総合研究大学院大学博士課程

情報学専攻博士論文

アカデミックコミュニティクラウドアーキテクチャに関する研究

横山重俊

総合研究大学院大学博士課程

情報学専攻

指導教員　吉岡信和准教授

２０１３年　３月

概要

クラウドコンピューティングに必要なリソースの確保を各組織で個別に実施するより，それらの組織群が形成するコミュニティ全体で賄うことで，リソースの有効利用を図ろうとする動きがある．このために，クラウド間でのリソース融通を行うインタークラウド基盤構築が各コミュニティで推進されて来ている．本研究では対象コミュニティ領域をアカデミックコミュニティとして，インタークラウド基盤構築によるコミュニティ内でのリソース利用の効率化を促進する新しいアーキテクチャの提案とその評価を行い，従来のアーキテクチャに比較して，その実現性と有効性の高さを示す．

現在のインタークラウド基盤構築にあたって取られているアプローチは二つある．一つはクラウド技術の標準化によるものであり，各クラウドを標準に準拠させることでクラウド間連携を促進する．もう一つは，クラウドフェデレーションポータルの構築により各クラウドを連携させるハブ的機能を構築することでクラウド間連携を進める．前者はまだ技術革新が続いているクラウド分野での標準化の難しさや，その標準への準拠のための各クラウド側での投資が制約となって，クラウド連携実現に時間がかかっている．後者は，各クラウドの仕様差分の吸収で工夫が必要で特定のクラウド間での機能限定の連携にとどまっている．

本研究では，特にアカデミックコミュニティが持つ先進的な利用環境（SINET-4 のような高速かつセキュアで安定運用されているネットワーク，学認のような高度な認証連携基盤）に着目し，それらの環境を活用した，アカデミックコミュニティクラウドに適用可能な新しいアプローチとインタークラウド基盤アーキテクチャを提案する．

具体的には，従来のアプローチが，その時に存在する既存のクラウド間の連携のみを対象にしていることにそれらのアプローチが持つ課題の源泉があるため，この前提を取り除くというアプローチを取る．つまり，クラウド連携の要請がある度に，オンデマンドで必要なクラウドを必要な場所に必要な構成で構築するアプローチである．このことで，各クラウドが必要に応じて拡大したり，移動したりできるようになり，クラウドの標準化やフェデレーションを必要とせず，クラウド連携で想定したユースケースが実現できることを示す．

さらに，そのプロトタイピングと評価を通じ，提案アーキテクチャの実現性と有効性を検証する．また，その限界を議論すると同時に将来展開の可能性についても述べる．将来展開の一つとしては，今回前提とするネットワークや認証連携基盤が利用できるという前提がアカデミックコミュニティ以外のコミュニティ領域（業界や企業グループなど）に広がった場合，それらのコミュニティでも本研究の成果が利用できる可能性がある．

目次

第1章　序言

クラウドコンピューティングに必要なリソースの確保を各組織で個別に実施するより，それらの組織群が形成するコミュニティ全体で賄うことで，リソースの有効利用を図ろうとする動きがある．このために，クラウド間でのリソース融通を行うインタークラウド基盤構築が各コミュニティで推進されて来ている．本研究では対象コミュニティ領域をアカデミックコミュニティとしてクラウド連携によるコミュニティ内でのリソース利用の効率化を促進する新しいアーキテクチャの提案とその評価を行い，従来のアーキテクチャに比較して，その実現性と有効性が高くこのアーキテクチャを用いたアカデミックコミュニティクラウド構築への可能性を示す．

1.1　アカデミックコミュニティクラウドとは

近年，クラウドコンピューティング[1]- [9]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．例えば，予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，必要な時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．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，お互いの持っている機能やリソースおよびデータを共有することにより，突発的なリソース増強や災害時への対応を相互協力で実現しようという，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来ている[10]．

1.2　研究の背景と目的

本研究の背景となるアカデミッククラウドの現状とその目的である目指すべき実現したいアカデミックコミュニティクラウド像について述べる．

1.2.1　アカデミックコミュニティクラウドの現状

各大学や研究機関でクラウドの柔軟性を活用した IT 基盤の構築が個別に始まっている．これらのクラウドはそれぞれの機関の事情によりサイズやアーキテクチャが様々でヘテロなクラウドの集まりとなっている．アカデミックコミュニティクラウドは，これらの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，お互いの持っているリソースを連携させることにより，突発的なリソース増強や災害時への対応を相互協力で実現しようという，アカデミックコミュニティの連邦型のクラウド連携である．従来，大学や研究機関が独立にプライベートクラウドを運営することでは限界のあった上記のユースケースに対応するために必要なリソース不足を，コミュニティとして解消しようとする試みである．クラウド上のアプリケーションは，自ら様々なモニタリングをすることで，必要に応じ，クラウド基盤が持つインタフェースを使って動的にリソース変動させることができる．例えば，アプリケーションへのトラフィックの変動へ自ら柔軟に対応できる．既存研究によってアプリケーションを構成するプログラムを動的にクラウド上の仮想マシンに配備する手法が有効であると示されている．本研究では，こうしたアプリケーションをクラウド型アプリケーションと定義する．既存クラウド連携機構は，クラウド型アプリケーションが本来のプライベートクラウドのリソース不足により他のプライベートクラウドのリソースを活用する際に以下の 二つの問題がある．

問題１：クラウド型アプリケーションに様にクラウド基盤を活用する場合，他クラウドと連携す

るためには，クラウド基盤間の API 差を吸収し，さらにネットワーク接続をアプリケーション個別に構築するなどプログラマにアドホックな実装を強要する．その結果，汎用性の低いプログラムとなってしまうため，クラウド型アプリケーションの継続的な保守を困難にする課題がある．
問題２：クラウド型アプリケーションが別のプライベートクラウドにマイグレーションする必要がある場合，マイグレーション先のクラウド基盤でのマシンイメージの起動のためには，マシンイメージの変換後，そのマシンイメージをネットワーク経由で転送とマシンイメージ登録を実施してからマシンイメージ起動を実行する必要がある．マシンイメージはそのサイズが大きいことからその変換や転送に要する時間を許容できない場合がある．

問題１や問題２を引き起こさない方策として，クラウドの標準化（クラウド基盤 API やマシンイメージフォーマット）やフェデレーションポータルによるマルチクラウドサポートが叫ばれて長いが，それぞれの要求に沿うための仕様は膨大となり，実装が伴った実用的なクラウド連携に適用されている標準は現在のところ存在しない．

すなわち，クラウド連携の従来のアプローチである標準化では，仕様が複雑になり実装が出来ていないし，今後実装されると言う見通しが無く，実現性が低い．たとえクラウド連携の標準化が進み，実装もいくつかのクラウド基盤に関してなされたとしても，今後クラウド基盤側の進展にともなう変化への対応を各クラウド基盤側で継続的に行なわなければならないというコストが発生する．現在議論されている標準化案の複雑さから，このコストはクラウド連携により得られるメリットを得るために妥当なサイズに収まらない可能性が高い．もう一つのアプローチであるクラウド連携ポータルでは，ポータルから呼び出されるクラウド基盤間の共通機能のみしか使えないという制約があり，クラウド連携により他クラウドを使えたとしても，既存のクラウド機能の一部の機能のみしか使えない場合があり，クラウド連携の有効性が低い範囲にとどまっている．

従って，実現性と有効性の両方を満たすクラウド連携方法がないためアカデミックコミュニティクラウドを構築する上で重要なクラウド連携が進まないという課題を抱えている．

1.2.2 アカデミックコミュニティクラウドの実現に向けて

一般のクラウド連携の実現のためは，既存のパブリッククラウドやプライベートクラウド間を接続するネットワークについての特定の前提条件を設定できないため，インターネット経由での接続を前提としてアプローチにならざるを得ない．インターネット上にそれぞれ VPN を構成してクラウド接続するのが一般的な方法となっている．従ってクラウド連携のために VPN 接続レベルのネゴシエーションプロトコルや，それを経由してリソース要求する場合のネゴシエーションプロトコルなど各レイヤのネゴシエーションをクラウド基盤種対毎に実施できる必要がある．クラウド基盤種対毎での対応では，そのコストが増大するので，そのコスト削減のために，クラウド連携に必要なプロトコル仕様の標準化や，クラウド連携ポータルのようにハブ機能側でクラウド基盤種に依存する差分を吸収するなどのアプローチが取られている．上述のように前者は実現性に課題があり，後者は有効性に課題があるため，現在まで有効性の高いクラウド連携が実現できていない．あるいは現実路線として，ハイブリッドクラウドのように一つのプライベートクラウドとパブリッククラウド連携という限定した実現のレベルにとどまっている．コミュニティクラウド形成の主な目的であるプライベートクラウド間の連携によるリソース融通を達成する実現性，有効性ともに高い方法が存在していない．

一方，アカデミックコミュニティクラウドと適用対象を限定した場合は，連携対象となるアカデミッククラウドが SINET に接続されているという前提が成り立つため，SINET の持つ機能である L2VPN/VPLS 機能を使うアーキテクチャが適用可能である．今までのクラウド連携ではネットワークに対する前提条件が置けなかったため L2 接続によるクラウド基盤の延伸というクラウド

連携手法は取れなかった．また，管理主体が異なるサービス間の認証をフェデレーションするアカデミックコミュニティで利用されている認証基盤「学認」によるサービス連携を実現できる．

アカデミックコミュニティに限定することで，利用広域網である SINET のネットワーク機能や認証基盤機能を用いた新しいアプローチを適用し，そのアプローチに従ったアーキテクチャを提案できる．適用範囲を限定し，その範囲で使える機能を前提とした，今まで一般的な適用範囲では取れなかったアプローチとアーキテクチャを提案し，その実現性や有効性を評価することに意味がある．

1.2.3 本論文の貢献

現在，アカデミッククラウドとして最大規模のものはコア数で 2,000 コアとなっている．このため，今後の拡大および複数アカデミッククラウドとの同時連携を考慮し，インタークラウド基盤側は 20,000 コアを連携規模とする．第 4 章で述べる全てのユースケースともクラスタの動的構築およびその上のクラウド基盤の構築，その上での仮想マシン群の立上によるサービス提供までをトータルにして，1 時間以内に達成できれば実用的と言えるため，コア数 2,000 コアのクラウド基盤をこの時間内に立ち上げ運用できることを性能に関する目標とした．但し，急激な負荷増加に対する性能保証に関するユースケースについては，クラウド基盤リソースが不足した場合の拡充の速度について，さらに迅速な対応が求められる可能性があるため，逐次クラウド基盤を拡張する場合については 10 分以内の立ち上げを性能目標とした．このような規模と性能要求を満たすアカデミックコミュニティクラウドのアーキテクチャを提案し，プロトタイピングを通じて実証したことが本論文の貢献である．

本研究では，特にアカデミックコミュニティが持つ先進的な利用環境（SINET-4[11]のような高速でセキュアで安定運用されてネットワーク，学認[12] のような高度な認証連携基盤）に着目し，それらの環境を活用した，アカデミックコミュニティクラウドに適用可能な新しいアプローチとクラウド基盤アーキテクチャを提案する（図 1-1）．

既存のアプローチがクラウド基盤の管理主体とそれを実行するハードウェアの管理主体が同一であると言う前提を持ち，連携時にすでに存在する既存のクラウド間の連携を対象にしていることに，それらのアプローチが持つ課題の源泉があることに着目する．どちらもすでにあるクラウド基盤間をつなぐための接続部分を作る困難さが課題のもととなっている．

既存のクラウド間をクラウド基盤の管理主体とそれを実行するハードウェアの管理主体を分離し，オンデマンドで必要な場所にハードウェアを確保し，その上にクラウド基盤を動的に構成可能とするアプローチを取る．このことで，各クラウドが必要に応じて拡大したり，移動したりできるようになり，クラウドの標準化やフェデレーションを必要とせず，クラウドがそのままのクラウド基盤 API やマシンイメージフォーマットを維持したまま，拡大，移動するためアカデミックコミュニティクラウド内でのクラウド型アプリケーションが走行できる継続的な実行が実現できる．このアプローチに従って，広域網の持つ L2VPN 機能を活用することで動的に構成するクラウド基盤を広域網上で動的に拡張できる仕組みを基本としたクラウド連携アーキテクチャを提案し，具現化した．

クラウド連携アーキテクチャは地域分散しているインタークラウド基盤センタ内にあるベアメタルマシンを動的にクラスタ化して，その上にクラウド基盤を自動構築することを Cluster as a Service と呼ぶ Web サービスとして提供する部分と，それらのクラウド基盤から共有される地域分散型のオブジェクトストレージサービスから構成される．Cluster as a Service を用いることで SINET を経由してベアメタルマシンを自分の持っているクラスタに L2 レベルで延伸できる．ベアメタルマシンが扱えることの重要性は，本提案アーキテクチャにおいて，クラウド基盤を広域に L2 網で拡張できることである．クラウド基盤では通常 Hypervisor によるサーバの仮想化技

術を活用する．この際，各種仮想化技術を各クラウド基盤用に実現するためには，仮想化の上の仮想化ではなく，ベアメタルからの仮想化である必要がある．また，アカデミックコミュニティでは性能評価の際などにベアメタルマシンを直接使いたいと言う要求もあるので，その要求に対応できる．

このアーキテクチャで，実現性と有効性の高いクラウド連携が実施できることをアカデミックコミュニティクラウドに関するプロトタイピングにより検証した．

具体的には，すでに標準化が定まって普及しているベアメタルマシンの起動・制御部分を出発点として，それを使った動的クラウド基盤構築手法を確立する．従来のアプローチのような既存のクラウド基盤を連携させる方法ではなく，必要に応じてクラウド基盤を創造することでクラウド連携を実現することを基本とするアカデミックコミュニティクラウドアーキテクチャを提案し，実現の際に解決しなければならない以下の課題に取り組み，上記プロトタイピングを通じて，その解決策を見出した．

1．　ベアメタルマシンを閉域ネットワークに組み入れたクラスタを数秒以内で動的に構成する．この際，クラスタ上にデプロイするクラウド基盤間では仮想マシンに割り当てる IP アドレス，MAC アドレスや VLAN-ID について同一のものが重複して利用される可能性があることへの対処が必要である．

2．　動的に構築したクラスタ上にクラウド基盤をオンデマンドでデプロイする．この際，デプロイ時間はクラスタのサイズに依存せずスケーラブルである．

3．　クラウド基盤から利用するマシンイメージやデータへのアクセスが高速に行なえる．

研究成果を実際の SINET と学認と連動するインタークラウド基盤の構成に活用することで，ネットワーク，認証基盤それにクラウド基盤が一体となった学術基盤を構成し，アカデミッククラウド利用者にとって使い易く，さらに投資対効果の高いサービス提供することで，アカデミックコミュニティクラウド実現を促進することができる．

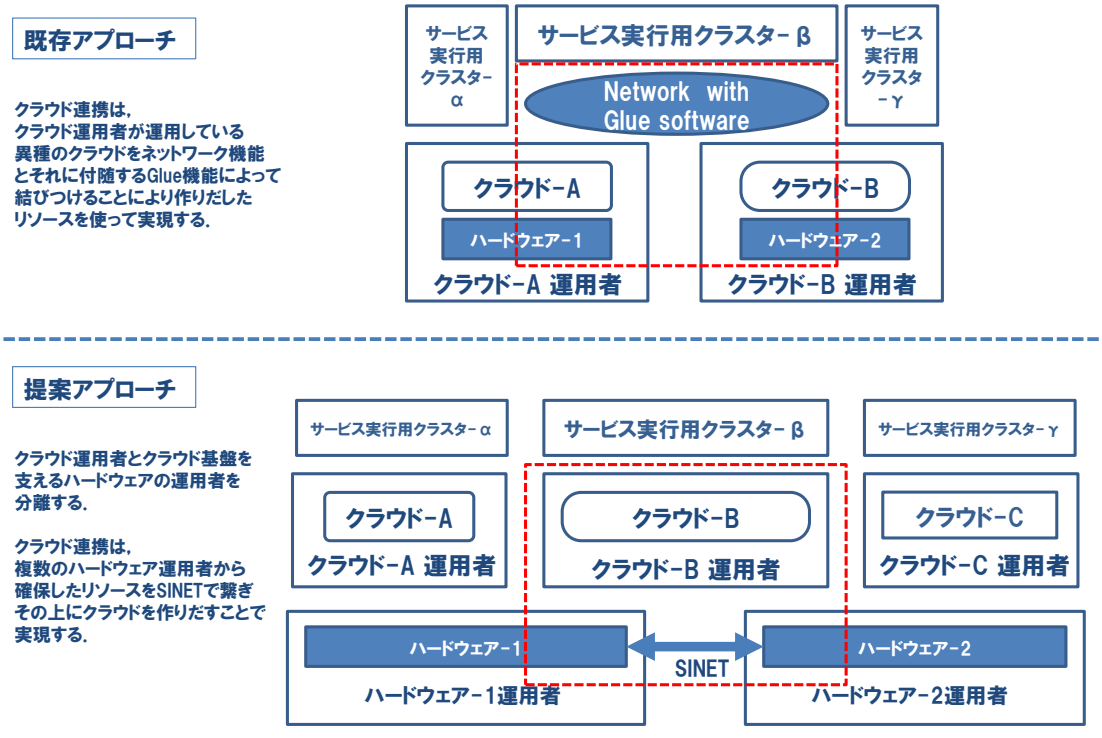


図 1-1　　既存アプローチと提案アプローチの比較

1.3 本論文の構成

本論文は 9 章から構成される．第 1 章は，本研究の背景や目的について述べる．第 2 章では，クラウド基盤連携に関する代表的なユースケースについて述べる．第 3 章では，クラウドコンピューティングの分野における動向と，従来のアプローチが持つ課題について説明する．第 4 章で，この課題を解決する本研究で追求する新しいアプローチの提案を行う．さらに，このアプローチに沿ってインタークラウド基盤を構成する際に必要な設計方針を整理する．第 5 章では，この設計方針を達成するためのアーキテクチャを示す．第 6 章では，アーキテクチャの有効性と実用性を検証するために構成したプロトタイプについて説明する．第 7 章で，関連研究についてまとめ本提案の位置づけを述べる．第 8 章では，解決策として提案した，動的配備を行う際のアーキテクチャについて，実現性と有効性の観点から議論する．第 9 章で，研究を通じて得られた知見をまとめると同時に，提案手法の有効性および限界について示し，本論文を結ぶ．

