情報処理学会論文誌 Vol.54 No.2 688–698 (Feb. 2013)

アカデミッククラウドアーキテクチャの提案と評価

横山 重俊[1,a)] 桑田 喜隆[2] 吉岡 信和[1]

受付日 2012年5月14日, 採録日 2012年11月2日

概要：クラウドコンピューティング技術は運用性，高信頼性，計算効率，柔軟性の高さにより，大学や研究機関など学術界でも活用が期待されている．本稿では教育および研究用途に特化したアカデミッククラウドのアーキテクチャについて提案する．従来技術と異なり，この研究の新規性である提案アーキテクチャの特徴は，研究グループや教育プログラムごとに，オンデマンドで物理マシンクラスタのセキュアに分離した形で構築し，その上にそれぞれの目的に合ったクラウド基盤（IaaS/PaaS/SaaS）を高速自動構築できるクラスタ提供をサービス化した Cluster as a Service（CaaS）と，地理的に分散するそれらクラスタから統一的に利用可能なインタークラウドストレージサービスを学認認証により統合し，今後の地域分散型のアカデミックコミュニティクラウド構築に活用することを考慮したことである．本アーキテクチャの適用により，地域分散型のアカデミックコミュニティクラウドがヘテロなクラウド基盤で構成された場合でも物理マシンレベルから動的にクラウド基盤を構成することで，たとえば，災害時には CaaS を用い，迅速に非常用クラスタを用意し，その上に通常時に使っているクラウド基盤を構築した後，インタークラウドストレージサービス内に保存している管理情報やマシンイメージをリストアすることで教育および研究活動の継続が可能となる．また，平常時においても，不足しているコンピュートリソースを CaaS によりオンデマンドで既存のクラスタに組み込むことができる．このことで，アカデミックコミュニティ全体でのリソース使用効率の向上に寄与する．

キーワード：クラウドコンピューティング，教育クラウド，研究クラウド，アカデミッククラウド，コミュニティクラウド

A Proposal of Academic Community Cloud Architecture and Its Evaluation

Shigetoshi Yokoyama[1,a)] Yoshitaka Kuwata[2] Nobukazu Yoshioka[1]

Received: May 14, 2012, Accepted: November 2, 2012

Abstract: This paper describes a proposal of academic community cloud architecture. The uniqueness of this architecture is that it can provision bare metal machine clusters on-demand and deploy heterogeneous cloud platforms on them. It scopes inter-cloud storage service by which users can retrieve machine images for the cloud platforms as if they come from local cloud storage.

Keywords: cloud computing, education cloud, research cloud, academic cloud, community cloud

1. はじめに

近年，クラウドコンピューティングは，運用性，高信頼性，計算効率，柔軟性の高さにより，大学や研究機関など学術界でも活用が期待されている．たとえば，あらかじめ教育や研究の標準的な環境をクラウド基盤サービス上に構築しておくことにより，必要なときに必要な環境が迅速に準備でき，その運用性を高めることができる．また，研究の実験環境として構築されたマシン環境を，スナップショットとして保存することにより，環境の再利用性を高め，研究の効率化が望める．

[1] 国立情報学研究所
National Institute of Informatics, Chiyoda, Tokyo 101–8430, Japan
[2] 株式会社 NTT DATA
NTT DATA Corporation, Koto, Tokyo 135–6033, Japan
[a)] yoko@nii.ac.jp

そこで，筆者らは，全国の大学・研究機関に対して，教育向けのクラウド（edubase Cloud）を提供している [1]．

edubase Cloud は，主に教育用途に特化して設計されており，講義や演習で活用できるように，改変性，専有性，連携性，保存性などの特徴を具備している．さらに，現在，この運用経験をふまえ，研究向けクラウド環境を構築している．その中で，研究用途のクラウド環境は，教育用クラウドでは顕在化しなかった性能面や特殊リソースの活用に関する課題が明らかになった．たとえば，一般にクラウドの基盤サービスは，運用性を向上するために仮想化されたマシンで提供される．しかしながら，仮想化されたマシンは仮想化のオーバヘッドや性能の揺らぎがあるため，マシンの性能を極限まで活用する研究や性能評価環境には向かない．また，研究によっては大容量メモリや特殊なプロセッサなどを活用するため，仮想化できないリソースが存在する．そのため，研究用のクラウド環境では，これら特殊なリソースの利用を考慮する必要がある．

さらには，大学や研究機関で個別に構築されているアカデミッククラウドが連携し，アカデミックコミュニティクラウドとして機能する仕組みが求められてきている．筆者らは自らのアカデミッククラウドを設計するにあたり，教育向けクラウドの運用経験と研究向けクラウドの要求条件を合わせて教育・研究向けアカデミッククラウドのアーキテクチャ設計で解決すべき課題を明確にし，それを解決すると同時に将来的なクラウド連携を視野に入れたアカデミッククラウドのアーキテクチャ設計を行った．従来技術と異なり，この研究の新規性である提案アーキテクチャの特徴は，クラスタ構築サービスとそれで構築されたクラスタから広域網を介して利用できるインタークラウドストレージサービスを学認認証により統合したことである．クラスタ構築サービスを利用して，研究グループや教育プログラムごとに，オンデマンドで物理マシンクラスタをセキュアに分離した形で構築し，その上にそれぞれの目的に合ったクラウド基盤を高速自動構築できる．また，地理的に分散するそれらクラスタから統一的に利用可能なインタークラウドストレージサービスにクラウド基盤で必要になる研究データやマシンイメージなどを保存することで，動的にクラウド基盤を構築してもその構築場所への制約が弱まる．

筆者らは，この特徴が今後の地域分散型のアカデミックコミュニティクラウド構築に役立つと考えている．実際，オンデマンドで物理マシンから動的にクラウド基盤までを構築するアーキテクチャとすることで，各大学や研究機関で個別に構築されている各種アカデミッククラウドの拡張部分を必要に応じて伸縮させることができ，共用リソースを柔軟に利用することができる．従来の様々な種類のクラウド基盤を結ぶフェデレーション用標準仕様を決めるというアプローチと異なり，リソース提供側とリソース利用側の物理マシンの利用方法に関する取り決めが個別にできれば連携が逐次実施可能となる．

本アーキテクチャの適用により，地域分散型のアカデミックコミュニティクラウドがヘテロなクラウド基盤で構成された場合でも物理マシンレベルから動的にクラウド基盤を構成することで，たとえば，災害時には CaaS を用い，迅速に非常用クラスタを用意し，その上に通常時に使っているクラウド基盤を構築した後，インタークラウドストレージサービス内に保存している管理情報やマシンイメージをリストアすることで教育および研究活動の継続が可能となる．また，平常時においても，不足しているコンピュートリソースを CaaS によりオンデマンドで既存のクラスタに組み込むことができる．このことで，アカデミックコミュニティ全体でのリソース使用効率の向上に寄与する．

