器の動作結果を資源に反映することができる。

3.3. 複数の計算リソースを協調動作させる仕組み

計算リソースを協調動作させる必要性を仮想マシンとネットワークを例にして先ほど説明した。ある計算リソースを先にプロビジョニングし、その情報を用いて後続の計算リソースをプロビジョニングする関係をここでは計算リソース間の依存関係と呼称する。依存関係は仮想マシンとネットワークだけでなく、Web3 層アプリケーション実行環境を構成する計算リソース間にも存在する

Apache Tomcat は PostgreSQL から Datasource 設定のための IP アドレス、ポート番号を必要とする。Apache Http Server は Apache Tomcat から mod_proxy_ajp 設定のための URL、通信プロトコルを必要とする。従って、プロビジョニングの順序は、PostgreSQL、Apache Tomcat、Apache Http Server の順序でなければならない（図 7）。

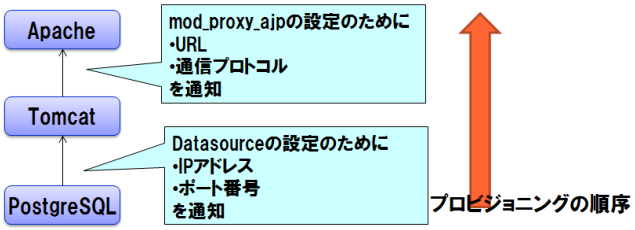


図 7 計算リソース間の依存関係

このようなプロビジョニングを可能とするために、本研究においては、資源間の依存関係を定義し、資源間の情報の授受やプロビジョニング順序を決定できる

ようにした。この定義情報を接続定義と呼称する。

3.3.1. 接続定義による依存関係の注入

接続定義は資源ごとに定義を行う。接続定義は資源の依存関係の定義と情報の授受の定義の 2 つが定義できる。

依存関係の定義とは、定義対象の資源が操作を行うためには依存する X 資源がどのような状態になっていなければならないかが定義されている。資源の操作の際にはこの情報を参照することで、依存関係のある資源の状態と整合性をとりながら資源をプロビジョニングすることができる。

情報の授受の定義とは、定義対象の資源と依存関係のある資源間でどの資源属性の情報を与えるあるいは受け取るか定義されている。この情報を参照することで定義対象の資源の操作前に依存関係にある資源から情報を設定、取得することができる。

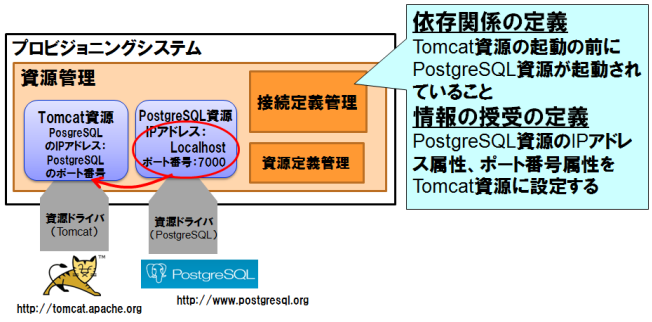


図 8 Apache Tomcat の接続定義例

図 8 は Apache Tomcat を対象とした接続定義の概要を表したものである。PostgreSQL との間の接続定義を記述している。プロビジョニングシステム上は接続定義は接続定義管理機能で管理を行う。

上図の接続定義について説明する。Tomcat 資源は「起動」の操作において自身の資源属性である「PostgreSQL の IP アドレス」「PostgreSQL のポート番号」に設定されている情報を用いて、Datasource 設定を行い、Apache Tomcat の起動を行う。PostgreSQL の IP アドレス、ポート番号は動的に変化するため、予め設定しておくことはできず、起動している PostgreSQL 資源から取得する必要がある。そこで、まず依存関係の定義として「Tomcat 資源の起動の前には PostgreSQL 資源が起動されていること」を定義している。この定義によって Tomcat 資源を起動する際に、既に起動している PostgreSQL 資源があるかどうかの検索を行い、起動していなければまず PostgreSQL 資源の起動を行うことで依存関係の制約をクリアする。

次に Tomcat 資源を起動する前に、情報の授受の定義を参照し、PostgreSQL 資源の「IP アドレス」、「ポート番号」資源属性から情報を取得し、Tomcat 資源の「PostgreSQL の IP アドレス」「PostgreSQL のポート番号」に設定する。

「情報の授受の定義」で定義されている処理が完了した後に、Tomcat 資源の起動を行い、Datasource 設定を行う。このように接続定義によって計算リソース間の協調動作が可能となる。

4. 開発システムの効果

本研究において開発したプロビジョニングシステムの効果について検証する。

4.1. プロビジョニングの観点

これまでに、本研究において開発したプロビジョニングシステムにおいて、

IaaS では、

・仮想マシン

・ネットワーク（IP アドレス、VLAN）

を資源として扱い、仮想マシン資源とネットワーク資源間で適切な接続定義を定義することで、ネットワークを経由してユーザがアクセス可能な仮想マシン環境をプロビジョニングできることが確かめられている。