第2章　クラウド基盤連携ユースケース

具体的なアカデミックコミュニティクラウドへの設計方針を明確にするための出発点として，コミュニティクラウドでのクラウド連携のユースケースからクラウド標準プロトコルまでを一貫性を持って議論したグローバルクラウド基盤連携技術フォーラム（GICTF）[13] で策定されてあユースケースを例に取る．この活動自体は第 3 章で述べるようにクラウド標準化という既存の第一のアプローチに沿った活動であるので，そのアプローチとの比較をする意味でも同一のユースケースを基準とすることとする．GICTF では，クラウド基盤連携の議論の出発点となるクラウド連携のユースケースとして以下の６つが提示されている．

U1: 急激な負荷増加に対する性能を保証
U2: 遅延に対する性能保証
U3: 災害や大規模故障発生に対する可用性保証
U4: サービス継続
U5: サービス連携による利便性向上
U6: ブローカー介在による市場取引

これらのユースケースを層別すると，このうち U1-U4 はクラウド基盤側の連携に関するものであり，アカデミックコミュニティクラウドで特に重要となると考えられる．U5 についてはその基盤の上にのるサービス間の連携であり，サービスコンピューティングの手法を用いてアプローチする分野であると考えられるれ，さらにアカデミックコミュニティクラウド特有の対象ではない．U6 については，市場性の問題であり，本研究の主要な対象から外して議論する．

一方，グローバルクラウド基盤連携技術フォーラムではあがっていないアカデミックコミュニティクラウド特有のユースケースとして，巨大なデータやスパコンなどの移動が困難な特殊リソースを活用した計算処理を高速に行なうというものがある．

U7: 特殊リソースを活用した効率的な計算処理

クラウド連携のユースケースは，あるクラウドが何らかの理由により他のクラウドのリソースを借用するというユースケースよりなる．U1 は自らのリソースが不足するため，他のクラウドのリソースを借用するユースケースであり，U2 は他のクラウドの方がネットワークレイテンシの関係から実行環境として条件が良いので，そちらを借用するというユースケースです．U3 は災害などにより自クラウドが使えないので他クラウドのリソースを借用して，そちらに非常用環境を構築するというユースケースである．U4 は自らのクラウドでのサービス継続が出来なくなったので，他クラウドのリソースを借用してサービス継続するというユースケースである．U7 は U2 と同様に他のクラウドの方がネットワークレイテンシの関係から実行環境として条件が良いので，そちらを借用するというアカデミックコミュニティクラウドに特有のユースケースである．

本研究では，ユースケース U1-U4，U7 を対象として，それらの実現の際に従来のアーキテクチャの持つ課題を克服する，より実現性と有効性の高いクラウド連携アーキテクチャを提案する．

以下に順にこれらのユースケースについてアカデミッククラウドのユースケースに置き換えて説明する．

2.1　急激な負荷増加に対する性能を保証（U1）

サービスの利用者は，インターネット経由で大学 A のクラウドシステムが提供している WEB サービス（Learning Management System など）へログインしていたが，学期末のため利用者からのアクセスが増大し，WEB サービスへの負荷が増加した．

そのため，クラウドシステムはモニタリングにより，負荷増加による WEB サービス の性能低下を判断し，大学 B が提供するコミュニティクラウドからリソースを借用することで同様の性

能での WEB サービス提供が可能となる.

2.2 遅延に対する性能保証（U2）

あるクラウドシステムが提供するサービス（例えば Desktop as a Serivice）の利用者が国際会議で海外出張した際，通常利用しているサービス拠点からの物理的距離が長くなることで，ネットワーク遅延の増大が発生する．これはサービスの応答性能低下を引き起こす． モニタリングによりクラウドシステムが応答の性能低下を検出し，移動先に近い拠点のクラウドシステムからリソースを借用することで，通常と同様の性能のサービスが利用可能となる.

2.3 災害や大規模故障発生に対する可用性保証（U3）

大学 A のクラウドシステムが自然災害を被災し，サービスの継続が不可能となった．あらかじめサービスの復旧（リカバリ）先となっている遠距離の大学 B，C，D のリソース（アプリケーション，ミドルウェア，DB サーバなど）を利用し，ディザスタリカバリを行う．そのことにより，大学 A により提供されていたサービスが一時的に他の大学から提供され，利用者はサービスを継続して利用可能となる.

2.4 サービス継続（U4）

大学 A の法定停電によりクラウドサービスが停止してしまうと，利用者が同じクラウドサービスを利用するには，他大学などが提供する同種のクラウドサービスに再度利用者登録などが必要となる． そのような状況を回避するため，あらかじめ大学 A の提供しているリソース，アプリケーション，利用者の ID 情報などを他大学のクラウドシステムへ転送する．これにより，利用者は大学 A のクラウドサービスの停止時にも，大学 B，C から同様のサービスを継続して利用可能となる.

2.5 特殊リソースを用いた計算処理（U7）

巨大なデータやスパコンなどの移動が困難な特殊リソースを活用した計算処理を高速に行なう．計算処理実行時間はその計算が実行される IT リソースと巨大なデータやスパコンなどの移動が困難な特殊リソースとの間のネットワーク遅延時間に依存することが多い．このため，移動の困難なリソースに計算処理を実施するリソースが動的に近付くことで計算処理時間の短縮をはかる[14].

第3章　　従来のアプローチ

第 2 章で述べたユースケースを満足するクラウド基盤連携を実現する方法として取られている従来のアプローチを整理する．大きく分けて，クラウド基盤の連携のトポロジとして以下の二つに分類される．一つは P2P，一つはハブ-スポークモデルである．

前者はクラウド基盤の標準化を進め，各クラウド基盤を中心にしてクラウド基盤間 P2P で連携する方法，後者は各クラウドのフロントエンドにあたるフェデレーションポータルを作る方法である．

3.1　　クラウド標準化によるアプローチ

3.1.1　　クラウド標準化を前提にしたモデル

クラウドの標準化を前提にしたアプローチでは，図 3-1 に示したクラウド-A 利用者には透過的にクラウド-A 運用者が必要に応じ，クライド標準のインタフェースやプロトコルを用い，クラウド-B 事業者からクラウド-B のリソースを借り出し，クラウド-A 利用者へのサービスのために利用する．逆に，クラウド-B 利用者には透過的にクラウド-B 運用者が必要に応じ，クライド標準のインタフェースやプロトコルを用い，クラウド-A 事業者からクラウド-A のリソースを借り出し，クラウド-B 利用者へのサービスのために利用する．

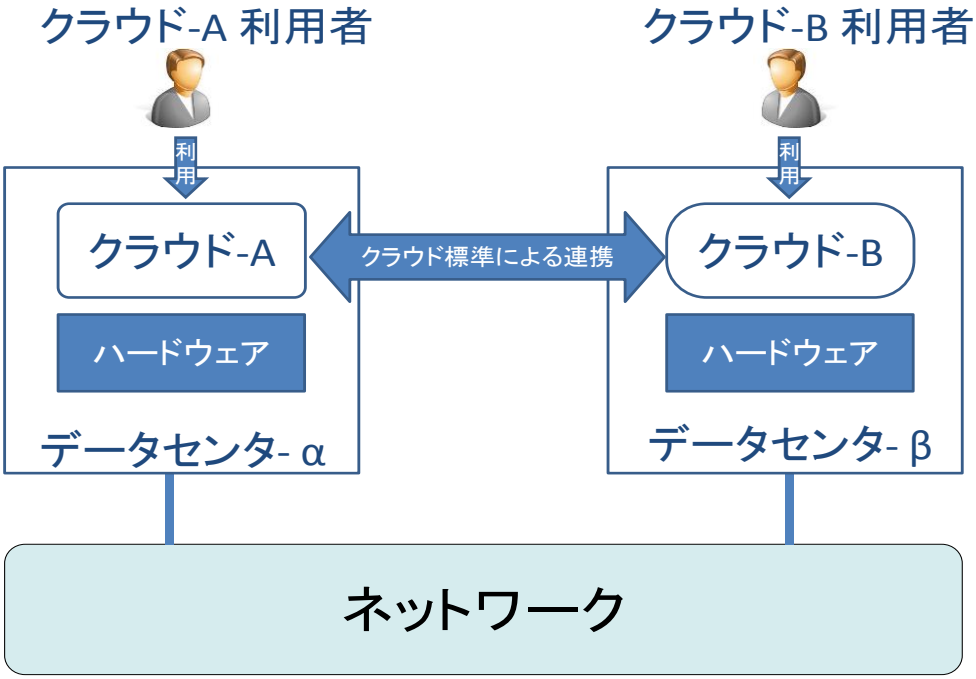


図 3-1　　クラウド標準化を前提としたクラウド連携

このことで，それぞれのクラウドの利用者にとっては自分が利用登録しているクラウドを利用しているつもりで透過的に他クラウドを間接的に利用し，U1-U4，U7 のユースケースが実現できているように見える．これは信号通信プロトコル他の標準化により国際電話網でのローミングサービスなどが機能している状況に類似している．この際，利用者への負担はない反面，提供者側

は実際に実装可能な国際的な標準化までを成し遂げ，実際に自らのクラウド基盤をその標準に準拠したものとする大きな負担がしいられる．クラウドの標準化については以下の事例に示すように，まだ標準が乱立している状況で，今後のクラウド技術の進展も吸収しつつ国際的な標準まで昇華させ，それを多くの提供者が実装するまでには多大なコストと時間が必要であると推定できる．

3.1.2　　クラウド標準化事例

各標準化団体では，IaaS の Interoperability 確保に向け，API 標準化やセキュリティ等を検討している ITU-T（FocusGroup，SG13），ISO/IEC JTC1（SC38）等，デジュール機関もクラウド標準化検討を開始した．主要クラウドベンダからは，顧客からの要望が強くない，現時点での標準化はイノベーションの障害となるなどの理由により，標準化は時期尚早との意見も出ている．図 3-2 がクラウド標準化活動の概観である．

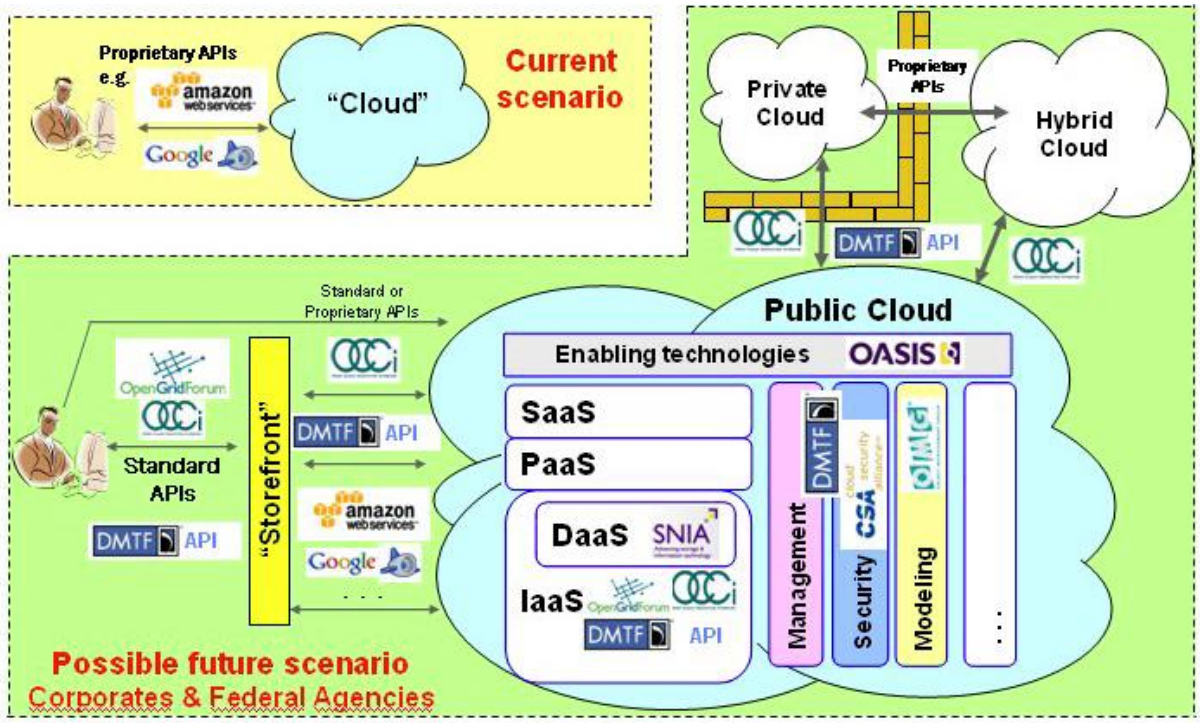


図 3-2　　クラウド標準化活動概観（GICTF 資料[13]より転載）

クラウド標準化によるクラウド連携に向かうアプローチの代表的な例として，日本のグローバルクラウド基盤連携技術フォーラム（GICTF）と欧州の RESERVOIR プロジェクトを紹介する．

3.1.2.1 グローバルクラウド基盤連携技術フォーラム（GICTF）

クラウドシステム間の連携インタフェースやネットワークプロトコルの標準化を推進し，より信頼性の高いクラウドサービスの実現等を目指して日本の標準化活動フォーラムである．

1.　設立背景

国際的なクラウドシステム間の連携を進めるために産学官合同で，クラウド間連携に関連す

る技術の研究開発や実証実験の支援を行う団体として，グローバルクラウド基盤連携技術フォーラムは設立された．

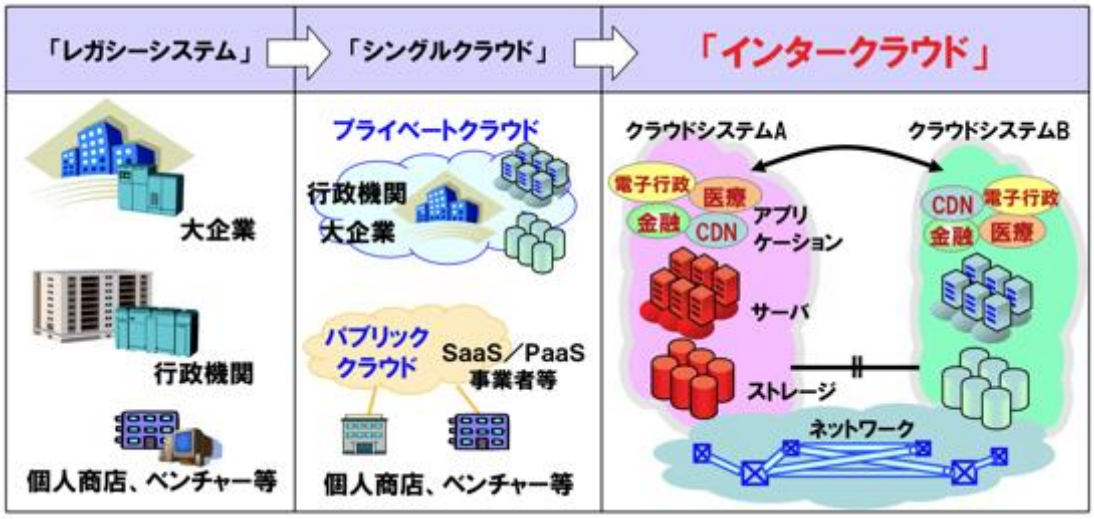


図 3-3　　インタークラウドの位置づけ（GICTF 資料[13]より転載）

2.　主な活動内容と目標

(1) クラウドシステム利用技術等の開発・標準化の推進

(2) クラウドシステム間連携を実現する標準インタフェースの提案（図 3-4）

(3) 技術交流会，講習会の開催，普及に向けた提言・要望のとりまとめ

(4) 欧米の関連フォーラムとのリエゾン，関連研究開発チームとの交流

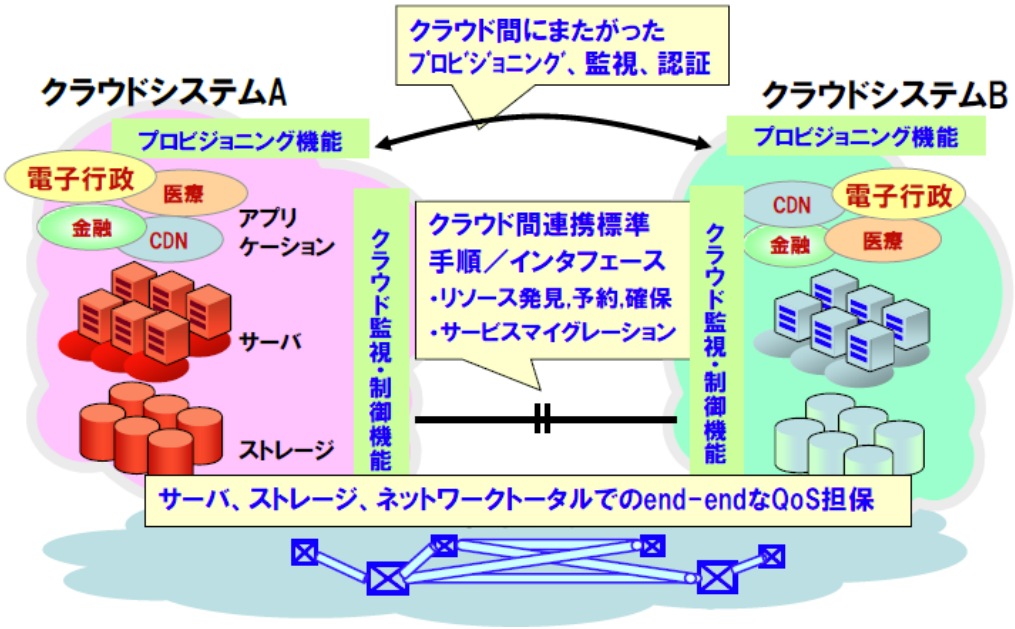


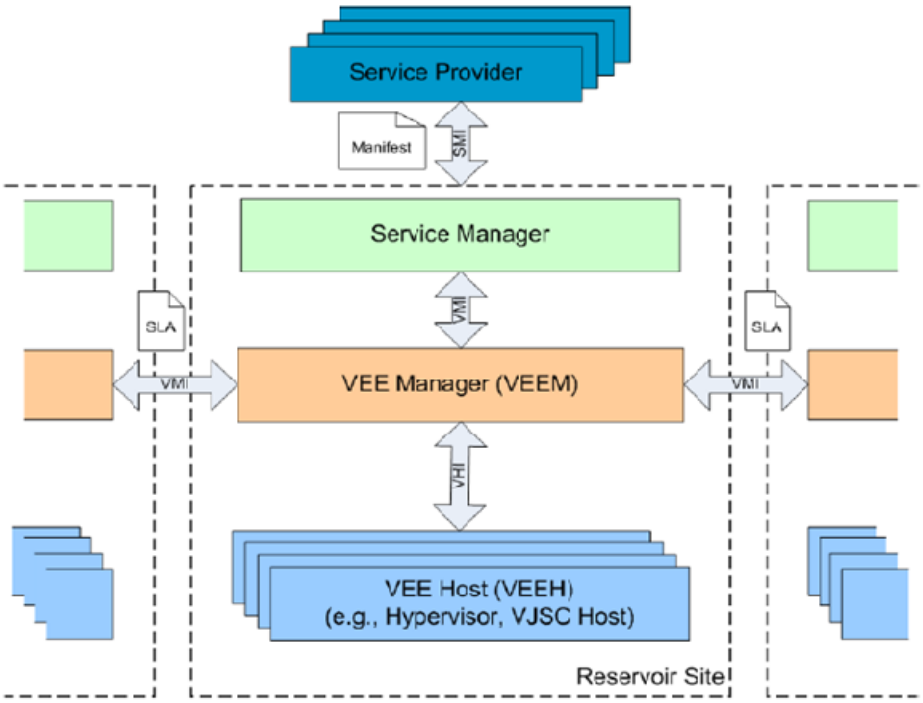
図 3-4　　クラウド間連携標準の位置づけ(GICTF 資料[13]より転載)