本稿では，2 章で筆者らの提案の出発点である教育クラウド edubase Cloud について述べ，3 章で教育向けクラウドの運用経験と研究向けクラウドの要求条件を合わせて教育・研究向けアカデミッククラウドのアーキテクチャ設計で解決すべき課題を要求条件としてまとめ，4 章でそれを解決すべく設計したアーキテクチャについて提案する．5 章では，提案アーキテクチャを構成するサービスコンポーネント群の実装と評価について述べ，将来的なアカデミッククラウド構築に向け，提案アーキテクチャが有効であることを示す．6 章で類似事例について述べ，7 章で本稿をまとめる．

2. 教育クラウド edubase Cloud について

ソフトウェア技術者の育成を目指し実践力強化につながる教育プログラムとしてプロジェクトベースラーニング（以下 PBL と呼ぶ）を支援する IT システムとして教育クラウド edubase Cloud を構築し，運用している．PBL に必要な要素として，IT 環境自身も実社会の IT 環境に近いものとする必要がある．しかし，各大学などで PC サーバのプールを保有し，それらをプロジェクトに配布する，あるいは新規に配備する，という対応になっていることでそのような IT 環境を構築する手間やコストが大きく，実践的な教育向けに現実世界に近い構成の IT 基盤の迅速な準備が難しいという課題が上がっていた．edubase Cloud の構築はこの課題に対応したものである．

筆者らは，オープンソースで提供されるクラウド基盤を活用し，幅広い範囲の IT 技術者を対象とした PBL 用教育クラウドを提供可能であると考えた．クラウドを対象とした IT 基盤技術者の育成もできる環境を構築する目標を置き以下の環境が必要であると結論付けた．

- クラウド基盤の仕組みを実習で確認し，変更することができる．（要件 1）
- 実習が他の利用者の環境に悪影響を及ぼしてはならな

い．（要件 2）

• ハイブリッドクラウドやクラウド間連携を教えるために，既存のクラウドとの連携ができなければならない．（要件 3）

• 試してみた環境を保存し，後で利用できる必要がある．

これらの条件を満たし，IT 技術者の育成用に用いる教育クラウドとするためには，通常のクラウドの特性のほかに，次の要件を満たす必要がある．（要件 4）

(1) 専有性

クラウド基盤を改変する際，性能測定を実施する際などの他のプロジェクトとの干渉が出ない．

(2) 改変性

クラウド基盤そのものを改変できる．

(3) 連携性

パブリッククラウドや他のプライベートクラウドとのマシンイメージ移行やアプリケーションの移植が容易である．

(4) 保存性

教育プロジェクトで作成された成果物（マシンイメージなど）格納するアーカイブ機能を持つ．

要件 1 は専有性と改変性により実現できる．要件 2 は専有性により実現できる．要件 3 は連携性により満たされる．要件 4 は保存性により達成できる．これらの性質を満たすクラウドシステムを edubase Cloud 設計構築したのでそのアーキテクチャとシステム構成について以下に説明する．

2.1 edubase Cloud のアーキテクチャ

上記性質を実現するために採用したアーキテクチャを順に説明する．

(1) 複数のクラウドから構成されたマルチクラウドシステム（専有性と改変性の実現）

クラウドの専有性を実現するため，クラウドを複数のミニクラウドの集まりとするマルチクラウド構成とする．1 つのミニクラウドをプロジェクトで専有することができる．

また，クラウド返却時に元の基盤部分ソフトウェアを貸し出し前の状態に書き戻す初期化機能を実現することにより，貸し出されたプロジェクトによってクラウド基盤自身の改変を許容する．

(2) オープンソースソフトによる基盤構築（改変性と連携性の実現）

クラウドの基盤ソフトを変更するためには，基盤のソースコードにアクセスし，改変することが必要である．このためソースコードが公開されており，改変も許されたライセンスを持つオープンソースソフトを基盤ソフトとして採用する．具体的にはパブリッククラウドとの連携性を考慮した Eucalyptus を改変することで edubase Cloud の基盤を構築した．

(3) プロジェクト成果物などの保存を行うためのアーカイブシステム（保存性の実現）

プロジェクト成果物を格納し，効率良く検索可能なアーカイブシステムを持つ．アーカイブに格納されるプロジェクト成果物としては，ドキュメントのほかにクラウド上で作成したプログラムや仮想マシンイメージなどが想定される．

(4) プロジェクト内のコミュニケーション支援機能

PBL を円滑に進めるため，プロジェクトを効率良く進めるために，電子会議室，メーリングリストなどのコミュニケーション支援機能を持つ．さらに，この支援機能を提供するサービスとクラウドをシングルサインオンできるようにし，サービス連携をスムーズに実施できる．

(5) 運用監視機能

数多くの物理マシンからなる基盤の運用を容易にするため，統合的な運用監視機能を持つ．

2.2 edubase Cloud システム構成

edubase Cloud は約 200 台の計算用サーバ，共用サーバ，監視系サーバおよびストレージからから構成されるクラウドシステムである．**図 1** に edubase Cloud の全体構成を示す．

(1) マルチクラウド

15 組のミニクラウド群から構成されるマルチクラウドシステムの構成をとることとした．ミニクラウドは 2 種類の構成をとり，用途に応じて使い分けられるようにした．**表 1** にマルチクラウドの構成を示す．

各ミニクラウドはオープンソースソフトウェア（OSS）である Eucalyptus に対し機能拡張と品質改善し構築した．構成 1 のミニクラウドは 16 台の計算機から構成される．

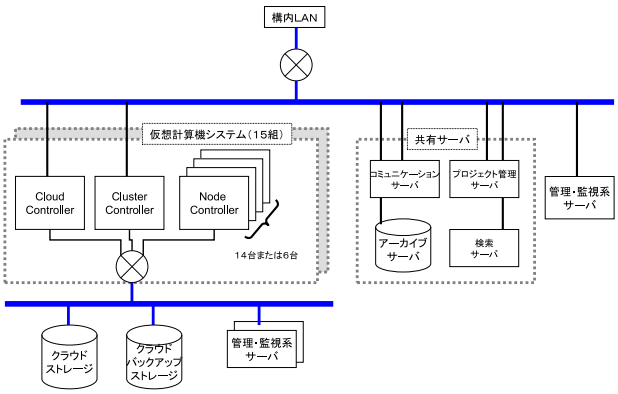


図 1 edubase Cloud の全体構成
Fig. 1 edubase Cloud system configuration.