また、PaaS では、

・Apache Http Server

・Apache Tomcat

・PostgreSQL

を資源として扱い、Apache Http Server と Apache Tomcat 資源間、Apache Tomcat と PostgreSQL 資源間で適切な接続定義を定義することで、Web3 層アプリケーション実行環境をプロビジョニングできることが確かめられている。

上記の結果から、本研究で提案した資源という概念を用いて、従来は同一システム上で扱えなかった、サービスレイヤが異なる複数の計算リソースをプロビジョニング可能なプロビジョニングシステムが実現可能であることが示せた。

SaaS や HaaS の計算リソースは今回は検証対処から除外しているが、SaaS ではアプリケーションを、HaaS では物理サーバを資源とすることでプロビジョニング可能であると考える。

4.2. 資源の追加容易性

新規の計算リソースを資源として追加するには、各種定義ファイル（資源定義、接続定義）を記述し、資源ドライバを実装してプロビジョニングシステムに組み込むだけでよい。資源ドライバの組み込みは設定ファイルの記述だけでよく、プロビジョニングシステム

のコア部分の変更を必要としないため、動的に追加可能である。

また、資源の追加に要する時間については、これまでの実績から、操作対象に詳しくない担当者が操作対象の調査を行う時間を含めて実質 1 人月程度で済むことが分かっている。作成した資源ドライバは異常系設計が不足しており、商用品質を達成するためにはさらに多少の時間を要するものの、この結果は短期間に資源の追加が可能であること示している。このことから、本プロビジョニングシステムを用いることで、新規サービスを迅速に提供可能であると考える。

また、プライベートクラウドでは、各企業内でプロビジョニングシステムを構築することを考えると、ある企業では仮想マシン、別の企業では Linux コンテナを用いて IaaS を構築したいというニーズが発生する可能性がある。このような場合でもそれぞれに対応した資源をプロビジョニングシステムに追加することで容易に対応可能であることは本プロビジョニングシステムの強みである。

5. 関連研究

本研究において開発したプロビジョニングシステムは複数のサービスレイヤを横断したプロビジョニングを行うことを目的としており、特定サービスレイヤでの最適化を行っている既存のプロビジョニングシステムとは目的が異なる。

計算リソースを共通した IF で操作しようという試みは DeltaCloud [12]などで見られる。

DeltaCloud は AmazonEC2 や Eucalyptus [13]など複数の異なるクラウド管理システムが提供する IF を抽象化したメタ IF を提供する。制御対象を抽象化した IF を提供するという意味で本研究と発想が類似している。しかし、DeltaCloud が提供する操作は仮想マシンやストレージに特化しており、あくまでも IaaS という特定サービスレイヤの操作を抽象化する設計思想である。本研究のように異なるサービスレイヤを横断したプロビジョニングを行うことはできない。

6. おわりに

クラウドではサービスレイヤによってプロビジョニングの対象とする計算リソースが異なる。既存のプロビジョニングシステムは特定のサービスレイヤに最適化されているため、異なるサービスレイヤの計算リソースのプロビジョニングに適用できないという課題があることを示した。そこで、本研究では割当、起動、停止、解放という資源 SPI を定め、各計算リソースを資源 SPI を具備した資源として抽象化して扱うことで、異なるサービスレイヤの様々な計算リソースに対応するプロビジョニングシステムが実現可能であることを示した。現在、本プロビジョニングシステムを利用したクラウド運用管理基盤を構築中である。今後、実運用を通じて得られた知見に基づき、本システムの改良を行い、より実用性の高いプロビジョニングシステムを開発していく予定である。

7. 参考文献

1. **IDC Japan 株式会社.** 国内クラウドサービス市場動向を発表. (オンライン) 2011 年年.

2. The NIST Definition of Cloud Computing. **Peter MellGranceTimothy.** 出版地不明 : National Institu of Standards and Technology, Information Technology Laboratory, 2011 年.

3. amazon web services. (オンライン) http://aws.amazon.com/jp/ec2/.

4. google app engine. (オンライン) http://code.google.com/intl/ja/appengine/.

5. **The OpenStack Project.** Open source software for building private and public clouds. (オンライン) http://openstack.org/.

6. **Abiquo, Inc.** Virtualization, Cloud Services, Software - Cloud Computing Platform | Abiquo. (オンライン) http://www.abiquo.com/.

7. **VMware, Inc.** CLOUD FOUNDRY deploy & scale your applications in seconds. (オンライン) http://www.cloudfoundry.com/.

8. lxc Linux Containers. (オンライン) http://lxc.sourceforge.net/.

9. **The Apache Software Fundation.** The Apache HTTP Server Project. (オンライン) http://httpd.apache.org/.

10. **The Apache Software Foundation.** Apache Tomcat. (オンライン) http://tomcat.apache.org/.

11. **PostgreSQL Global Development Group.** PostgreSQL. (オンライン) http://www.postgresql.jp/.

12. **The Apache Software Foundation.** DeletaCloud | Many Clouds. One API. No problem. (オンライン) http://deltacloud.apache.org/.

13. **Eucalyptus System, Inc.** Cloud Computing Software from Eucalyptus. (オンライン) http://www.eucalyptus.com/.