3.1.2.2 RESERVOIR

欧州の RESERVOIR プロジェクト[15]-[18]は「EC (European Commission) が, 2009 年春に Expert Group を DG INFSO/D3 に発足, FP7 ワークプログラムの中で活動中」に対応するプロジェクトである. FP7 の ICT 部門では, 2 年毎に作業プログラムが策定されている. 現在まで, 2007-2008 年度作業プログラム, 2009-2010 年度作業プログラム (WP2009-2010), 2011-2012 年度作業プログラム (WP2011-2012) が発表されてきたが, クラウドコンピューティングという言葉がクローズアップされるようになったのは, WP2011-2012 になってからである. これは, EU の「クラウドコンピューティングの未来」(2010 年 1 月発表) が作業プログラムの策定に影響した可能性が考えられる. だが, クラウドコンピューティングという言葉は既存の技術やサービスを総称して指示するものであり, 過去の公募 1 (WP2007-2008) 及び公募 5 (WP2009-2010) においてもクラウドに関連するプロジェクトが募集され, 採用されている. 公募 5 では, すでにクラウドという語も登場している. RESERVOIR は WP2007-2008 で採用されたプロジェクトである.

RESERVOIR は EU がスポンサーのプロジェクトで, 第 7 次研究・技術開発のための枠組み計画 (FP7) に沿い, ソフトウェアや通信分野の大手企業が中心となって, 2005 年に発足したオープンスタンダードにもとづくソフトウェアとサービス開発を目指す NESSI (Networked European Software and Services Initiative) と連携するものである. プロジェクトの課題は, 異なる IT システムやサービスのバリアを無くし, 真にユーザーフレンドリーな環境の 提供を目指すもの. このために Cloud Computing を用いて異なる IT プラットフォームや IT サービスを境目無く提供する管理技術を研究する. プロジェクトの実際の活動は, イスラエルに ある IBM の Haifa Research Lab が核となり, パートナーとして独 SAP Research, 米 Sun Microsystems, スペイン Telefonica Investigacion y Desarrollo, 仏 Thales, スウェーデン Umea University, 英 University College of London などが参加. また, NESSI には, 仏 Atos Origin, 英 BT Group, 伊 Engineering Ingegneria Informatica, IBM, HP, フィンランド Nokia, 独 SAP, 独 Siemens, 独 Software AG, 伊 Telecom Italia, スペイン Telefoica, 仏 Thales, 現 OW2 (旧 ObjectWeb コンソーシアム) などが参加している.

図 3-5 に示すアーキテクチャに基づいて標準化とそれに沿った実装を進めている.



VEE: virtual execution environment

VEE host: The virtualized computational resources, and all the management enablement components.

図 3-5　　RESERVOIR アーキテクチャ（RESERVOIR 資料[18]より転載）

マルチサイトへの配置についてはそれぞれの Service Application がそれぞれの VEEM へ依頼してリソース確保し，アプリケーション配備をすることを前提としている．（図 3-6）

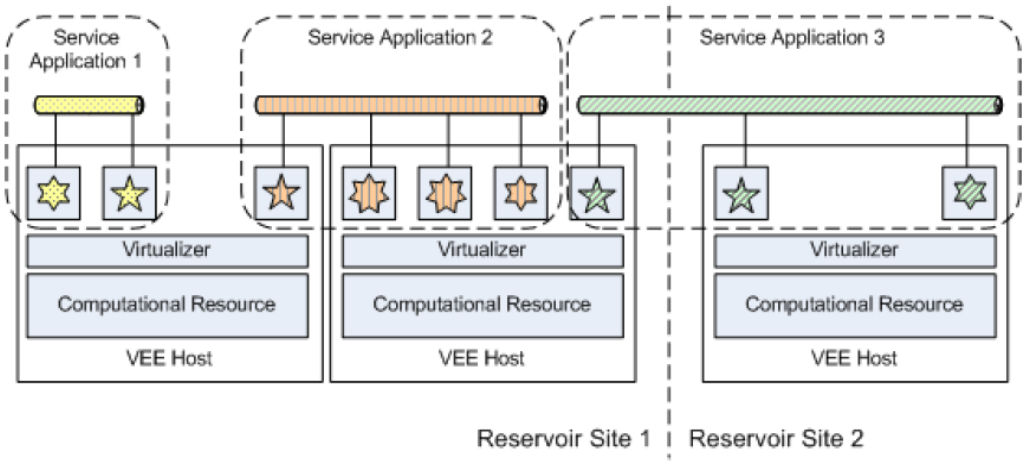


図 3-6　　RESERVOIR のマルチサイト配置（RESERVOIR 資料[18]より転載）

3.2　クラウドフェデレーションによるアプローチ

3.2.1　クラウドフェデレーションを前提にしたモデル

クラウドフェデレーションポータルを前提にしたアプローチでは，図 3-7 に示したフェデレーションポータル利用者の要求に応えてクラウドフェデレーションポータル運営者は必要に応じ，クラウド-A やクラウド-B のリソースを借り出し，フェデレーションポータル利用者へのサービスに利用する．この場合，直接クラウド-A を利用するクラウド-A 利用者はクラウド-A の利用のみできフェデレーションポータル利用者のように透過的にクラウド-B のリソースも利用することはできない．クラウド-B 利用者についても同様である．

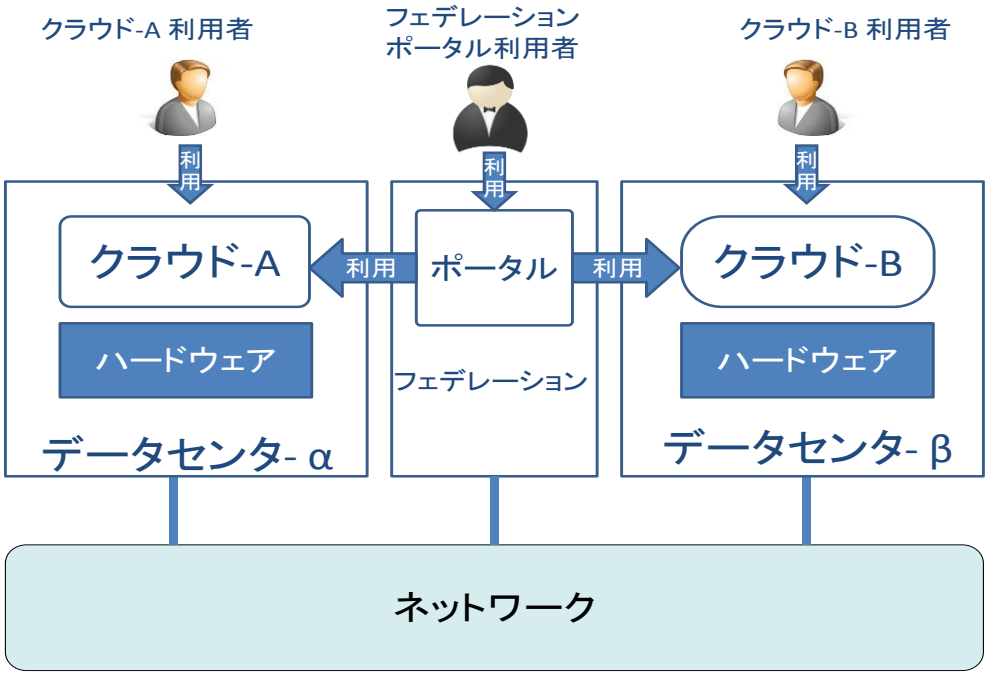


図 3-7　フェデレーションポータルを前提としたクラウド連携

フェデレーションポータルがその利用者に提供できるサービスは，委託先であるクラウド基盤の持っているサービスの最大公約数的な共通部分にならざるを得ない．最大公約数的なサービスを超えるためにはその超えた部分についてはフェデレーションポータル側で実装，サービス提供する必要があり本来のこのアプローチの趣旨に反してフェデレーションポータル側が肥大化してしまい，一つの巨大なクラウド基盤を作ることにつながる．これはクラウド連携によるコミュニティクラウド作りという方向から逸脱することになる．従って，本来利用者が得たいサービスを制限することになる．実際，このアプローチによるクラウド連携は連携先が少数にとどまる．同一クラウド基盤ソフトウェアのマルチクラウド連携にとどまるケースも多い．もちろん，この範囲でのフェデレーションによる効果はあるが，それを超える方策が今のところ発見されていない．

3.2.2　　フェデレーションポータル事例

クラウドフェデレーションによるクラウド連携に向かうアプローチの代表的な例として，クラウドAPIライブラリ delta cloudとクラウドフェデレーションツール scalrを紹介する．

3.2.2.1　Delta cloud

IaaS型クラウドに対するインスタンスの作成，起動，終了，リブートなどの操作は，当然ながらクラウドごとに独自のAPIが用意されている．The Apache Foundationがオープンソースとして開発するDeltacloud[19]は，こうしたクラウドごとに異なるAPIを共通化するためのソフトウェアである．

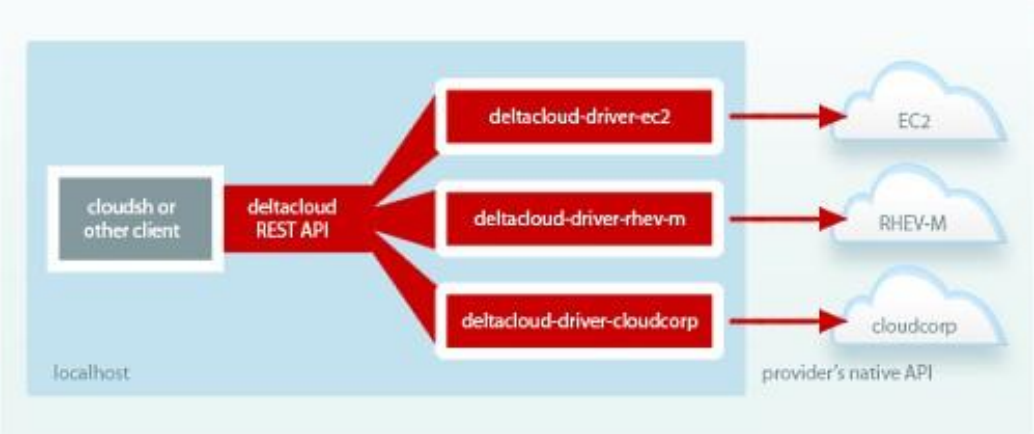


図 3-8　　Delta cloudの位置づけ（Delta cloudサイトより転載）

Deltacloudを利用することで，クラウドに対するAPIが共通化されるため，たとえクラウドを別のものに乗り換えても，アプリケーション側を変更することなくそのまま利用できるとの利点がある．Deltacloudはサーバ上でデーモンとして実行しておき，DeltacloudのREST API経由でクラウドに対してインスタンスの作成や起動といった操作を行うと，それが各クラウドのネイティブAPIへと置き換えられ，実際の操作が行われるようになっている．

3.2.2.2 Scalr

Scalr[20]はオープンソースソフトウェアとして配布されているマルチクラウド管理ソフトウェア．CloudStack，OpenStack，AWS などと連携できる．また，CloudFoundry のプロビジョニング機能も持っているため，PaaS 基盤との連携も可能である． Web ベースの UI による簡便な操作が可能で，LAMP 構成，RDBMS 用，KVS 用，ロードバランサ用などの目的に特化したロールを標準で実装している．

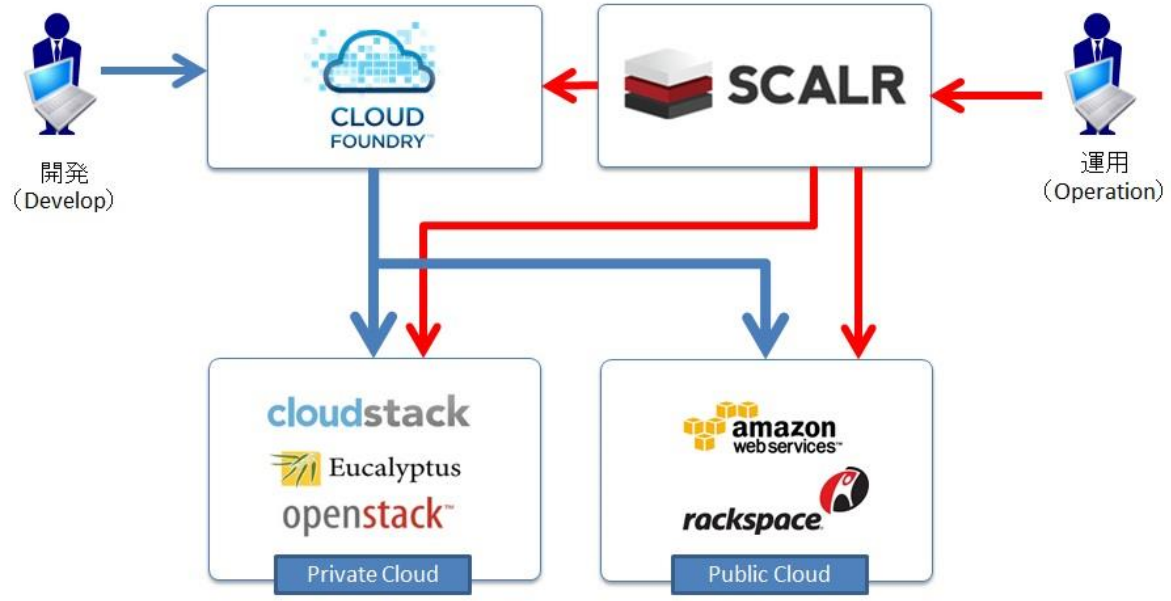


図 3-9 Scalar の位置づけ （クリエーションラインサイトより転載）

「Scalr」の主な機能

1. マルチクラウド連携機能（API 連携）
2. 仮想サーバのオートスケール機能
3. スクリプトジョブ管理
4. フォルトトレランス（Fault tolerance）機能
5. 自動リカバリ，自動バックアップ機能
6. DNS 管理
7. サーバグループ管理
8. アクセスキー管理

3.3 従来アプローチの持つ課題

第 2 章で述べたユースケースを実現するために必要なクラウド連携への既存アプローチで実施する場合の要求条件は以下の通りである．

3.3.1　　業務系連携への要求

3.3.1.1 ネットワーク関連

アドレス管理，移動管理，経路管理，VPN 管理が必要となる．

アドレス管理では，アドレス衝突の回避が重要課題である．異なったクラウド基盤間で，クラウドリソースの移動に合わせて，クラウドリソースに割り当てられたアドレスの置き換えや，アドレス解決情報の更新等の検討が必要となる．

移動管理では，クラウドシステムへの接続ポイントの特定が重要課題である．接続ポイントが明らかになることで，より最適なクラウドリソースの提供形態を確定でき，また提供形態に応じてクラウド連携を進めることで，より高度なクラウドサービスの提供につながる．

経路管理では，クラウドリソースの移動に伴うアドレスの変更やユーザ移動に合わせて，トラヒック経路を柔軟に取り扱うことが求められている．負荷分散やユーザ移動に追随した経路制御が求められる．

VPN 管理については，既存の静的なクラウド VPN 構築モデルから，動的なクラウド VPN 構築モデルへ移行するための要件が多く列挙された．クラウドシステムのリソースが，動的に移動する時代へと遷移しつつある現状を考慮すると，ネットワークを，よりダイナミックに構築・変更・削除できる機能が求められる．

3.3.1.2　コンピュート関連

インフラ管理，システム間連携が必要となる．

インフラ管理に関しては，性能管理，構成管理や障害管理，稼働管理についての要求条件がある．例えば，クラウドシステム・ネットワークは，多くの物理機器によって構成され，さらにそれらの物理機器が持つ資源が分割され，仮想機器に割り当てられる．その結果，システムの利用効率が向上するメリットがある反面，システムの構成や量を捉えにくいというデメリットもある．それは，障害発生時に障害原因を特定しにくくなり，個々の物理機器の稼働状態も把握しにくくなる．クラウドシステムやネットワークが安心して利用され，また容易に運用されていくためには，ネットワーク・インフラ管理に対する要件が重要である．

3.3.1.3　ストレージ関連

仮想ボリューム管理，オブジェクトストレージ管理が必要となる．

仮想ボリューム管理は各仮想マシンに接続し利用される仮想ボリュームを集中管理する部分である．通常，各仮想マシンホストコンピュータから iSCSI インタフェースなどのリモートストレージとして見えるストレージサーバ内に実装される．オブジェクトストレージ管理は仮想マシンイメージや各種メディアファイルなど大きなオブジェクトファイルを格納する領域として確保される．通常，分散ストレージ技術を用いて実装されることが多い．