表 1 マルチクラウドの構成概要
Table 1 Multi-cloud configuration.

構成	名称	数	計算機台数	最大仮想マシン数
構成 1	ミニクラウド（大）	10	16	112
構成 2	ミニクラウド（小）	5	8	48

Eucalyptus のアーキテクチャに従って，仮想マシンの割当てを司るクラウドコントローラ（CLC）および通信の制御を行うクラスタコントローラ（CC）を各 1 台割り当て，残りの 14 台を実際に計算に利用するノードコントローラ（NC）に割り当てた．NC は 4 コアの CPU を 2 機搭載している．1 コアあたり 1 仮想マシンを起動することができる設定としたため，1 個の NC あたり，8 個の仮想マシンを利用することが可能である．

したがって，構成 1 のミニクラウドでは最大 112 個の仮想マシンを起動することができる．構成 2 のミニクラウドは 8 台の計算機から構成される．CLC，CC に関しては構成 1 と同様であるが，構成 2 では利用可能な NC が 6 台に減るため起動可能な仮想マシンも最大 48 個となる．

(2) クラウド用ストレージ

クラウド用のストレージとしてネットワーク経由でアクセス可能な iSCSI ストレージを複数台用意した．RAID 構成時の実効総容量は約 40 TB である．

(3) 共用サーバ群

PBL での利用に適用するため共用サーバ内のプロジェクトに対応したプロジェクトという単位をクラウド基盤側で持てるような連携機能を実装している．

- プロジェクト情報管理サーバ

アーカイブ機能などのプロジェクトの管理機能を提供する．OSS である Drupal を利用して実現した．

- ユーザ管理サーバ

クラウドや共用サーバで利用する利用者情報を管理するサービスを提供する．LDAP によって一元管理する仕組みを導入した．

- コミュニケーションサーバ

プロジェクトを進めるにあたっての情報ポータルを提供する．OSS である Moodle を使って実現した．

(4) 監視システム

監視システムも教材となる可能性があることから，クラウドごと監視サーバを導入する構成とした．OSS である Hinemos を使って実現した．

3. アカデミッククラウドへの要求条件

前章までで述べた教育クラウド edubase Cloud は，2010 年 5 月より，主に大学などでの講義や演習および PBL で活用されている．その延長として学生研究での利用も始まってきている．この運用を通じて当初想定していた以上の要求が出てきており，これらの要求への対応が必要になっている．さらには，本格的な研究活動を支えるための研究向けクラウドの構築も求められている．クラウド運用の効率化をはかるため，これら教育向けクラウドおよび研究向けクラウドを合わせたアカデミッククラウドとして拡張することとした．この際に求められる要求条件について以下に整理する．

3.1 教育クラウドの運用を通じて得られた要求条件

(1-1) ミニクラウドの動的サイズ変更

Web サービス構築に関する講義で負荷試験に対するグループ演習を実施する際に，各グループがそれぞれ開発中の Web サービスを構成する仮想マシンに加えて負荷生成のための仮想マシン，それと負荷に合わせてスケールアウトした際の仮想マシンなどを必要とした．この際，グループ数も大きかったため全グループに必要な仮想マシンを供給するためには，ミニクラウド（大）1 つでは容量が不足した．このため 3 ミニクラウドを利用する設定で実施した．利用者の設定や事前マシンイメージの登録など運用上の工数がその分増加した．また，学生研究で Hadoop の改造を実施した際の性能評価を行う際にもミニクラウド 1 つのサイズを超えたノード数での実験を希望された．この際にはパブリッククラウドへ実験環境を移行してもらうなどの措置をとらざるをえなかった．

これは現状の教育クラウドの持つ専有性と，上記柔軟性への要求を両立させるアーキテクチャになっていないことが原因であるけれど，オンデマンドでのクラウドサイズの変更ができ，この両条件を同時に満たすことで教育クラウドの適用範囲の拡大と運用の省力化がはかれる．

(1-2) クラウド初期化ツールの汎用化

教育クラウドの 1 つの特徴である専有性と改変性をいかして，たとえばクラウド基盤自身の改変を実施するクラウド基盤構築演習の実施を行った．この際，改変を実施後は通常のクラウド基盤設定に戻す必要があるためミニクラウドのクラウド運用者向け初期化ツールを開発し，利用した．また，クラウド基盤の HPC 利用に関する学生研究としてクラウド基盤内の Xen のコードを改変して性能測定をする必要があるケースにも同様の対応を実施した．いずれのケースも運用者側への稼働が発生し，ひいては専有性や改変性を活用した教育クラウド利用に対して，利用者側の利便性が十分確保できたとはいえない状況であった．この経験からクラウド初期化あるいはクラウド構築ツールの汎用化や使い勝手の向上をはかり利用者自らが利用できるアーキテクチャを導入する必要があることが分かった．

(1-3) アーカイブ機能利用シーン拡大

教育クラウドのアーカイブ機能を構築した当初の目的は PBL などの成果を他のプロジェクトでも活用できるよう蓄積することであった．このため，厳選した成果物の置き場として管理する趣旨で，クラウド利用者は読み出せるけれど，書き込みについては申請制にし，クラウド運用者のみが実施できる仕様とした．

実際の運用では複数ミニクラウド間でマシンイメージの共有をしたい場合やミニクラウド間の仮想マシンの引っ越しなどの際のミニクラウド間共有ストレージとして利用されるシーンも多数存在した．この際に申請制でクラウド運用者がそのつど関わるオーバヘッドが問題となった．した

がって，当初の目的以外のミニクラウド間共有ストレージとは別にクラウド利用者がクラウド間の共有ストレージとして活用できるものが必要であることが分かった．

(1-4) ユーザ管理の外部サービス化

クラウド運用にあたっては監視・障害対応以外に利用者からの申請や問合せ対応が大きな稼働を占めると想定していたけれど，特にユーザ登録申請への対応が予想以上の負担となった．このため，学認のような外部認証サービスとの連携により，ユーザ管理の負担軽減をはかる必要がある．

(1-5) クラウド間連携

アカデミッククラウドに関して各地に様々な用途や実装で構築が進んでいる．それぞれの独立性を重視しつつ必要に応じてリソースの融通をすることや，教育・研究データやマシンイメージの共用をはかるためのクラウド間連携をはかろうという要求が顕在化している．

(1-6) 災害対応

今回の震災後の関東圏における電力不足により教育クラウドを 1 カ月余り停止する事象が発生した．この際，他の地域のクラウドやパブリッククラウドにマシンイメージや各種データを移行することで復旧を早める必要があった．