これら通常は一つのクラウド内でサービス提供されるストレージ管理について，クラウドをまたがったサービスとして提供されることが求められる．

3.3.2　　認証・セキュリティ系連携への要求

セキュリティに関しては，暗号化／認証／承認／監視の点から，要求条件が出てくる．

マルチクラウド環境における認証系に対する要件にフォーカスし，例えばユーザの利便性につながるシングルサインオンや，クラウド VPN の安全性を維持するアクセス制御が挙げられる．一方，情報の秘匿性や，クラウドシステム・ネットワークの不正利用の監視，検出といった点の要件条件となる．

3.3.3 運用系連携への要求

運用監視に関するクラウド基盤間の連携機能が必要である．
自らのサービスに関する状況監視を一つのクラウド内にサービスが存在していたのと同様にワンストップで運用管理する機構が求められる．そうすることでクラウド連携に伴う運用稼働の増加を減らすことが可能となる．

3.3.4 クラウド連携への要求条件から引き起こされる既存アプローチの持つ課題

これらの連携を実現するためには各領域やレイヤでの標準化やフェデレーションポータルが必要になり，標準化によるアプローチの場合には，標準仕様自体が膨大となって実装及び普及を難しものとしている．またフェデレーションポータルによるアプローチでは，クラウド基盤間の差異を吸収するアダプテーションにかかる工数が莫大になるとともにクラウド基盤側の仕様変更への追随を困難にしている．

前者は標準化仕様策定を進め，その標準仕様を各クラウド基盤側に実装しなければならないという問題を持っている．つまりクラウド基盤固有の機能や API，イメージフォーマットの実装に加えて，標準インタフェースや標準イメージフォーマットの実装が必要となる．後者はそのアーキテクチャの特性から各クラウドの持つ機能の最大公約数的機能の提供になり本来利用者の満足度を上げることが難しいという課題を持っている．特にクラウド基盤のプログラマビリティを活かしたスケールアウト可能性などの機能を持たせたクラウド型アプリケーションの場合，クラウド基盤の API マシンイメージ形式をそのアプリケーションが意識する必要があるためその課題から来る制約を特に受ける．

第4章　　提案アプローチ

第 3 章で述べたそれぞれの既存アプローチが持つ課題を克服する活動はそれぞれのアプローチで続けられている．筆者らはそれらの取り組みとは別に，クラウドの動的構築を前提としたアプローチで，インタークラウド連携のユースケース U1-U4，U7 をターゲットにして，より実現性と有用性がともに高いアカデミックコミュニティクラウドの実現に取り組む．

従来のアプローチは，それらが前提としている，クラウドサービス実現のための物理リソースがクラウド連携時に固定的に割り当てられていて，その上に構築されているクラウド基盤がそれぞれヘテロな API，　機能やイメージフォーマット等を持っているという制約により，そのヘテロな環境を標準化やフェデレーションポータルで吸収する必要が出ている．これがクラウド連携の実現性や有効性を低くしている大きな要因であると分析できる．この制約を外すという発想で筆者らは新しいクラウド連携へのアプローチを構成した．

それは，各ユースケースにおいてクラウド基盤が必要になった時点でクラウド基盤自身をオンデマンドで実用的な時間内に構成するというアプローチである．このアプローチを取ることで，例えば，欲しい API，機能やイメージフォーマットを持つクラウド基盤を構築すれば良いので，クラウド基盤間の API やイメージフォーマットの変換や標準化に沿った実装なども不要となる．

4.1　　クラウドの動的構築を前提としたアプローチ

クラウド基盤に物理リソースが必要になった際にそれを供給するクラウド基盤データセンタを持つこととし，オンデマンドでその物理リソースの上にクラウド基盤を拡張したり，新規構築したり，移動したりするのがクラウドの動的構築を前提とした筆者らの提案アプローチである．

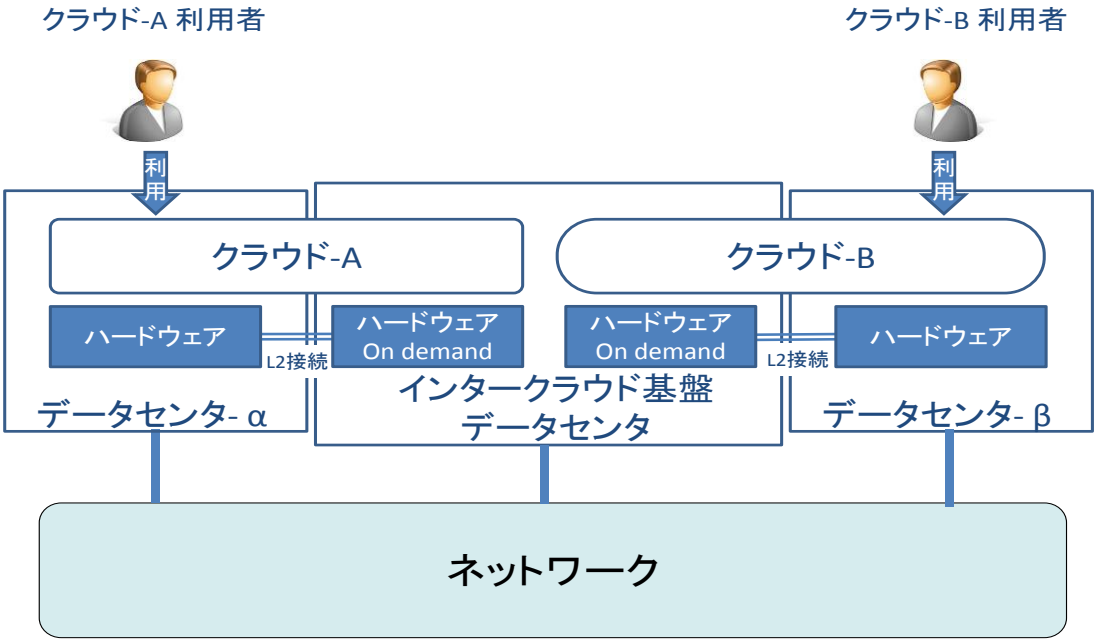


図 4-1　　提案アプローチによるクラウド連携

つまり，クラウドごとに，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)[21] 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[22] を統合し，それぞれのクラウドを遠隔地へ動的に拡大し，元々のユースケースであるクラウドバーストやディザスタリカバリ，サービス継続，リモートデスクトップ，特殊リソースを利用した計算処理に対応することである．

このアプローチの発想はレイヤ積み上げと地理的拡大の二つの軸で順次クラウド連携拡大することでクラウド連携の実現性を上げようとするものである．レイヤの積み上げは，図 4-2 に示すように，クラウド型アプリケーションの動的なデプロイメントの仕方（仮想マシンの確保と，その上へのソフトウェアインストールの自動化）を，それらのアプリケーションの乗るクラウド基盤自体にも適用する Cluster as a Service を導入するということである．

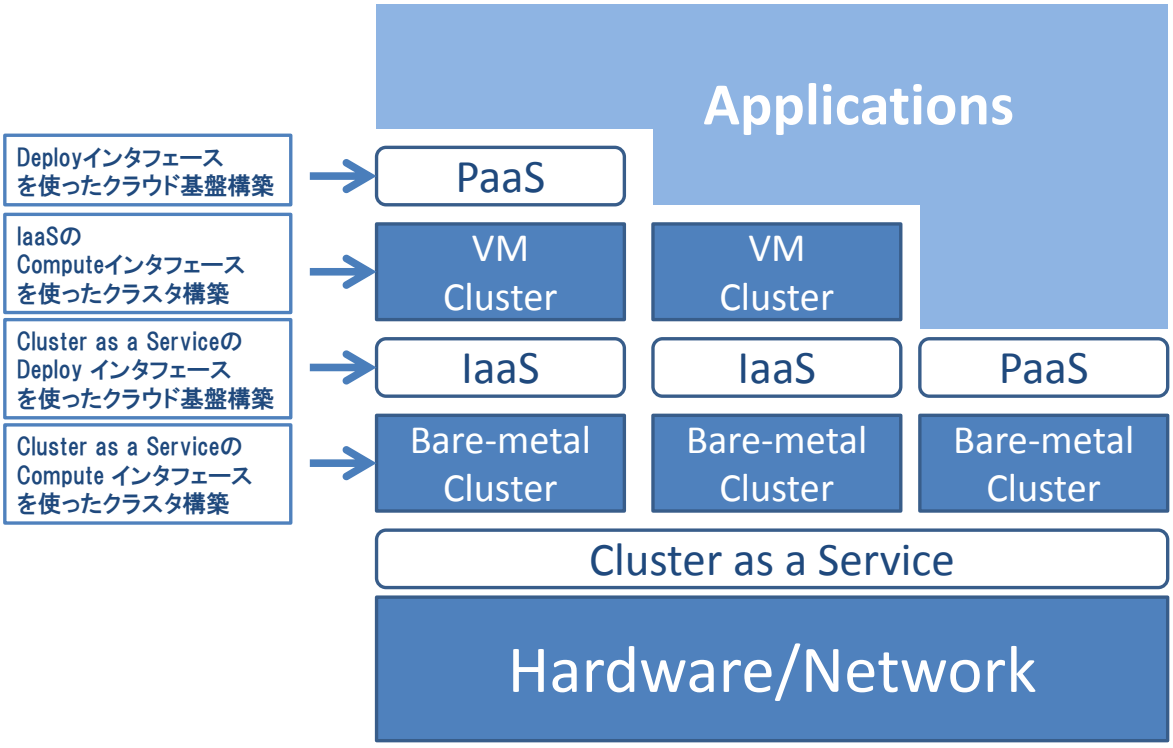


図 4-2　Cluster as a Service の導入

もう一つ空間的拡大は各アカデミッククラウドの構成アーキテクチャと同型のアーキテクチャを全国規模のクラウド連携時単に相似形で拡大するだけで適用して行くというものであり図 4-3 の各段階を経て大きなアーキテクチャ変更をすることなく，拡大することが可能な仕組みを特徴とする．

図 4-3-1 はクラウド基盤の構成要素を示す．これが基本の構造であり，この構造を相似的に拡

大して全体としてはグローバルなクラウド連携まで階層的な構造で作り上げていくアーキテクチャを目指す.

実際に仮想マシンが実行される仮想マシンホストである compute，どの compute の上で仮想マシンを実行するかを決定し，それら仮想マシンを接続する仮想ネットワークおよび仮想マシンにアタッチ，マウントする仮想ボリュームの管理を行う manager，それと仮想マシンイメージ他のオブジェクトデータを保存する Object Storage がその構成要素になる.

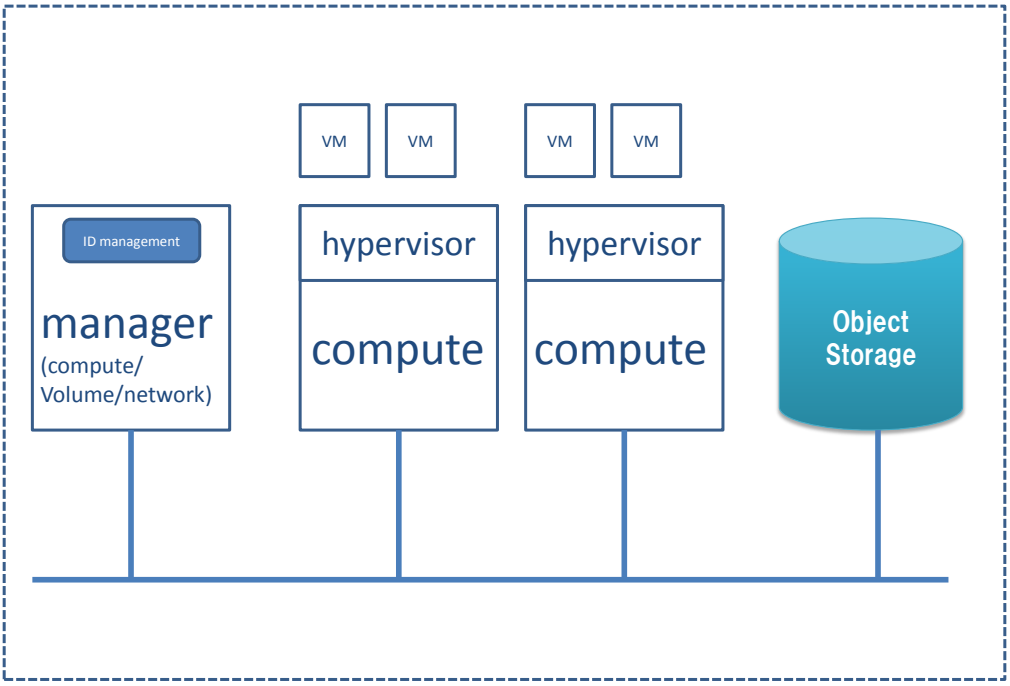


図 4-3-1　クラウド基盤の基本構成要素

通常，組織内で一つのクラウド基盤でプライベートクラウドを構成する場合は少なく，その組織に属する各グループ等の都合でプライベートクラウドをそれぞれのグループの要求に合わせた複数のクラウド基盤で構成することがある．図 4-3-2 は，二つのクラウド基盤から構成されるプライベートクラウドを示しており，その組織内の統一されたユーザ管理（例えば LDAP）と各クラウド基盤が連携し，さらに各クラウド基盤から共通に利用できる Shared Object Storage でクラウド連携がなされている状態である．この Shared Object Storage にはそれぞれのクラウド基盤で利用可能な仮想マシンイメージやそれぞれのクラウド基盤から参照されるアプリケーションデータなどがオブジェクトデータとして格納されている.

さらに一般的にプライベートクラウドを導入する組織は，既存の IT 資産(Legacy Cluster)を持っており，それと独立にプライベートクラウドを利用するのではなく，それら Legacy Cluster と構築したクラウド基盤を一体の IT 資産として利用することを望む．図 4-3-3 はそれらの Legacy Cluster を所有しているグループとそのグループが構築したクラウド基盤が直接ネットワーク接続され，Legacy Cluster がクラウド基盤を使って延伸していることを示している.

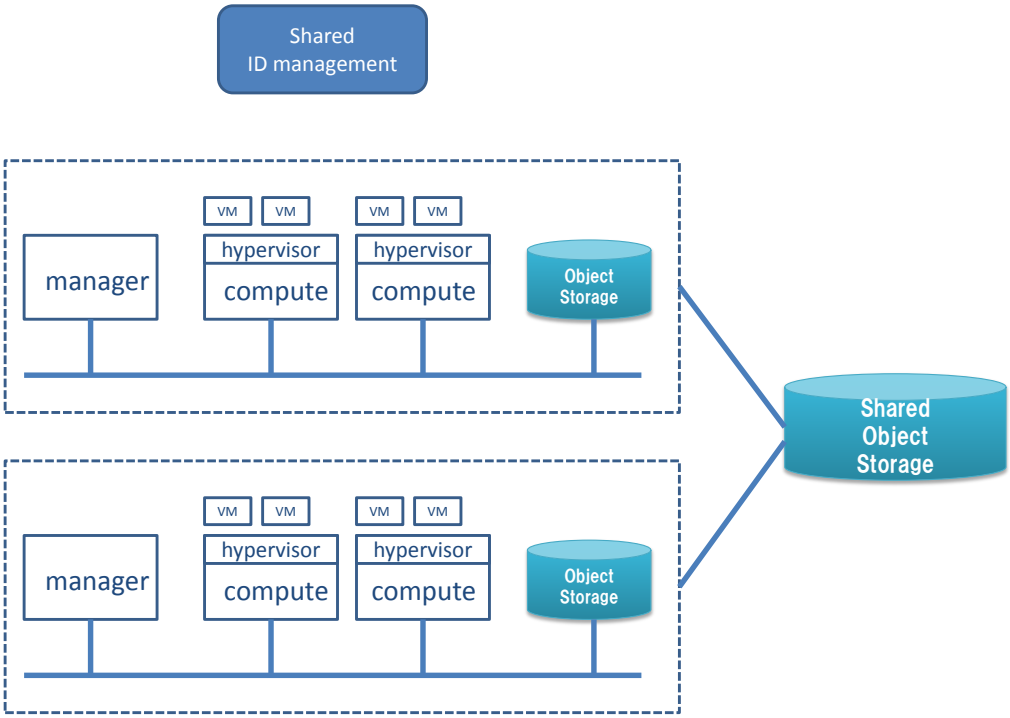


図 4-3-2　複数クラウド基盤の構成

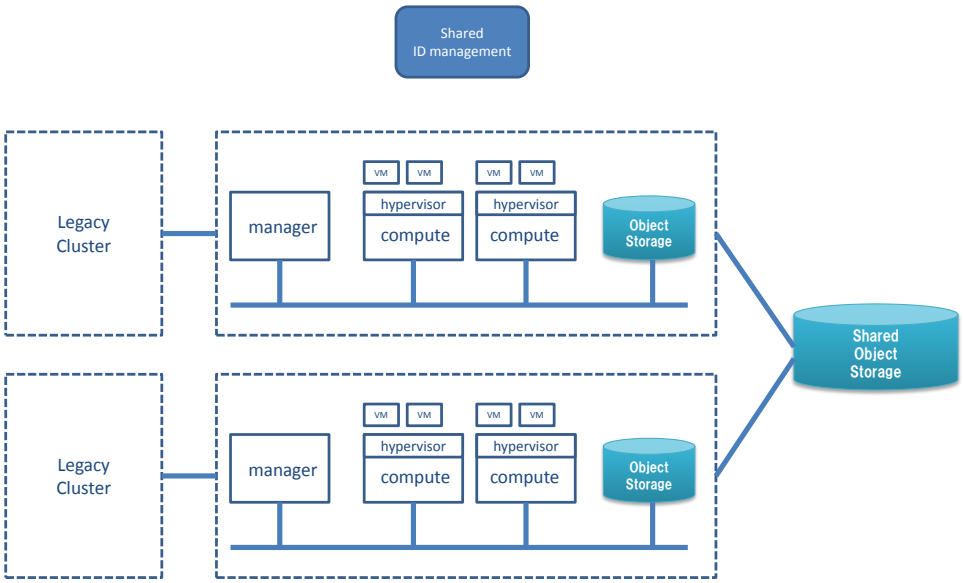


図 4-3-3　複数クラウド基盤と既存資産との連携構成

次の階層は，大学や研究所といった組織が属するコミュニティ全体のクラウド基盤の構成を図

4-3-4 に示す．右側の IT 資産はそのコミュニティ全体が共有するものとして管理されていて，災害対策やアクセス時のネットワークレイテンシを小さくするために地域分散した数か所のデータセンタに配置されている．さらにはそれらのデータセンタから共通に利用できるコミュニティ全体での Shared Object Storage も配置される．この図では Shared Object Storage 一つのストレージのように表示されているが，これは論理的な表現であり，やはり災害対策やアクセス時のネットワークレイテンシを小さくするために物理的には地域分散したストレージとする必要がある．認証は学認のような全国規模の認証基盤と連携する．

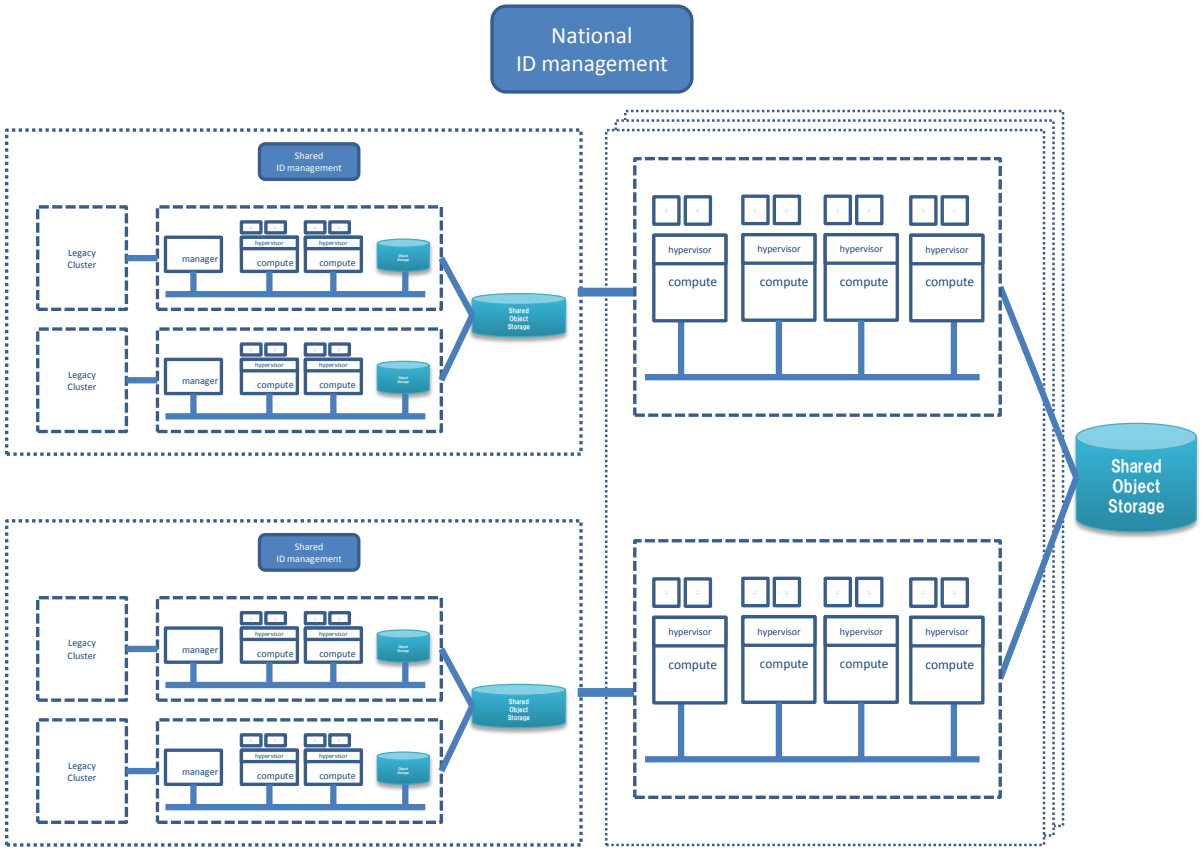


図 4-3-4　各アカデミッククラウド基盤とインタークラウド基盤の接続構成

このアプローチの利点は，最初はこの拡張領域としてインタークラウドハブなるものを中立的な機関が構築し，インタークラウド連携をすぐに始められることであり，しかもそれぞれのクラウド機能はそのままの形で実現できる点である．それが定着した後，同様の機能を各クラウド側で提供できるアーキテクチャとなっているため，それを採用し，インタークラウド基盤としてその一部を提供する機関が増えることでさらに，連携規模が順次増大する仕組みを持っている．つまり，図 4-3-4 ではコミュニティ全体が共有する IT 資産が各プライベートクラウドとは別に配置されているものだけで構成されるように表現されているが，さらに進んだ形態では各プライベートクラウドの一部の IT 資産を他のプライベートクラウドから使えるような P2P 型のアーキテクチャも比較的スムーズに取り入れることができる．

4.2　インタークラウド基盤の設計方針

提案するアプローチにより，第 2 章で述べたユースケースを満たすためのインタークラウド基盤に関する設計方針を整理する．

設計方針 1：クラウドへのリソース割り付けの柔軟性

クラウド毎に動的なリソース割り付けを地域的に分散して可能である必要がある．また，既存のクラスタと L2 接続できなければならない．さらに，これら機能がソフトウェアから制御可能な API を持つ必要がある．

設計方針 2：ベアメタルなマシンでのクラスタ構築

仮想マシンだけでなく Xen や KVM などの仮想化基盤ソフトウェアを前提とする IaaS 基盤を動的に構築するために，ベアメタルなクラスタが構築できる必要がある．これがソフトウェアから制御可能な API を持つ必要がある．しかも，適用ユースケースに合った性能でのクラスタ構築が必要になる．具体的にはユースケース　U1 に応えるために，物理サーバ数百台で構成される規模の IaaS の構築が 10 分以内に構築できる．この目標値を満たせばユースケース U2，U3，U4 についても実用的な運用が可能になると考えている．

設計方針 3：クラウド間の分離

物理リソースは共有するものの，以下の条件を守ったクラウド間分離が実現されている必要がある．この条件が整っていれば，クラウド間の干渉がなくあたかもそれぞれ独立なクラウドが，ユースケース U1，U2，U3，U4 を満たして存在することができる．

(1) クラウド間ではネットワークトラフィックが分離されている．
(2) クラウド間で同一 MAC アドレスや IP アドレスの仕様が可能である．
(3) クラウド間で同一 VLAN-ID が利用可能である．
(4) クラウド間での cpu や memory 使用に関する干渉がない．

設計方針 4：クラウド構築/運用の簡単化

クラウド基盤の構築が構成情報を与えれば構築及び構築後試験まで自動的に実行できる．この際構築時間は構築対象ノード数の増加に依存せず，物理サーバ数百台で構成される規模の IaaS を 10 分以内の構築を実現できる必要がある．

設計方針 5：地域分散型オブジェクトストレージ

クラウド間での連携を容易にするために各クラウドから共通に使える地域分散型のオブジェクトストレージが必要となる．これを使ってクラウド基盤上で動作するマシンイメージやアプリケーションデータの共有を行う．このマシンイメージの起動時間が実用的な範囲になるためには分散オブジェクトストレージから実行クラウド基盤へのマシンイメージ転送時間がオブジェクトストレージと同一データセンタ内にある場合と同程度でなくてはならない．

設計方針 6：外部認証の利用

学認などの外部認証連携基盤を活用して複数クラウドサービスの連携を容易にすることができる必要がある．

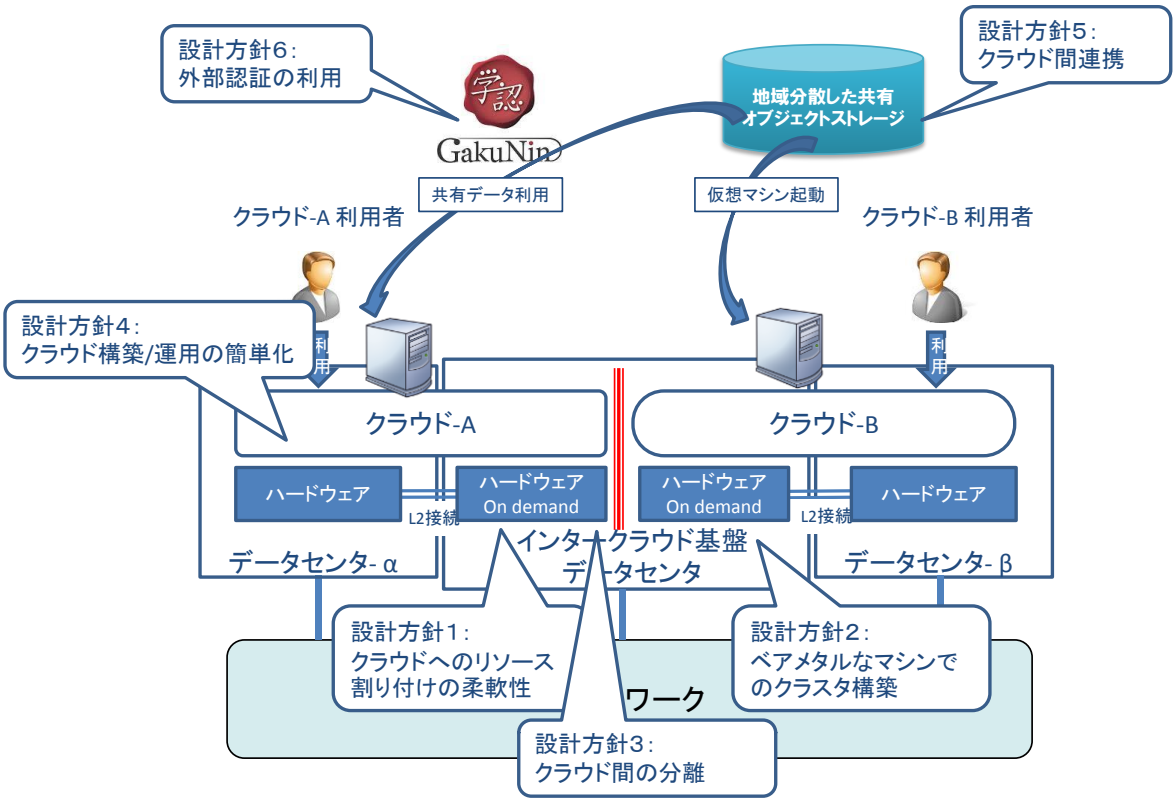


図 4-4　インタークラウド基盤の設計方針

表 4-1 にユースケースと設計方針の対応についてまとめる．

ユースケース 3 (U3)：災害や大規模故障発生に対する可用性保証とユースケース 4 (U4)：サービス継続についてはクラウド基盤自体の全体のマイグレーションになるため既存のクラスタとの L2 接続機能は必要としないユースケースとなる．これ以外のユースケースは全ての設計方針を必要としている．従って，対象とするユースケースを満足させるためにはこれらの設計方針は必須であることが分かる．逆に，これらの設計方針を満足すれば，それぞれのユースケースが実現できることは以下のように概観できるけれど，詳しくは第 6 章で述べるプロトタイピングを通じて検証できる．

表　4-1 ユースケースと設計方針

	R1	R2	R3	R4	R5	R6
U1	○	○	○	○	○	○
U2	○	○	○	○	○	○
U3	-	○	○	○	○	○
U4	-	○	○	○	○	○
U7	○	○	○	○	○	○

ユースケース１(U1)：急激な負荷増加に対する性能を保証
ユースケース２(U2)：遅延に対する性能保証
ユースケース３(U3)：災害や大規模故障発生に対する可用性保証
ユースケース４(U4)：サービス継続
ユースケース７(U7)：特殊リソースを用いた計算処理

設計方針１(R1)：クラウドへのリソース割り付けの柔軟性
設計方針２(R2)：ベアメタルなマシンでのクラスタ構築
設計方針３(R3)：クラウド間の分離
設計方針４(R4)：クラウド構築/運用の簡単化
設計方針５(R5)：地域分散型オブジェクトストレージ
設計方針６(R6)：外部認証の利用

4.2.1 急激な負荷増加に対する性能を保証

あるクラウド基盤で急激に負荷が増加した場合，R1, R2 を利用し，インタークラウド基盤データセンタに依頼して必要なベアメタルノードを確保し，それをクラスタ化し，自らのクラウド基盤を構成しているネットワークを L2 延伸する．この際，ベアメタルを獲得するためのインタークラウド基盤のサービスを利用する際に R6 を用いて学認 ID でサービス利用する．その後，R4 を利用し，既存のクラウド基盤を構成するノードと同一 L2 ネットワークに組み込まれたベアメタルノードに既存のクラウド基盤のコンピュートノード用ソフトウェアをデプロイメントし，それを既存クラウド基盤に組み込む．その拡張されたクラウド基盤上で仮想マシンを起動する際に利用するマシンイメージは既存クラウド基盤側にあるオブジェクトストレージに格納されていると，そちらから遠隔地にあるインタークラウド基盤データセンタ内にあるコンピュートノードで起動する場合，起動速度が遅くなる．R5 を利用して，マシンイメージを地域分散型オブジェクトストレージに格納しておけば，既存クラウド基盤側のコンピュートノードをそちら側にあるマシンイメージを使い，インタークラウド基盤データセンタ側のコンピュートノードはインタークラウド基盤データセンタ側にあるマシンイメージを起動するため，どちらの場合もローカルでの起動となり，起動速度が遅くなることはない．

4.2.2 遅延に対する性能保証遅延に対する性能保証

遠隔地に出張した場合などに普段の勤務地で使っている Desktop as a Service にアクセスすると画面転送がレイテンシの大きいネットワークを経由するために性能低下し，その使用感が大きく悪化することがある．この悪化を回避するために Desktop as a Service を出張先に近いインタークラウド基盤センタに延伸し，そちらで Desktop に対応する仮想マシンを起動することでレイテンシを小さく保つことで使用感を良好にする．このためには，R1, R2 を利用し，インタークラウド基盤データセンタに依頼して必要なベアメタルノードを確保し，それをクラスタ化し，自らのクラウド基盤を構成しているネットワークを L2 延伸する．この際，ベアメタルを獲得するためのインタークラウド基盤のサービスを利用する際に R6 を用いて学認 ID でサービス利用する．その後，R4 を利用し，既存のクラウド基盤を構成するノードと同一 L2 ネットワークに組み込まれたベアメタルノードに既存のクラウド基盤のコンピュートノード用ソフトウェアをデプロイメントし，それを既存クラウド基盤に組み込む．R5 を利用して，利用者の Desktop に対応

するマシンイメージスナップショットを地域分散型オブジェクトストレージに格納しておけば，新たに構築したクラウド基盤側で起動する際にも起動時間を高速にできる．なお，各利用者の出張の毎にクラウド基盤の延伸を行うのではなく，ある程度出張者の利用を見込んで予め遠隔地のクラウド基盤の構築を行っておくことができる．さらにはニーズの増減に合わせたサイズ調節もR1，R2 を利用して動的に実施できる．

4.2.3 災害や大規模故障発生に対する可用性保証

あるクラウド基盤が存在する地域に大規模な災害や事故が発生し，そのクラウド基盤が使えなくなった場合，そのクラウド基盤上のサービスを別の地域等で使用できる必要がある．この場合，まず定期的にクラウド基盤の管理情報（ユーザ情報，クラウド基盤構成情報，マシンイメージスナップショット，ボリュームスナップショットなど）について R5 を用いて地域分散型オブジェクトストレージに格納する．実際に，大規模な災害や事故が発生した段階で，R2 を利用し，インタークラウド基盤データセンタに依頼して必要なベアメタルノードを確保し，それをクラスタ化する．この際，ベアメタルを獲得するためのインタークラウド基盤のサービスを利用する際に R6 を用いて学認 ID でサービス利用する．その後，R4 を利用し，既存のクラウド基盤を新規に構築したクラスタ上に再現する．この時，R5 を用いて，定期的に蓄積してあったクラウド基盤構成情報を用いる．さらに再構築したクラウド基盤上に R5 を用いてクラウド基盤のユーザ管理情報やマシンイメージスナップショット，ボリュームスナップショットをリストアする．この状態でクラウド基盤ユーザに，再構築したクラウド基盤を開放することで利用者自ら各自のサービス再起動を行え，定期的に取ってあったスナップショットの最新版の状態でのサービス再開が可能となる．

管理情報を定期的に取る際の時間間隔は実際の管理情報サイズや重要度に合わせて個別に調整しなければならない．災害時でも動作させる必要のあるサービスを取捨選択することが重要になる．

4.2.4 サービス継続

あるクラウド基盤の事業者が何らかの事情でクラウド基盤サービスが出来なくなった場合，そのサービスの継続をインタークラウド基盤センタのリソースを使って一時的にでも実施可能となる．まずサービス停止の直前の状態をクラウド基盤の管理情報（ユーザ情報，クラウド基盤構成情報，マシンイメージスナップショット，ボリュームスナップショットなど）について R5 を用いて地域分散型オブジェクトストレージに格納する．R2 を利用し，インタークラウド基盤データセンタに依頼して必要なベアメタルノードを確保し，それをクラスタ化する．この際，ベアメタルを獲得するためのインタークラウド基盤のサービスを利用する際に R6 を用いて学認 ID でサービス利用する．その後，R4 を利用し，既存のクラウド基盤を新規に構築したクラスタ上に再現する．この時，R5 を用いて，定期的に蓄積してあったクラウド基盤構成情報を用いる．さらに再構築したクラウド基盤上に R5 を用いてクラウド基盤のユーザ管理情報やマシンイメージスナップショット，ボリュームスナップショットをリストアする．この状態でクラウド基盤ユーザに，再構築したクラウド基盤を開放することで利用者自ら各自のサービス再起動を行え，事業者のサービス停止直前に取ってあったスナップショットの最新版の状態でのサービス再開が可能となる．