3.2 研究クラウド構築にあたってヒアリングを通じて得られた要求条件

筆者らの所属する研究所の研究者向けのクラウドを構築するため研究グループ側と研究管理側へのヒアリングを実施し，その結果を研究クラウドへの要求条件としてまとめた．研究グループのほとんどは自グループ専用のサーバクラスタを保有しており，それと同等以上の専有性，性能と容量を研究クラウドに求めると同時にクラウドであることによる使い勝手の良さを求めている．一方，研究管理側からは，研究グループ個別のサーバクラスタを構築，運用することによる分割損の研究クラウドによる解消を求めている．

(2-1) 高速分散処理可能な大容量クラスタの動的構築

常設でなくてもよいけれど大規模実験を実施してデータ収集を行うため，ある一定期間，大規模メモリ，大規模コア数，大規模ストレージをそれぞれの研究に合わせたバランスで持つ大容量のクラスタがオンデマンドで必要となる．さらには最終的なデータ収集のためには仮想化によるオーバヘッドや揺らぎのない環境が必要である．

(2-2) 容易な構築/運用

各研究グループに IT 基盤に詳しいメンバが必ずしも在籍しているとは限らないし，そのつど外部業者に委託するのも時間や費用の制約から難しいため，クラスタを構築する際にはその構築要素など IT 技術に精通していなくても容易に構築できる必要がある．

(2-3) 独立性の高い評価環境

構築したクラスタ上で性能評価する際には他の利用者が発生させる負荷などによる揺らぎが許されないためクラウドいえども単独に構築されたクラスタと同様の独立性が要求される．

(2-4) コンソリデーションによるコスト削減

一方，研究管理を実施している立場からは可能な限りリソースの共用化をはかり IT 投資の削減が求められる．

(2-5) 運用省力化/集中化によるコスト削減

構築後の IT 環境の運用についても各研究グループでの実施による分割損を減らすための運用の集中化とその省力化が求められる．

4. アカデミッククラウドアーキテクチャ提案

前章で述べたアカデミッククラウドへの要求条件を整理すると以下のようになる．この要求条件抽出は，前章で述べたように筆者らが運用する教育クラウドと構築する研究クラウドからのものである．しかしながら，以下のような教育クラウドや研究クラウドを構築する際の要求条件となる汎用性を持っていると筆者らは考えている．

4.1 対象範囲

(1) 教育クラウドの対象教育範囲

講義，演習，PBL で一定期間クラウドリソースを専有する教育プログラムを対象範囲とする．

(2) 研究クラウドの対象研究範囲

一定期間サーバクラスタを専有して，大規模並列計算を実施する研究を対象範囲とする．

4.2 要求条件

3.1 節，3.2 節で述べた要求条件を合わせて，4.1 節の対象範囲で汎化し，筆者らの目指すアカデミッククラウドへの要求条件と設定した．

要求条件 1：クラウドへのリソース割り付けの柔軟性

(1-1) ミニクラウドの動的サイズ変更，(2-1) 高速分散処理可能な大容量クラスタの動的構築，(2-4) コンソリデーションによるコスト削減，より導かれる要求条件

要求条件 2：ベアメタルな性能

(2-1) 高速分散処理可能な大容量クラスタの動的構築，より導かれる要求条件

要求条件 3：クラウド間の分離

(1-1) ミニクラウドの動的サイズ変更，(2-3) 独立性の高い評価環境，より導かれる要求条件

要求条件 4：クラウド構築/運用の簡単化

(1-2) クラウド初期化ツールの汎用化，(2-2) 容易な構築/運用，(2-5) 運用省力化/集中化によるコスト削減，より導かれる要求条件

要求条件 5：クラウド間連携

(1-3) アーカイブ機能利用シーン拡大，(1-5) クラウド間連携，(1-6) 災害対応，より導かれる要求条件

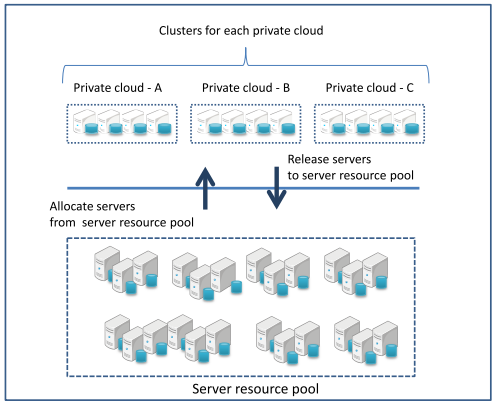


図 2 Elastic private cloud
Fig. 2 Elastic private cloud.

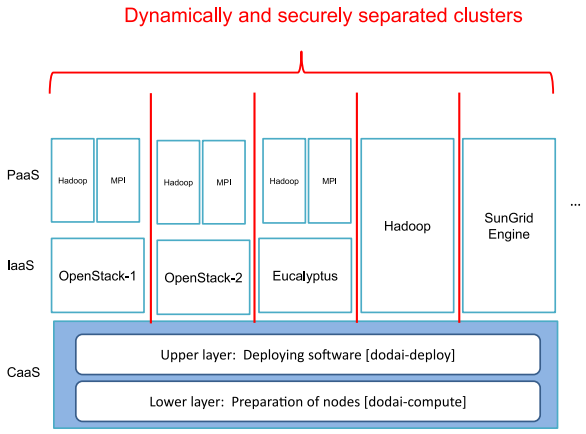


図 3 CaaS の位置づけ
Fig. 3 CaaS layer.

要求条件 6：外部認証の利用

(1-4) ユーザ管理の外部サービス化，より導かれる要求条件

これらの要求条件 1，2，3，4 を満たすアカデミッククラウドアーキテクチャとして，筆者らは物理マシンで構成されるプライベートクラウドを動的に構築できる Elastic Private Cloud（**図 2** 参照）を Cluster as a Service により実現できるコンピュート部分と要求条件 5 を満たすために必要な各プライベートクラウドから共通に利用できるインタークラウドオブジェクトストアサービスから構成されるものを考案した．そして要求条件 6 への対応として，これらを構成するサービス群を学認認証 [2] により利用できるアーキテクチャを採用した．

4.3 Cluster as a Service

ベアメタルマシンもあたかも仮想マシンのように API 経由で制御し，さらにはネットワーク機器も同様に制御することで複数のベアマシンをクラスタとして 1 つの独立したプライベートクラウドとして自動構築するサービスを，ここでは Cluster as a Service（以下 CaaS）と呼ぶ [3]．CaaS により Eucalyptus や OpenStack などで構成される IaaS を構築できることから CaaS を IaaS よりも下層レイヤに位置付けることができる．同様に Hadoop クラスタなどの PaaS も CaaS の上層として構築することができる．これらの関係を**図 3** に示す．