4.2.5 特殊リソースを用いた計算処理

スーパーコンピューターやビッグデータなどアカデミックコミュニティの共有財産となる特殊リソースでかつ場所の移動が困難なものがある．この際，U2 では人がそうであったように，その特殊リソースにクラウド基盤が近づくことでレイテンシ増加に起因する計算処理性能低下を防ぐことが出来る．このためには，R1，R2 を利用し，特殊リソース近くのインタークラウド基盤データセンタに依頼して必要なベアメタルノードを確保し，それをクラスタ化し，自らのクラウド基盤を構成しているネットワークを L2 延伸する．この際，ベアメタルを獲得するためのインタークラウド基盤のサービスを利用する際に R6 を用いて学認 ID でサービス利用する．その後，R4 を利用し，既存のクラウド基盤を構成するノードと同一 L2 ネットワークに組み込まれたベアメタルノードに既存のクラウド基盤のコンピュートノード用ソフトウェアをデプロイメントし，それを既存クラウド基盤に組み込む．R5 を利用して，利用者の計算処理に必要なマシンイメージスナップショットを地域分散型オブジェクトストレージに格納しておけば，新たに構築したクラウド基盤側で起動する際にも起動時間を高速にできる．なお，各利用者の計算処理が必要になる毎にクラウド基盤の延伸を行うのではなく，利用を見込んで予め遠隔地のクラウド基盤の構築を行っておくことができる．さらにはニーズの増減に合わせたサイズ調節も R1，R2 を利用して動的に実施できる．

第5章　アーキテクチャ提案 (Ameba Cloud Architecture)

第 4 章で述べたクラウドの動的配備の設計方針を満たすアカデミッククラウドアーキテクチャとして，筆者らは設計方針 1,2,3,4 に従って，既存のクラスタ環境と直結するプライベートクラウドを動的に構築できる Elastic Private Cloud（図 5-1 参照）を Cluster as a Service により実現できるコンピュート 部分と設計方針 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レージサービスから構成されるものを考案した．さらに設計方針 6 に沿って，これら二つのサービス統合を外部認証である学認認証により実現するアーキテクチャを採用した．柔軟に広域ネットワークを使って延伸するクラウドアーキテクチャであることから筆者らはこのアーキテクチャのことを Ameba Cloud Architecture(ACA)と呼んでいる．

基本的なアイデアは物理マシンから構成されるクラスタ上に動的にクラウド基盤を構成し，それが既存のクラスタと L2 接続できれば，異種のクラウドとの接続を考慮する必要が無く，ただ必要に応じて自分のいつも使っているクラウド基盤を拡張するのに必要な物理マシンを物理マシンプールから借り出し，その上にクラウド基盤拡張部を構成すれば良い．この物理マシンプールは SINET の様な広域網の先にあっても，L2 レベルの接続が出来れば良い．これは SINET 等がすでに持っている L2VPN 機能や VPLS 機能を活用することで実現できる．その際各サイトのプライベートクラウドと SINET との接続ポイントには OpenFlow Switch/OpenFlowContoroller が配置されていて，それらにより閉域網を各クラスタ向けに構成する．ここで OpenFlow 技術によりクラスタを分離するアーキテクチャとしたのは，各クラスタの上に構築されるクラウド基盤側で VLAN 技術によりテナントごとのネットワークを分離することが一般的であり，それと重複してクラウド構築時に VLAN 技術を利用することができないためである．

このアーキテクチャを取ることで，たとえクラウド型アプリケーションであっても自分の実行場所を広げことが出来る．「1.2.1 アカデミックコミュニティクラウドの現状」で示した問題１が解決できる．さらに，アプリケーション実行のためにリモートで起動する必要のあるマシンイメージも地理的に分散したインタークラウドストレージに保存しておくことであたかもその起動場所ローカルにスナップショットが保存してあったのと同じ起動時間で利用できる．すなわち問題 2 も解決できる．

本論文では次に述べる三つの提案に基づくアーキテクチャにより実際に，これらの課題が解決できることを示す．

提案１：　クラウド型アプリケーションの実行環境を動的に構成するアーキテクチャの提案（CaaS: Cluster as a Service）

ベアメタルマシンのクラスタを地理的に離れたデータセンタを跨っても構成するモジュールおよび構成されたクラスタ上に全自動で実行環境を構築するモジュールから構成される．

提案２：　遠隔地でも高速にクラウド型アプリケーションを起動する手法の提案（インタークラウドオブジェクトストレージ）

マシンイメージを分散配置し，それらを高速に配備する手法について提案する．

提案３：　認証連携によるサービス統合アーキテクチャの提案　（学認連携）

構成サービスを学認対応することで，それらのサービスをシームレスに連携する．

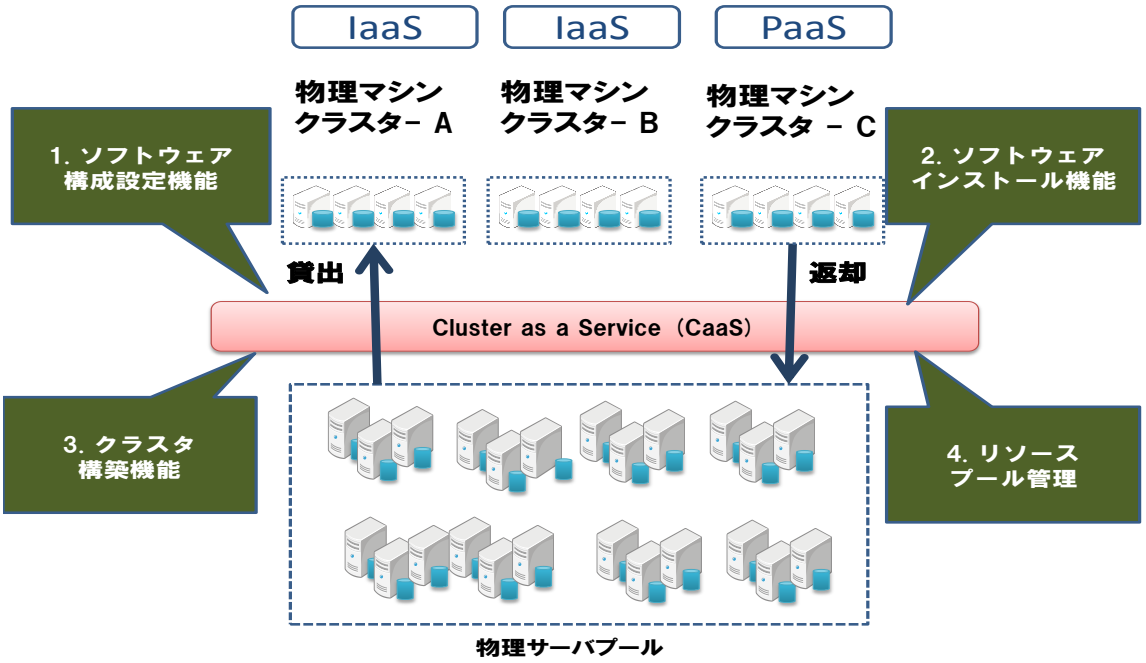


図 5-1　Elastic Private Cloud

以下，各提案について順次説明する．

5.1　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一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 Clusteras a Service（以下 CaaS）と呼ぶ． 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 クラスタ等の 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 5-2 に示す．

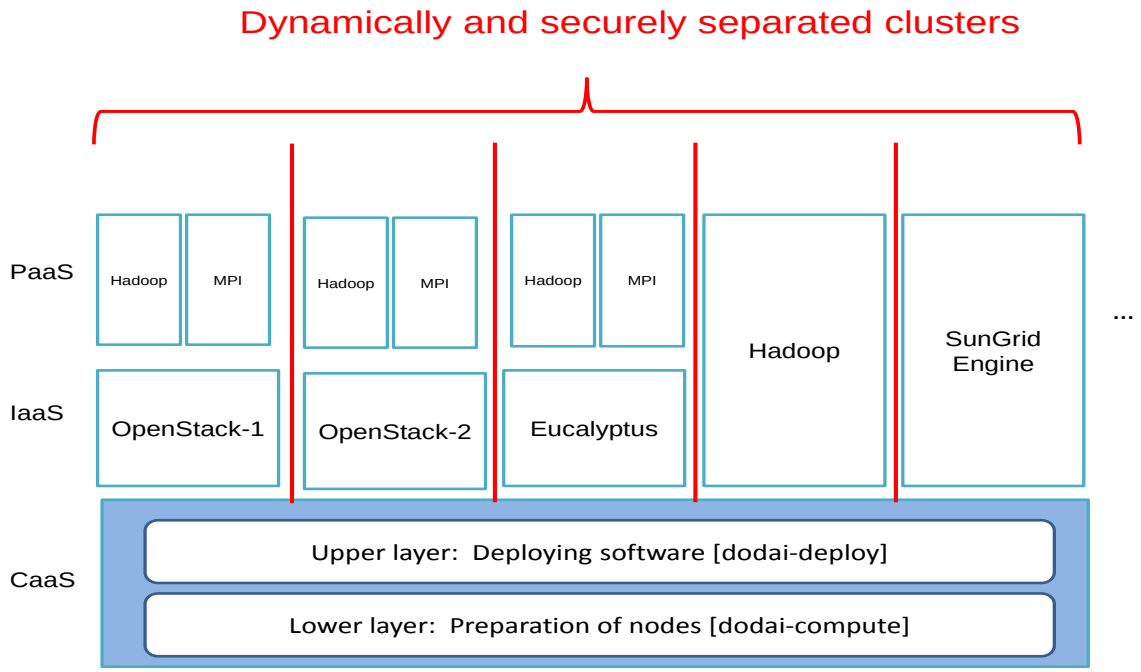


図 5-2　CaaS の位置づけ

CaaS 層の一つの実装としては図 5-2 に示すように現在上下二層から構成されているクラスタ管理フレームワーク dodai[23], [24] 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 から生成される一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 とみなし，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． 従って，dodai-deploy は図 5-3 に示すように通常の IaaS に対しても適用可能であり，例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．

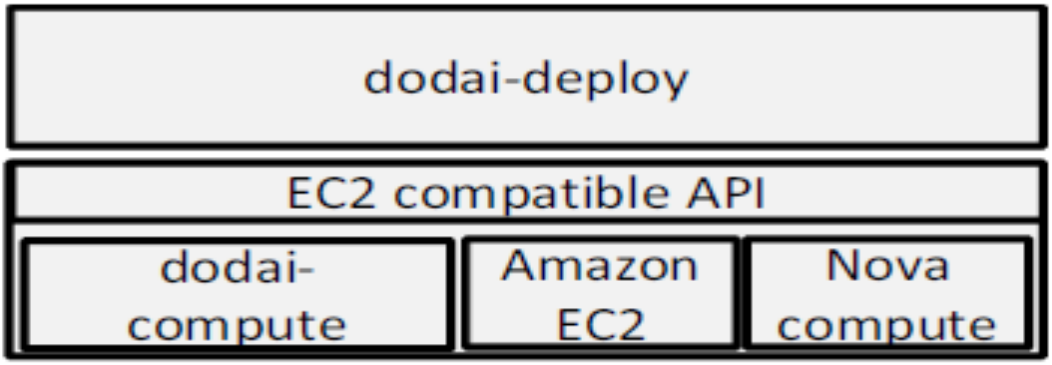


図 5-3　Dodai-deploy 活用方法

図 5-4 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 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 コントローラとスイッチを利用する．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，設計方針 1，2，3 を同時に満たすことができる．

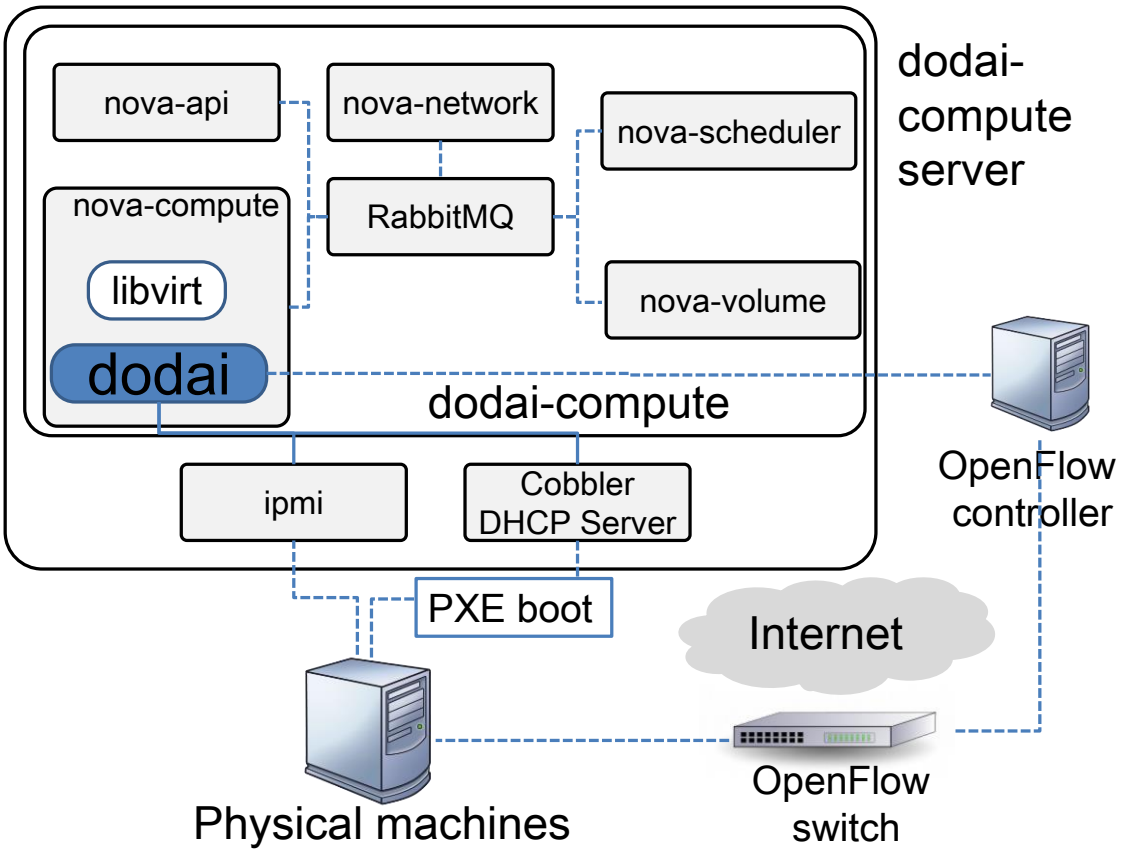


図 5-4　Dodai-compute のアーキテクチャ

(2) Dodai-deploy

以下が dodai-depoly 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通りである．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[25])用の設定ファイルを生成し，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 [26]) 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 5-5 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設計方針 4 を満たすことができる.

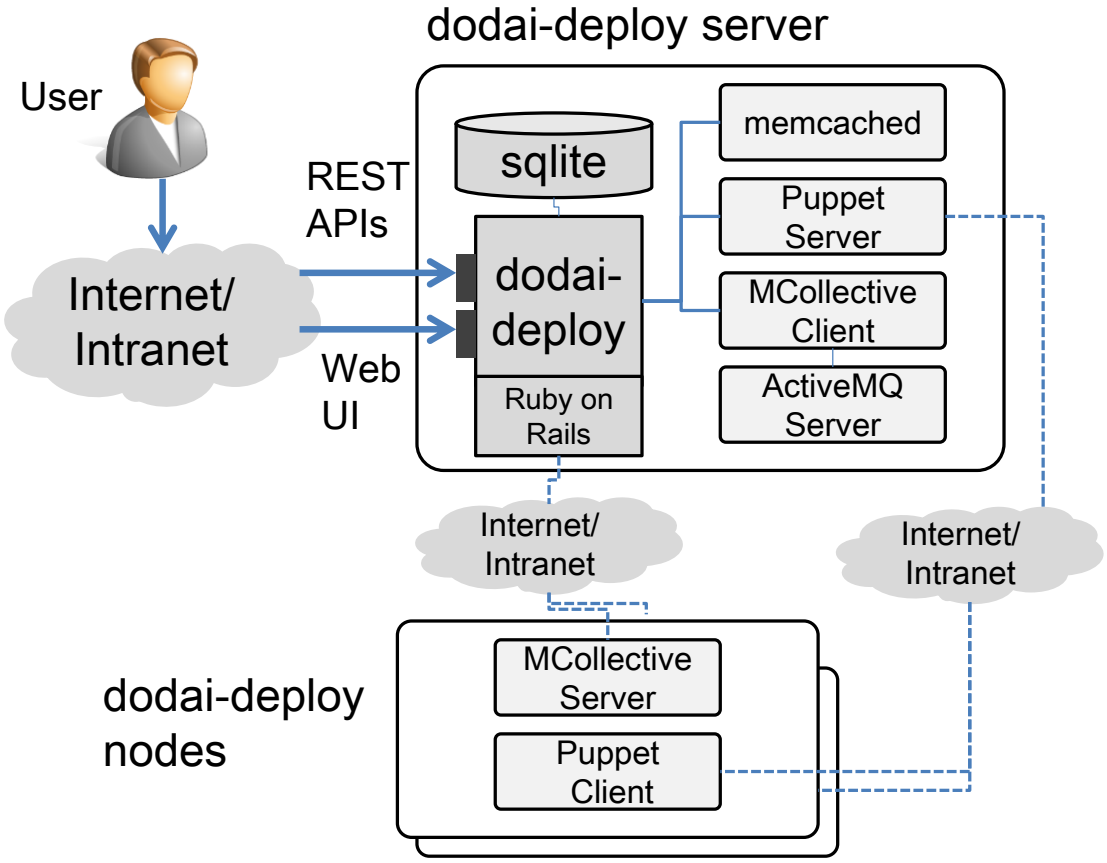


図 5-5 Dodai-deploy のアーキテクチャ

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5-1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り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三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．つまりElastic Private Cloudが，我々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設計方針 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．

	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	○	○	○	○

表5-1　既存クラウドアーキテクチャとElastic Private Cloudの比較

Dodai-compute(OpenStack (Bare metal) + OpenFlow[27]　の組み合わせ)で設計方針 1-3，5を満たし，dodai-deployと合わせることで設計方針4を満たす．学認連携により設計方針6を満たす．

5.2　インタークラウドオブジェクトストレージサービス

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レージ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レージのように見えるインタークラウドオブジェクトストレージサービス（図5-6参照）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（設計方針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．

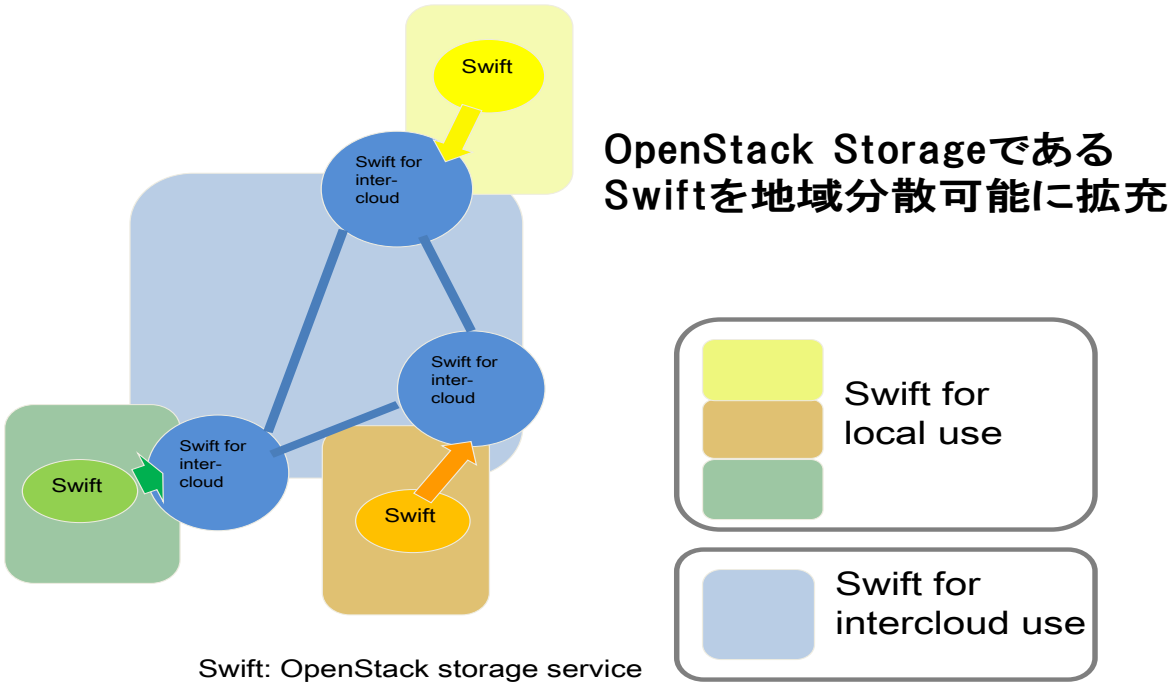


図 5-6-1　インタークラウドオブジェクトストレージサービス

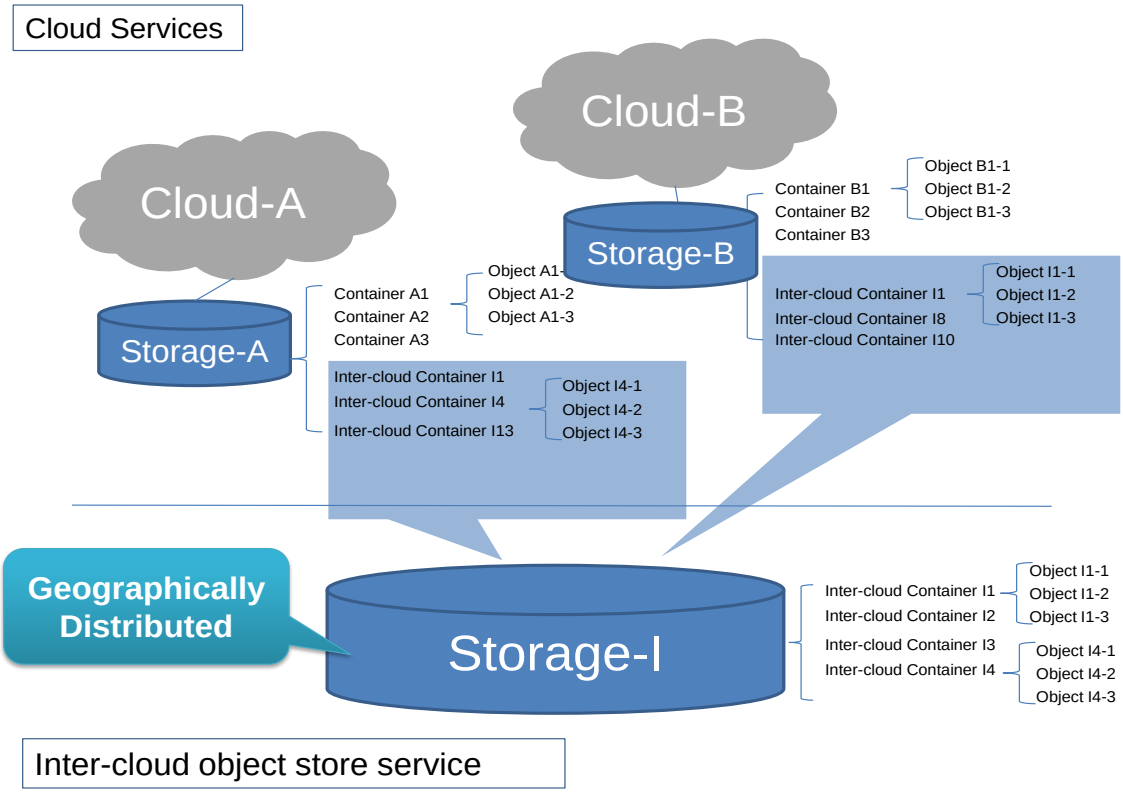


図 5-6-2　インタークラウドオブジェクトストレージサービス

インタークラウドオブジェクトストレージサービスの実装として筆者らは OpenStack [28]オブジェクトストレージサービス swift を広域分散向けに改変した colony を採用している.

dodai と colony を図 5-7 のように組み合わせることアーキテクチャとすることで 1.2 で述べた二つの課題を解決することができる.

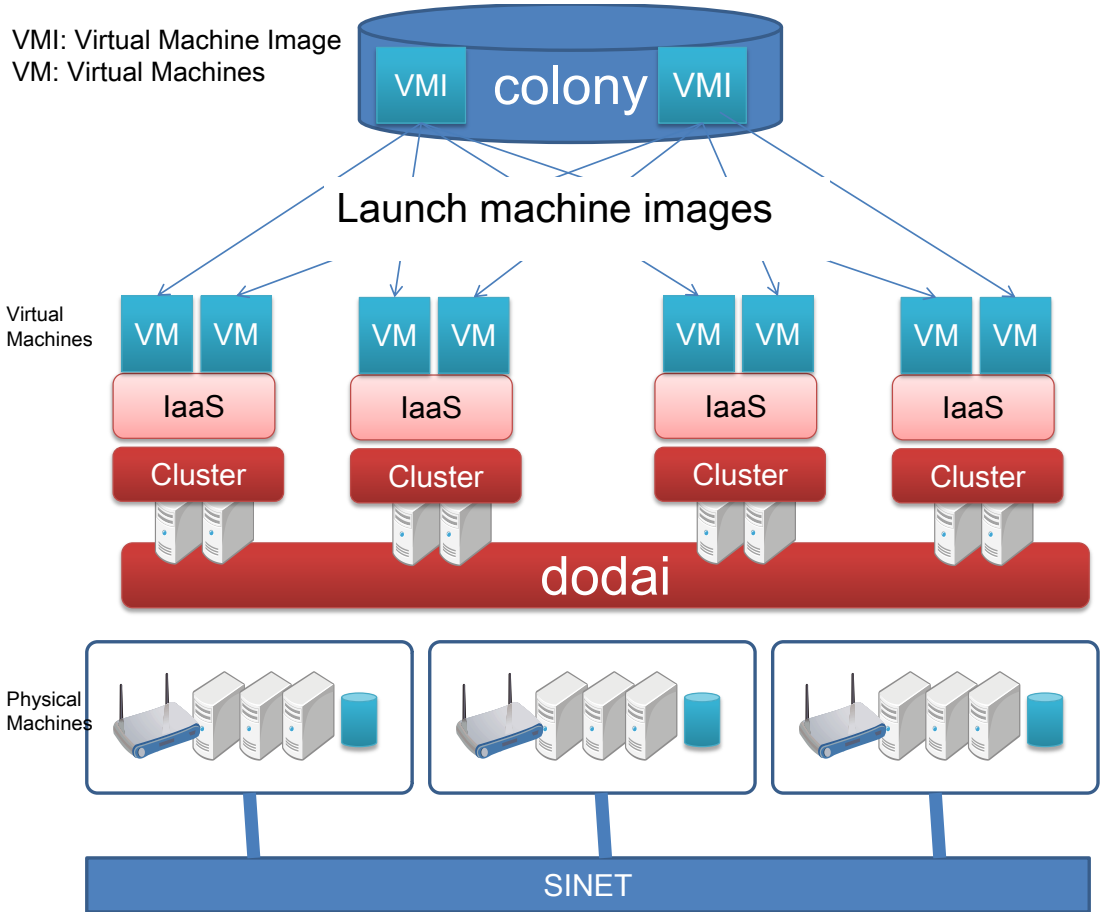


図 5-7 クラウド連携アーキテクチャ

5.3 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，設計方針 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出来た. 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プロトタイピングによる評価を実施したので，その評価結果について第 6 章に述べる.

第6章　プロトタイピングと評価

提案アーキテクチャにより第 4 章で述べた設計方針を満足するクラウド連携基盤が実現でき，従来のクラウド連携のアプローチが持っていた課題を克服できることを示すために，筆者らは複数のプロトタイプシステムの構築を行い，それらを用いて評価検証実験を実施した．

6.1　設計方針，解決策，検証の関係

6.1.1　設計方針

第 4 章で述べた以下の設計方針は，アカデミックコミュニティクラウドのクラウド連携を活用して実現を目指す 5 つのユースケースのために有効な設計方針であることを示した．

設計方針 1：クラウドへのリソース割り付けの柔軟性
設計方針 2：ベアメタルなマシンでのクラスタ構築
設計方針 3：クラウド間の分離
設計方針 4：クラウド構築/運用の簡単化
設計方針 5：地域分散型オブジェクトストレージ
設計方針 6：外部認証の利用

これらの設計方針をシナリオとしてまとめると，以下のシナリオが地域に分散した構成で達成可能であるということになる．

6.1.2　シナリオ

１．　ベアメタルマシンを閉域ネットワークに組み入れたクラスタを 10 秒以内で動的に構成する．この際，クラスタ上にデプロイするクラウド基盤間では同一 IP アドレスや VLAN-ID が利用される可能性があることへの対処が必要である．

２．　動的に構築したクラスタ上にクラウド基盤をオンデマンドでデプロイする．この際，デプロイ時間はクラスタのサイズに依存せずスケーラブルであり 5 分以内に完了する．また，オンデマンドで構成する場合，その都度にクラウド基盤のソフトウェアライセンスについての解決を行うのではなく，予め解決しておく．

３．　クラウド基盤から利用するマシンイメージやデータへのアクセスが同一データセンタ内にそれらが存在するのと同等の速度で行なえる．

これを達成するために以下の解決策を採用した．

４．　これらの操作を外部認証を利用して実施できる．

6.1.3　解決策

１．既存クラスタとの接続性の高いベアメタルクラウドサービス

IaaS 向けオープンソースクラウド基盤で仮想化層(Hypervisor) を使って仮想マシン起動を実施している部分を改造し，標準インタフェースのみを用いて，物理マシンの管理・制御が行う．この際，クラスタに組み入れられていない物理マシンについてはプール化し，このプール内でその物理マシンの再利用までの間マシンの自動メンテナンスを行った後，デフォルト OS での立ち上げまで実施しておく．このことで，再利用の際にはプールに対応した OpenFlow 閉域網から割り付け対象のクラスタの対応した閉域網への所属替えを OpenFlow controller より実施することにより，OS などの立ち上げ時間の削減を実現する．

また，ネットワーク分離のために OpenFlow 技術を用い，クラウド基盤間では同一 IP アドレスや VLAN-ID が利用できるようにした．

さらに既存クラスタとの接続は既存クラスタが所属する VLAN とベアメタルクラウド内のクラスタに対応する OpenFlow 閉域網を OpenFlow Switch でマッピングすることにより実現する．この LAN 内用のアーキテクチャは SINET の L2VPN/VPLS 機能を用いることで広域網経由の場合にも適用できる．すなわち遠隔地にある既存クラスタの SINET 接続点と対応した VLAN-ID を OpenFlow controller が対応するベアメタルクラウド内のクラスタにマッピングすることで遠隔地の既存クラスタとベアメタルクラウド内のクラスタが L2 接続される．このサービスを学認認証により利用可能にする．

２．スケーラブルな自動デプロイサービス

デプロイするクラウド基盤の構造より事前に依存関係を抽出しておくことにより，デプロイ処理で並列動作可能な部分について最大限並列動作させる仕組みを MCollective と Puppet により実装し，スケーラビリティを確保する．このサービスを学認認証により利用可能にする．

３．地域分散オブジェクトストレージサービス

広域ネットワーク上の位置情報を認識するようオープンソースオブジェクトストアシステムの，オブジェクト配置メカニズムを改善することとキャッシュメカニズムを導入することでクラウド基盤から利用するマシンイメージやデータへのアクセスの高速性を達成する．このサービスを学認認証により利用可能にする．

４．クラウド基盤を各所に動的にデプロイできる前提条件として，ライセンス的な扱いに問題を持たないオープンソースクラウド基盤をデプロイメント対象とした．

6.1.4 検証

これらの解決策が有効であることを以下のプロトタイピングで検証した．

1. 教育クラウド　edubase Cloud
   品質向上と監視強化を実施することで，前提条件であるオープンソースクラウド基盤で実運用可能なクラウド基盤が構成できることを検証した．
2. 研究クラウド　gunnii/dodai
   一つの機関向けに既存クラスタとの接続性の高いベアメタルクラウドサービスが上記の解決策で実現できることを検証した．
3. 地域分散オブジェクトストレージ　colony
   クラウド間連携で重要となる分散オブジェクトストレージが実現可能であること．さらに，マシンイメージの起動時間が実用的な範囲になるためには分散オブジェクトストレージから実行クラウド基盤へのマシンイメージ転送時間がオブジェクトストレージと同一データセンタ内にある場合と同程度であることを検証した．
4. インタークラウド基盤実験システム
   SINET と学認を活用することで，研究クラウドで採用したアーキテクチャが一つのデータセンタ内だけではなく，複数のデータセンタをまたがって適用可能であることを検証した．

6.2 教育クラウド edubase Cloud （検証 1）

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[29]-[32]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，各大学等で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出来なければならない．（要件3）
・ 試してみた環境を保存し，後で利用出来る必要がある．

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他に，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．

6.2.1 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．一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，効率よ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，ドキュメントの他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よく進めるために，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．

6.2.2 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6-1 にedubase Cloudの全体構成を示す．

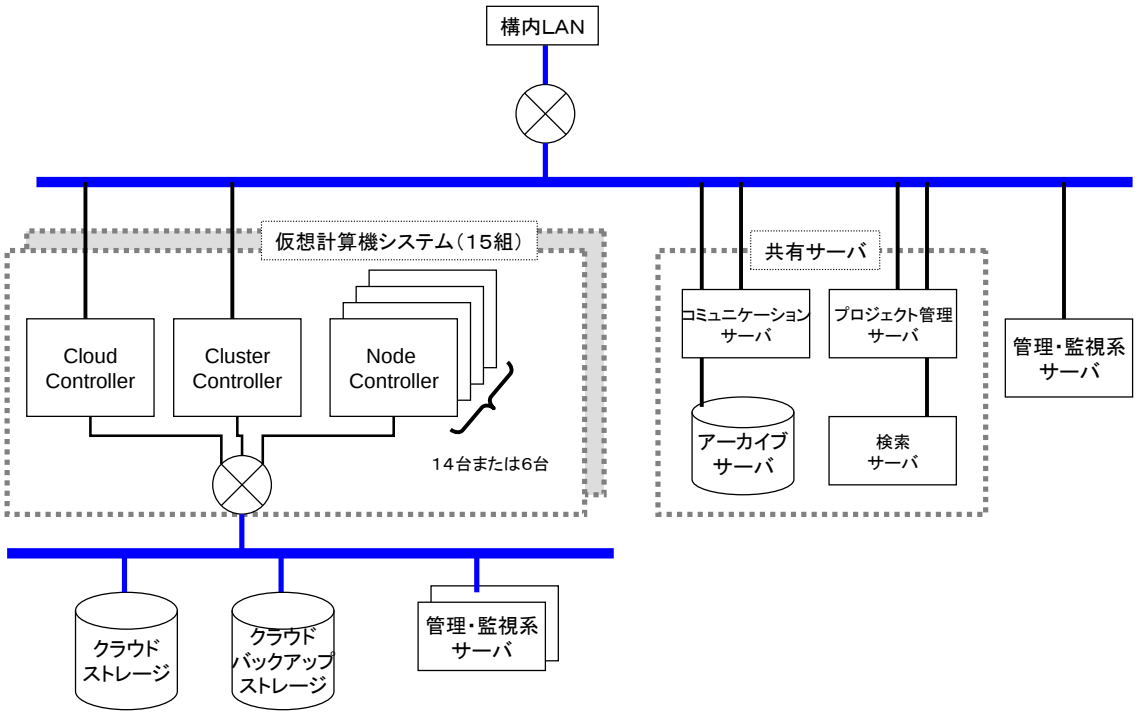


図6-1 edubase Cloudの全体構成

(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取ることとした．ミニクラウドは 2 種類の構成を取り，用途に応じて使い分けられる様にした．表 6-1 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．Eucalyptus のアーキテクチャに従って，仮想マシンの割り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出来る設定としたため，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．

従って，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.

表6-１　マルチクラウドの構成概要

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

(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 40TB 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.

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来ている. 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.

実際のPBLに適用した際のedubase Cloudの評価については[51] にある通りケーススタディを通じた評価により以下の要求により設計が妥当であったことが分かる.