CaaS 層の 1 つの実装としては図 3 に示すように現在上下 2 層から構成されているクラスタ管理フレームワーク dodai [4] を筆者らは採用している．Dodai は，物理マシンで構成されるクラスタを管理する dodai-compute と，物理，および仮想マシン上のソフトウェア環境を管理する dodai-deploy で構成され，dodai-compute は仮想マシンと同様の API を提供している．

(1) Dodai-compute

各クラスタ環境は，dodai-compute から生成される 1 つの仮想閉域ネットワーク上に構築されるため，他のクラスタ利用者と排他的なアクセス制御が可能である．

本フレームワークを用いることで，dodai-compute や EC2 互換の API で設定された物理，および仮想マシンのクラスタ環境に対して，研究に必要な OS・ソフトウェア環境を dodai-deploy により構築し，柔軟性の高い研究環境を動的に提供することができる．以下が dodai-compute の主な機能である．

リソースプール管理機能：

サービスとして提供可能なマシンやストレージを管理する．クラスタを構築するために必要なリソースは本プールから選択され，利用中の状態として管理される．不要となったリソースは，初期化後，再利用可能なリソースとなる．そして必要に応じて電源が落とされる．

マシンイメージ管理機能：

物理マシンを起動する際に読み込むソフトウェア環境のイメージを管理する．必要なマシンイメージの登録，削除機能も具備する．

クラスタ構築機能：

マシンイメージの ID やマシンの必要台数を指定し，それらで構成されるクラスタを構築する．これらの機能は，仮想マシン，物理マシンの双方を統一的に扱えるようにするために，Amazon EC2 に準じた API でも呼び出せるようになっている．ただし，EC2 には，クラスタを構築する機能がないため，現状，ゾーン ID をクラスタ ID と見なし，クラスタへのマシンの追加はゾーンを指定したマシンの起動に対応させている．したがって，dodai-deploy は**図 4** に示すように通常の IaaS に対しても可能であり，たとえば研究フェーズが初期の場合には仮想マシンを活用してアルゴリズムの開発/検証行い，そのフェーズが終了し，ベアメタルでの性能評価などが必要になった段階で，研究環境を仮想環境から物理環境へスムーズに移行できるというメリットがある．

図 5 に dodai-compute のアーキテクチャを示す．物理マシンの制御には IPMI および cobbler 経由で PXEBOOT

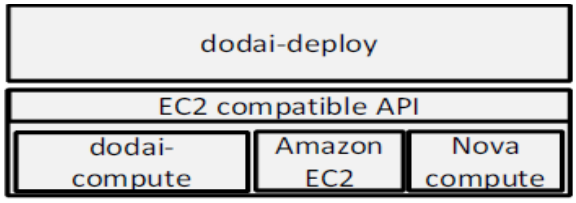


図 4 Dodai-deploy 活用方法
Fig. 4 Dodai-deploy usage.

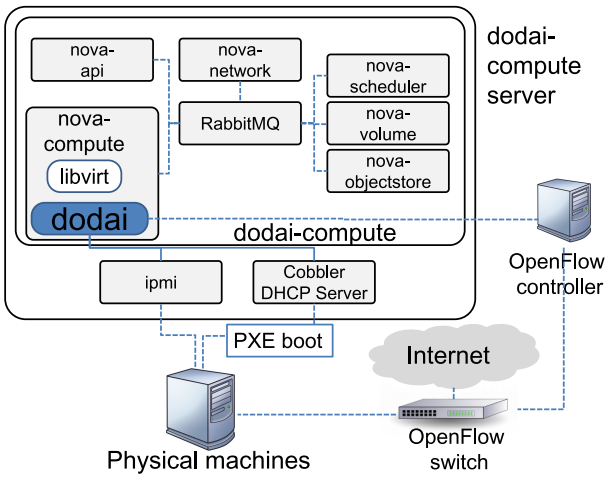


図 5 Dodai-compute のアーキテクチャ
Fig. 5 Dodai-compute architecture.

の仕組みを利用し，同時に動的なネットワークセグメント構築のために OpenFlow [5] コントローラとスウィッチを利用する．Dodai-compute によりベアメタルクラスタを仮想閉域ネットワーク上にソフトウェアコントロールにより動的に構築できるため，要求条件 1，2，3 を同時に満たすことができる．

(2) Dodai-deploy

以下が dodai-deploy の主な機能である．

ソフトウェアのインストール：

Dodai-compute で構築されたクラスタ中の複数マシンに対して，複数のソフトウェアをインストールする．これにより，Hadoop や OpenStack 環境などをクラスタ上で利用可能になる．

インストール状況の確認のためのテスト：

上記のインストールで構築されたソフトウェア環境が，正しく動作するかどうかのテストを登録し，それを実行することができる．

インストール計画（Proposal）の設定：

クラスタ中のどのマシンに対してどのソフトウェアを，どういった設定で構築するかを Proposal と呼ぶ設定ファイルによって宣言できる．目的に応じてクラスタ環境の Proposal を定義することができる．

Dodai-depoly の基本的な動作手順は次のとおりである．Deploy サーバは，インストールの要求があると，ソフトウェア自動設定ツール（Puppet [6]）用の設定ファイルを

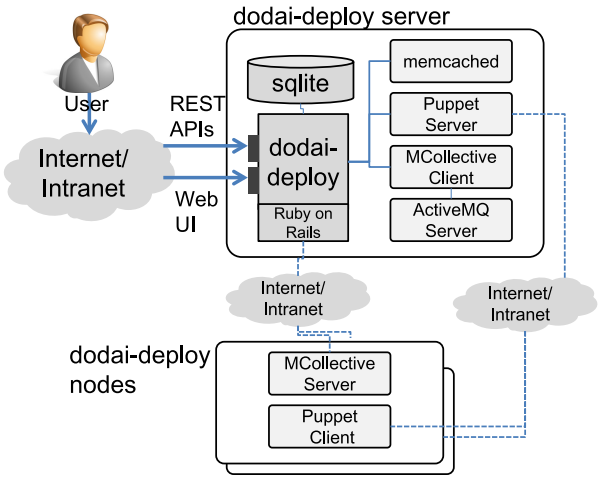


図 6 Dodai-deploy のアーキテクチャ
Fig. 6 Dodai-deploy architecture.

表 2 既存クラウドアーキテクチャと Elastic Private Cloud の比較
Table 2 Cloud architecture comparison.

	リソース割り付けの柔軟性	ベアメタルな性能	クラウド間分離	クラウド構築/運用の簡単化
Public Cloud	○	×	×	n/a クラウドとして提供されているため構築が不要
Private Cloud	× Private Cloud自体のリソースは固定的	×	○	× 個別構築が必要
Hybrid Cloud	○	×	△ Public Cloudとの接続方式によっては分離可能	× Private Cloud側の構築およびPublic Cloudとの接続が必要
Elastic Private Cloud	○	○	○	○

生成し，指定されたマシンにそれを送り込む．本方式では，多数のソフトウェアを並行インストールし，環境構築時間を短縮するために，オペレーションを並行実行するフレームワーク（MCollective [7]）を経由してインストールの指示を各マシンに出している．**図 6** に dodai-deploy のアーキテクチャを示す．Dodai-depoy を利用する際には GUI に加えてソフトウェアから制御できる API も持つ．Dodai-compute で確保したベアメタルクラスタの上に dodai-deploy で各種クラウド基盤がオンデマンドで構築できるので要求条件 4 を満たすことができる．

本節で述べた CaaS を使って Elastic Private Cloud を実現するメリットを既存のクラウドアーキテクチャとの比較として**表 2** に示した．すなわち，Elastic Private Cloud により今まで相反する要求条件であったリソース割付けの柔軟性によるリソース共有化，クラスタ分離によるセキュリティ確保，そしてクラウド自体をサービス提供されることによる利便性の 3 つを同時に満足させることができるようになった．それに加えて直接ベアメタルクラスタとして利用することもできるため，性能的な優位性も確保できる．

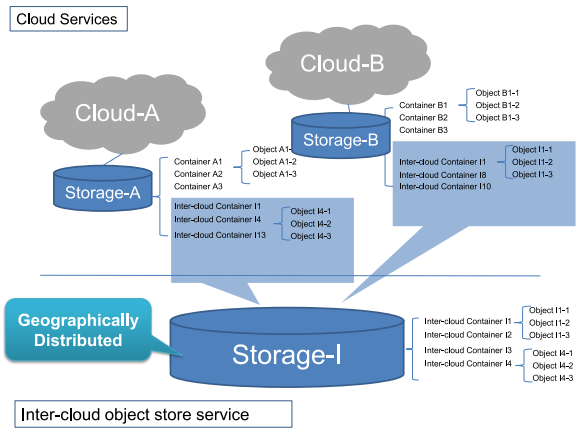


図 7 インタークラウドオブジェクトストアサービス
Fig. 7 Inter-cloud object store service.

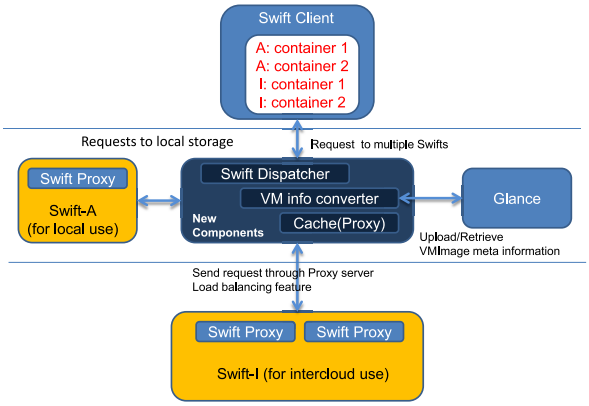


図 8 Colony アーキテクチャ
Fig. 8 Colony architecture.

つまり Elastic Private Cloud が，筆者らが解決しなければならないアカデミッククラウドに対する要求条件 1，2，3，4 を同時に解決するソリューションであることを示している．

4.4 インタークラウドオブジェクトストアサービス

各地域に存在するクラウドでローカルに存在するオブジェクトストアサービスとシームレスにあたかも同一オブジェクトストアのように見えるインタークラウドオブジェクトストアサービス（**図 7** 参照）を持つことで教育クラウドのアーカイブ機能を拡張したクラウド間連携（要求条件 5）に役立つストレージサービスを提供することができる．

インタークラウドオブジェクトストアサービスの実装として筆者らは OpenStack [8] オブジェクトストアサービス swift を広域分散向けに改変した colony を採用している．Colony は**図 8** に示すように swift に 3 つの追加モジュールを提供することと swift 内部のレプリケーション部分の広域分散対応部分からなる．

4.5 学認連携

認証基盤学認は Shibboleth 準拠であるため dodai のフロントエンドサービスと colony のフロントエンドサービスをそれぞれ Shibboleth 対応し，初期アクセス時に学認 ID と申請時にローカル側に仮登録した ID を紐づけることで学認連携を実現でき，要求条件 6 への対応している．

以上のアーキテクチャと実装方針に基づきアカデミッククラウドを実装できる見通しができた．実装の前に，それぞれの構成サービスコンポーネントについて評価を実施したので，その評価結果について次章に述べる．

5. サービスコンポーネント評価

提案アーキテクチャが要求条件 1–6 を満足できるものであること示すために，そのサービスコンポーネント例である dodai，colony と学認連携について各要求条件への寄与について評価した．

5.1 Dodai 評価

要求条件 1，2，3 については dodai-compute が要求条件 4 については dodai-deploy が寄与することを想定している．

(1) Dodai-compute の評価

リソースの割付けの柔軟性を実現するためには，割付け，開放がソフトウェアから簡単に行え，かつその速度が実用的であることが重要である．また，構築されたクラスタでベアメタル性能が出せることとセキュアにネットワーク分離できることも確認する必要がある．

Dodai-compute のインタフェース run-instance で物理マシンを立ち上げ，それをクラスタに組み込み，物理マシンクラスタを複数個構築できた．さらに，それらがネットワーク的に分離できることを確認した．この際，その割付け，開放性能も測定した．Run-instance についてはディフォルトマシンイメージの際には，あらかじめそのマシンイメージで物理マシンを立ち上げ物理マシンプール内で待機しているために run-instance 自身はネットワークセグメントを切り替える 3 秒程度で終わるし，実際に OS が立ち上がっているためすぐに利用可能な状態である．この評価実験を通じて run-instance により動的にしかも素早くベアメタルマシンの立ち上げを実施でき，仮想閉域ネットワークによりクラスタ間の分離ができていることを確認した．このことにより，dodai-compute が “要求条件 1：クラウドへのリソース割付けの柔軟性”，“要求条件 2：ベアメタルな性能” および “要求条件 3：クラウド間の分離” の実現を可能としていることが検証できた．

ただし，ディフォルト以外のマシンイメージで立ち上げる場合には PXE ブートからマシンの立ち上げを行うためこれに要する時間が 10 分程度かかってしまうことも分かった．こちらについては kexec の活用などによる性能

向上の取り組みがさらに必要である．Terminate-instanceについては現状の実装では返却後の物理マシン内に外部に出てはならない情報が残らないように物理マシンプールに戻す前にストレージをゼロクリアする処理を走らせる．ストレージサイズによるが，4 TB 程度のストレージの場合でも 10 時間程度この処理に要している．この処理は利用者見えないバックヤードで非同期に実施されるためterminate-instance をする利用者は影響されないものの，運用の観点から，改善が必要である．

(2) Dodai-deploy の評価

クラウドの構築/運用の簡単化が dodai-deploy で実現できることを確認するため，初見の利用者でも簡単な操作でクラウド構築ができることを検証した．また，クラスタサイズが大きくなった場合でも deploy 速度が実用的である必要があるため，dodai-deploy の並行実行に関する性能を測定した．

Dodai-deploy を使い勝手について評価するために一般受講者を対象とした OpenStack on edubase Cloud ハンズオンセミナを用いて OpenStack diablo を自動インストールする際の試験を実施した．この際，14 名の受講生全員が講師のサポートなしに各自で OpenStack 用クラスタの構築に成功した．また，クラスタ構築要した時間は実際のdeployment 時間である 8 分程度でありハンズオンセミナ内での利用には適用可能な時間であった [9]．

また，クラスタを構築する際の性能評価を OpenStack nova と Hadoop について n ノードで構成される deploy 時間を測定することで実施した．OpenStack nova については nova compute を deploy するノード数を増加させた．Hadoop については data_node と task_tracker を deploy するノード数を増加させた．この結果，deploy に要する時間は，dodai-deploy が持つあらかじめ指定されたソフトウェアコンポーネントの依存関係を考慮しつつ可能な限り並列deploy 実行を指示する機構の効果で，**図 9** に示すようにn に依存せずほぼ一定であることが分かった．ただし，この評価環境（edubase Cloud 上の仮想マシンを利用）ではHadoop の場合（n = 7, 8）で deploy 時間が 8%程度増加している．これは puppet プロセスが deploy サーバの cpuを使いきる状態になったためであり，今後サーバ側の処理分散などの改善が必要であることが分かった．これらのことより dodai-deploy が “要求条件 4：クラウド構築の自動化” 実現に寄与するための問題点が明らかになった．

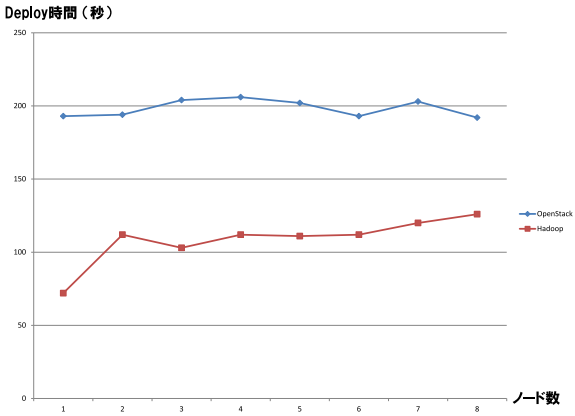


図 9 Dodai-deploy の性能評価結果
Fig. 9 Dodai-deploy deployment performance.

5.2 Colony の評価

要求条件 5 については colony [10] が寄与することを想定している．

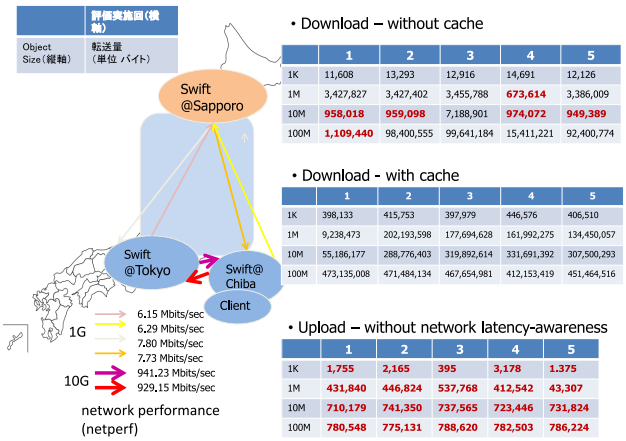
クラウド連携を今後広域網経由で地理的に分散するプライベートクラウド間で実現する際必要なデータ共有機能であるインタークラウドオブジェクトストアサービスの性能評価を行った．たとえば，各プライベートクラウドからマシンのスナップショットをこのインタークラウドオブジェクトストアサービスへ蓄積する場合（upload）もそれを別のプライベートクラウドから利用する場合（download）もあたかも各プライベートクラウド内に存在するオブジェクトストアサービスへの upload および download スピードにすることが利用性向上の観点から重要になる．筆者らがインタークラウドオブジェクトストアサービスの実装に選択した colony の評価として，OpenStack swift とそれを広域分散に向くように改変した colony を広域分散環境で比較し，その改変の効果を測定した．具体的には図 9 に示すように，東京，千葉，札幌に配置した swift クラスタ間をSINET-4 L2VPN 機能を用いて接続し実験を行った．測定の結果オブジェクトのダウンロードに関してはキャッシュの効果が出ることが確認できた．さらに，キャッシュされていない場合でもネットワーク遅延が最小であるクラスタからのダウンロードを優先する効果も出ている．

また，アップロードについてはオリジナルの swift ではレプリケーションする際にいつもネットワーク遅延の最大のものへの書き込みの完了まで同期的に待つために性能がその遅延の大きいクラスタによる．**図 10** の Upload - with network latency-awareness の表に示すように，Colony の実装ではネットワーク遅延を意識し，それが最小のクラスタにレプリケーションを一時的に書き込んだ後，非同期に他のクラスタに複製をする．この仕組みにしたために性能が Upload - without network latency-awareness の結果と比較して 1 桁程度向上している．

このことにより要求条件 5 にあがっているクラウド間連携で実用的な性能でマシンイメージのスナップショット作成やその起動などが実施できる見通しができた．

5.3 学認連携機能評価

要求条件 6 については学認連携機能が寄与することを想定している．



・Download – without cache

	1	2	3	4	5
1K	11,608	13,293	12,916	14,691	12,126
1M	3,427,827	3,427,402	3,455,788	**673,614**	3,386,009
10M	**958,018**	**959,098**	7,188,901	**974,072**	**949,389**
100M	**1,109,440**	98,400,555	99,641,184	15,411,221	92,400,774

・Download - with cache

	1	2	3	4	5
1K	398,133	415,753	397,979	446,576	406,510
1M	9,238,473	202,193,598	177,694,628	161,992,275	134,450,057
10M	55,186,177	288,776,403	319,892,614	331,691,392	307,500,293
100M	473,135,008	471,484,134	467,654,981	412,153,419	451,464,516

・Upload – without network latency-awareness

	1	2	3	4	5
1K	**1,755**	**2,165**	**395**	**3,178**	**1.375**
1M	**431,840**	**446,824**	**537,768**	**412,542**	**43,307**
10M	**710,179**	**741,350**	**737,565**	**723,446**	**731,824**
100M	**780,548**	**775,131**	**788,620**	**782,503**	**786,224**

・Upload -with network latency-awareness

	1	2	3	4	5
1K	**11, 356**	**13, 157**	**13, 074**	**12, 758**	**12, 680**
1M	**9, 824, 750**	**11, 205, 249**	**7, 599, 312**	**10, 931, 206**	**11, 199, 982**
10M	**52, 294, 403**	**51, 437, 092**	**51, 050, 686**	**52, 641, 471**	**52, 300, 141**
100M	**97, 937, 987**	**101, 847, 002**	**102, 385, 002**	**102, 413, 801**	**101, 462, 855**

図 10 Colony の性能評価結果
Fig. 10 Colony performance.

Dodai-deploy，dodai-compute のフロントエンドにそれらを統合する web サービスを開発し，それに学認対応機能を持たせた．さらには colony のフロントエンド web サービスにも学認対応機能を持たせた．これらを学認フェデレーション内のサービスプロバイダとして登録することで，リアルな学認 ID を使って，それぞれのサービスの認証・承認が実施できることを確認した．このことにより要求条件 6 である外部認証の利用によるユーザ管理が実現できることを確認した．

6. 類似事例

インタークラウドオブジェクトストアを目指した colony に類似の研究は筆者らの知る限りでは見当たらない．一方 OpenStack オープンソースコミュニティではマルチサイトの swift に関しての議論が立ち上がろうという機運がある．このため，筆者らはこのコミュニティへも colony の提案を実施している．

Dodai-compute のようにベアメタルも制御しようとしているクラウド関連プロジェクトとしては VCL [11]，crowbar [12]，MAAS [13] がある．いずれも仮想マシンへのインタフェースとは異なるインタフェースになっているため IaaS のレイヤと CaaS のレイヤで同一のソフトウェアを使えない．

Dodai-deploy のような Deployment 用のフレームワークとしては crowbar の barclamp や JuJu [14] などがある．前者はデータセンタなどでのクラスタの初期構築に焦点をあてたものであり，物理マシンでのクラスタ構築とその上のクラウド基盤構築が一体となった動きする．後者は提供元である Canonical が配布している Linux ディストリビューション Ubuntu に依存した部分が大きい．Dodai-deploy は物理マシンおよび仮想マシン，さらにそれらを混在させたクラスタ上で動作する OS のディストリビューションに依存しないニュートラルな Deployment 用のフレームワークとしての新規性を持つ．

7. まとめ

5 章の各サービスコンポーネント評価のところに述べたように，terminate-instance 時のバックヤードでの処理時間の向上や dodai-deploy のスケーラビリティの確保などさらに改善する点もあるものの，dodai-compute により要求条件 1，2，3，dodai-deploy により要求条件 4，colony により要求条件 5，学認連携により要求条件 6 が実現できる見通しが立ち，本提案アーキテクチャに沿ったアカデミッククラウド構築の実現可能性が検証できた．

今後は，このアーキテクチャを実装したアカデミッククラウドを構築運用することで，さらなる要求条件の精緻化や改善項目の獲得を実施すると同時にアカデミックコミュニティクラウド構築に向けたテストベッドを構築し，広域分散クラウド連携を現実のものとするために活用してゆきたい．

参考文献

[1] Yoshioka, N., Yokoyama, S., Tanabe, Y. and Honiden, S.: edubase Cloud: An Open-source Cloud Platform for Cloud Engineers, *SECLOUD '11, Proc. 2nd International Workshop on Software Engineering for Cloud Computing*, p.73 (2011).
[2] 学認，入手先 ⟨https://www.gakunin.jp/⟩.
[3] Yokoyama, S. and Yoshioka, N.: Cluster as a Service for self-deployable cloud applications, *CCGrid 2012*, pp.703–704 (2012).
[4] Dodai, available from ⟨https://github.com/nii-cloud/dodai⟩.
[5] OpenFlow, available from ⟨http://www.openflow.org/⟩.
[6] Puppet, available from ⟨http://puppetlabs.com/⟩.
[7] MCollective, available from ⟨http://puppetlabs.com/mcollective/⟩.
[8] OpenStack, available from ⟨http://openstack.org/⟩.
[9] Yokoyama, S., Yoshioka, N. and Shida, T.: Cloud in a Cloud for Cloud Education, *Principles of Engineering Service Oriented Systems (PESOS) ICSE*, pp.63–64 (2012).
[10] Colony, available from ⟨https://github.com/nii-cloud/colony⟩.
[11] VCL, available from ⟨http://vcl.ncsu.edu/⟩.
[12] Crowbar, available from ⟨https://github.com/dellcloudedge/crowbar⟩.
[13] MAAS, available from ⟨https://launchpad.net/maas⟩.
[14] JuJu, available from ⟨https://launchpad.net/juju⟩.

横山 重俊

1979 年大阪大学理学部数学科卒業．1981 年大阪大学大学院理学研究科修士課程修了（数学専攻）．同年日本電信電話公社横須賀研究所入所．オペレーティングシステム，分散処理技術，インターネット技術，クラウドコンピューティング基盤技術等の研究開発に従事．この間（1989～1991 年）マサチューセッツ工科大学客員研究員．現在，国立情報学研究所勤務．電子情報通信学会会員．

桑田 喜隆 （正会員）

1986 年群馬大学大学院工学研究科修士課程修了（電子工学専攻）．同年日本電信電話（株）に入社．エキスパートシステム，リアルタイム AI，マルチエージェントシステム，グループウェア，クラウドコンピューティング基盤技術等の研究開発に従事．この間（1991～1993 年）マサチューセッツ大学計算機科学科客員研究員．現在，（株）NTT データ．人工知能学会，計測自動制御学会，AAAI，ACM 各会員．博士（工学）．

吉岡 信和 （正会員）

1971 年生．1993 年富山大学工学部電子情報工学科卒業．1998 年北陸先端科学技術大学院大学情報科学研究科博士後期課程修了．博士（情報科学）．同年（株）東芝入社．2002 年より国立情報学研究所に勤務．現在，同研究所准教授．2007 年より総合研究大学院大学准教授を兼務．セキュリティ技術，エージェント技術，ソフトウェア工学，学術クラウドの研究・開発に従事．2011 年より日本ソフトウェア科学会理事，電子情報通信学会，日本ソフトウェア科学会各会員．