6.2.3　edubase Cloud の運用を通じて得られた要求条件

ケーススタディでは顕在化しないレベルの共有が教育クラウドの約2年間(2010.5-2012.3)の運用から抽出された.

(6.2-1) 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）一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一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取らざるを得なかった．

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．

(6.2-2) クラウド初期化ツールの汎用化

教育クラウドの一つの特徴である専有性と改変性をいかして，例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，引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言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．

(6.2-3) 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だせるけれど，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都度関わるオーバヘッドが問題となった．従って，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．

(6.2-4) 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．

(6.2-5) 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．

(6.2-6) 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か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.

6.2.4 評価実験

教育用クラウドの持つべき重要な特性として専有性, 改変性, 連携性, 保存性の４つを上げた. edubase Cloudがこれらの特性を持つことを評価実験により示す. 本評価では, 4つの特性について, 主に以下の観点で評価実験を実施した.

- 試作したedubase Cloudが, 実際に4つの特性を持つか.
- 基盤技術者の教育のために, ４つの特性が役に立つか.

6.2.4.1 評価実験の概要

プロジェクトA：クラウド基盤ソフトウェアを改造し仮想マシン起動処理の高速化を行う.

プロジェクトB：大規模分散モデル検証アルゴリズム評価のためベンチマークを行う. edubase Cloudで扱えない大規模なモデル検証は, 商用クラウドでベンチマークを行うために移行する.

表6-2 実証評価プロジェクトの主な評価ポイント

	専有性	改変性	連携性	保存性
プロジェクトA	◎	◎	ー	○
プロジェクトB	◎	ー	◎	○

◎：主要な評価項目, ○：評価項目, ー：評価対象外

6.2.4.2プロジェクトA

6.2.4.2.1 概要

プロジェクトAのテーマとしては, クラウド基盤の改変, クラウド基盤連携を取り上げた. プロジェクトAの概要を以下に示す. 基盤の改造を行う必要があるため, 利用する計算機資源としてはミニクラウド（大）を一式占有した. 改変のベースとしてedubase Cloudの基盤であるEucalyptusをそのまま利用し, 高速化に必要な部分を改造することとした.

(1) 試行評価の条件
プロジェクトメンバ（被験者）：ＳＥ経験７年のIT技術者
保有スキル：クラウド基盤であるEucalyptusの内部構造の知識
プロジェクト期間：平成22年2月から3月
プロジェクト内容：クラウド基盤上の仮想マシンの起動高速化（実装および性能確認）
利用リソース：ミニクラウド（１式）を専有
(2)プロジェクト実施内容
1. 仮想マシン起動処理のボトルネックの解析
2. 高速化方法の検討および実装
3. クラウド上での性能測定
4. 結果を考察としてまとめる
5. プロジェクト報告会およびデモ開催

6.2.4.2.2 実施結果

(1)クラウドのボトルネックの解析

起動のボトルネックの解析のため, 以下の作業を実施した.

- Eucalyptusのドキュメントの確認
- Eucalyptusのソースコードの解析
- Eucalyptusを動作させてのプロファイリング

図6-2　ボトルネックの測定

結果として図6-2 に示すような測定結果が得られた. 揮発性ディスクの作成およびSWAP領域作成の多くの時間がかかっていることがこのステップで判明した.

(2) 高速化方法の検討および実装

ステップ1で解析したEucalyptusの起動手順を基にして, 高速化方法の検討を実施した. 図6-3 にその概要を示す.

方針として, 起動に必要な処理のうち予め実行可能なステップについては事前に実行しておくことで高速化をすることとした. 具体的には, 以下の4ステップを事前に実行してプールしておくこととした.

- イメージのダウンロード
- イメージのコピー
- 揮発性ディスクの作成
- SWAP領域の作成

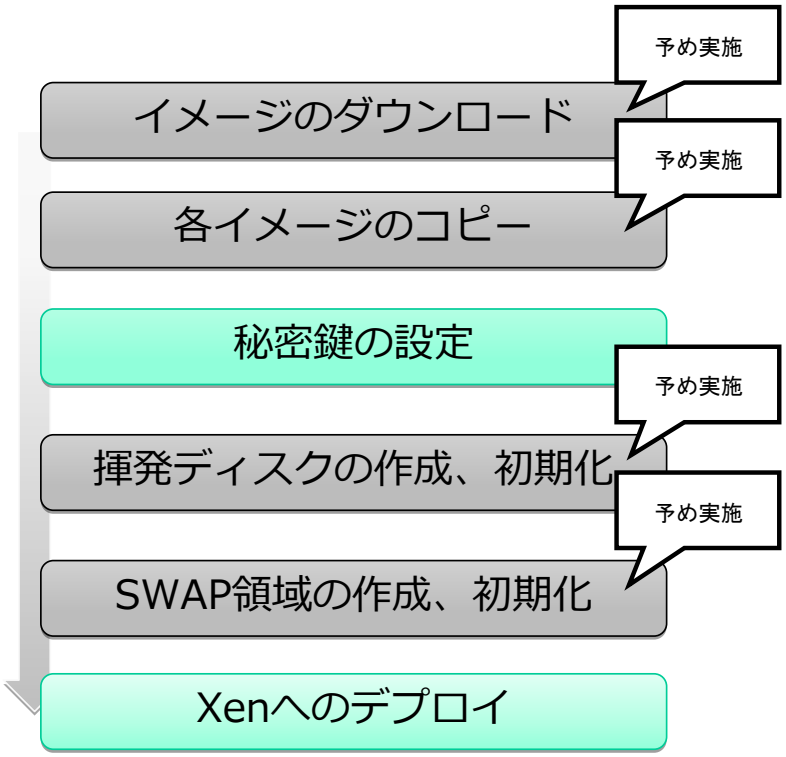


図6-3　起動プロセスと高速化のための手法の検討

実際の処理はNCで行われるため，該当するNCのコードを書き換えた．NCに予めキャッシュが生成されている場合にはキャッシュを基にして高速起動を行う．

(3) クラウド上での性能測定

ミニクラウド１式を占有し，起動高速化の性能測定を実施した．図6-4 に代表的な測定結果を示す．

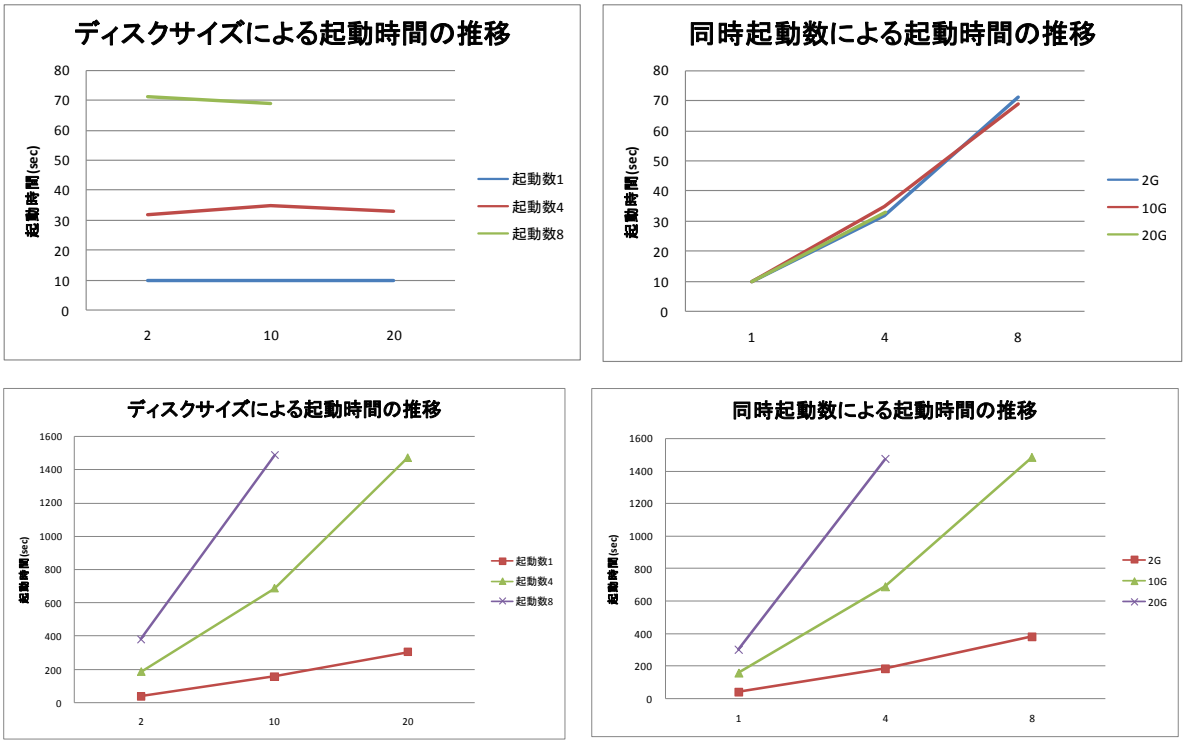


図6-4　起動高速化の測定結果

上段：高速化前の測定結果

下段：高速化後の測定結果

高速化実施前においては，利用するディスクサイズ，および同時に起動する仮想マシンの数に比例して起動時間がかかっていることが分かる．これは仮想マシンイメージをコピーするための処理時間である．

これに対して，高速化後はディスクサイズによらずに一定時間で仮想マシンを起動することができている．また，仮想マシンの同時起動数に対して起動時間はより小さな係数となっており，高速か実施前に比べ20倍程度の高速化が可能になったことが分かる．

6.2.4.2.3 プロジェクトAの評価

プロジェクトAでは，edubase Cloudの基盤ソフトウェアを改変して，起動処理の高速化を行った．また，改変結果の性能を測定のために，ミニクラウドを1台占有した．edubase Cloudの持つ改変性および専有性が生かされた事例と判断される．

クラウド上でソフトウェアのボトルネックを測定し，その結果を反映して改変を行うためには，自由に改変可能なクラウド基盤を準備する必要がある．通常の方法ではハードウエアを調達し，その上に基盤ソフトを整えた上でプロジェクトを開始することになる．このプロジェクト開始までに通常１か月以上要すると考えられる．従って，プロジェクトAにおいて2ヶ月という短期間に基盤を準備し，実験を行うことができたのはedubase Cloudを利用したことによりプロジェクト実施を2/3以下にできたと考えられる．

6.2.4.3　プロジェクトB

6.2.4.3.1 概要

プロジェクトBのテーマとしては，分散モデル検査の実施をedubase Cloudおよび商用クラウドで実施することを取り上げた．分散モデル検査のアルゴリズムや実行条件を変更してベンチマークが容易に行えるように，クラスタ構築用のスクリプトを用意した．また，商用クラウドとして，Amazon Web Services(AWS)を対象とすることとした．

プロジェクトBの概要を以下に示す．

6.2.4.3,2 実施結果

(1) 分散モデル検査ベンチマーク方法の検討

モデル検査を実施する場合，組み合わせの爆発が生じる場合が多く，分散モデル検査に対する期待が高い．分散モデル検査アルゴリズムの研究を行うプロジェクトを想定して，分散モデル検査のベンチマークを行う方法をプロジェクトとして検討することとした．

複数のアルゴリズム間の関係やモデルとの相関，利用する計算機資源との関係などを研究するためには，仮想計算機環境を利用する方法の効率が良いと考えられる．そこで，本プロジェクトでは，edubase Cloud上で直ぐに実施可能な仮想マシンとして分散モデル検査の実行環境を用意し，アルゴリズムのベンチマークが手軽に実施できる環境を用意することとした．

また，大規模な分散モデル検査を実施するために，仮想マシン環境を変換することで商用仮想計算機環境であるAmazonでも実行できるようにすることとした．

(1) 試行評価の条件
プロジェクトメンバ（被験者）：ＳＥ経験5年のIT技術者
保有スキル：モデル検査の基礎知識，パブリッククラウド(Amazon)，edubase Cloudの利用知識
プロジェクト期間：平成22年１月から3月
プロジェクト内容：大規模分散モデル検証アルゴリズム評価のためベンチマーク
利用リソース：ミニクラウド（１式）を専有
(2) プロジェクト実施内容
1. 分散モデル検査ベンチマーク方法の検討
2. 分散モデル検査ベンチマーク用仮想マシンイメージの作成
3. edubase Cloudを使った分散モデル検査ベンチマークの実施
4. 商用クラウドを使った分散モデル検査ベンチマークの実施
5. プロジェクト報告会およびデモ開催

(2) 分散モデル検査ベンチマーク用仮想マシンイメージの作成

分散モデル検査の基盤として，最も一般的なDiVinE Clusterを採用することとした．DiVinE ClusterはチェコのMasaryk UniversITyのParaDiSe Labsが開発したオープンソースソフトである．

通常のクラスタ環境では利用するマシン台数やＩＰアドレスなどを指定し，環境設定を設定した上でクラスタを利用する．edubase Cloudにおいてはクラスタ内のマシン台数やメモリ容量などのマシンスペックを自由に決めることができる．ベンチマークのパラメータとして設定する可能性が高いことから，仮想マシン内にMPIクラスタ起動用のスクリプトを用意することとした．本手順で最も注意した点は，仮想マシンの認証や仮想マシン同士の通信に利用するＩＤ情報や秘密鍵などの扱いである．商用環境での利用も想定されていたため，セキュリティに関して注意して設計を実施した．

(3) edubase Cloudを使った分散モデル検査ベンチマークの実施

前述の分散モデル検査ベンチマーク用仮想マシンイメージを利用してDiVinE Clusterをedubase Cloud上で実行し，64台の仮想マシンでの分散処理までの性能評価を実施した．

図6-5 は，利用した仮想マシン台数および実行時間の関係を，アルゴリズムごとにプロットした例である．クラスタ起動をスクリプト化することで，このような測定が自動的に実施できるようになり，大幅な手間の削減につながった．また，ベンチマーク実行にあたっては，クラウドの一つを占有することで外乱の少ない測定を行うことができた．

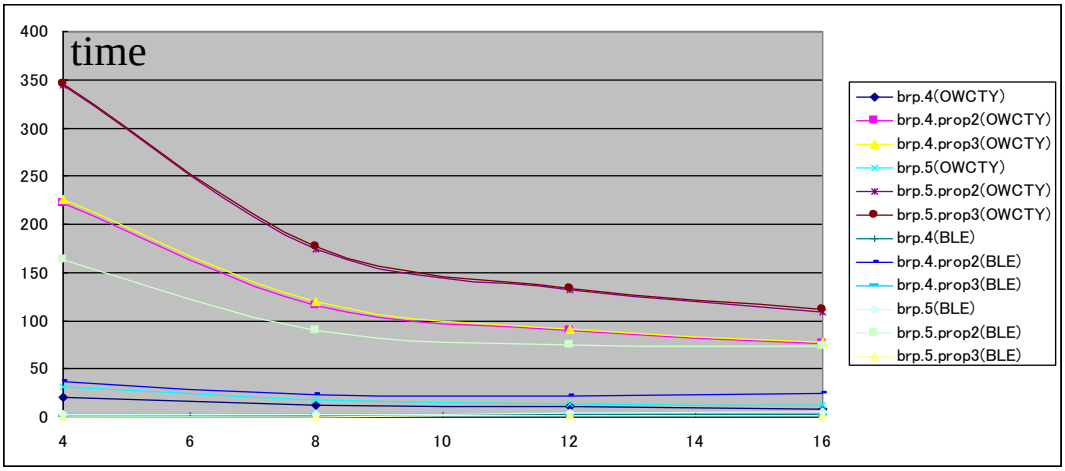


図6-5　edubase Cloud上での分散モデル検査の実行例

(4) 商用クラウドを使ったモデル検査ベンチマークの実施

edubase Cloudの内で扱えるマシン数より大規模な分散モデル検査をAWSで実行した．分散モデル検査ベンチマーク用仮想マシンイメージをAWS用に変換後，AWSに登録したうえで仮想マシンイメージを実行した．AWS上でもクラスタ起動用のスクリプトをそのまま利用することとした．図6-6は128台までの分散モデル検査ベンチマークをAWS上で実行した例である．

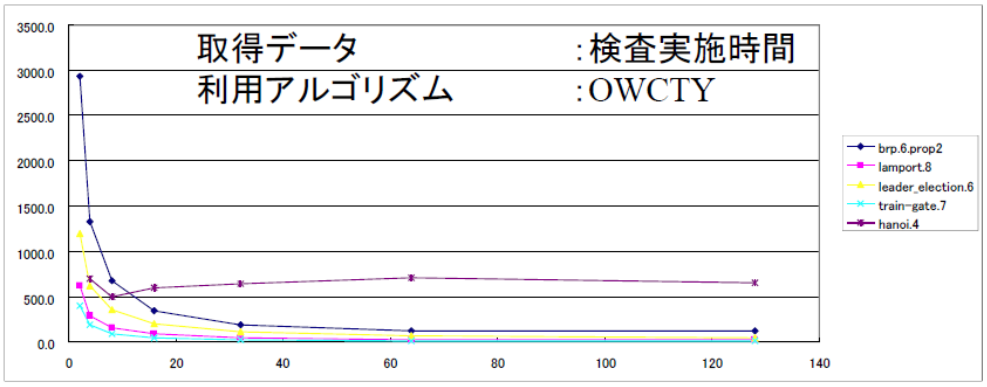


図6-6　AWS上での分散モデル検査の実行例

edubase Cloudの基盤であるEucalyptusはAWS互換のインタフェースを持つことを特徴の一つとしているが，仮想マシンイメージの形式まで同じではない．仮想マシンの管理方法や暗号化の方法なども異なるため，変換処理が必要になった．基盤にEucalyptusを採用したことで，AWSとの互換性が高く結果として軽微な変更のみで移行が可能であった．このことから，edubase Cloudは商用クラウドとの連携性が高いと考えられる．

6.2.4.3.3 プロジェクトBの評価

プロジェクトBの実施結果から，AWSとの連携性が示されたと考える．これは，基盤で利用したEucalyptusとAWSとの親和性が高いことと，今回用意したようなクラスタ構築用スクリプトによる

ことが大きい．さらには，edubase Cloud側で用意したマシンイメージの変換ツールを活用することでこの連携プロジェクトがスムーズに実施できた．
今後，教育目的で異なるクラウドの連携が必要になることが考えられるが，インタフェースが開示されているオープンなクラウド同士の仮想マシンイメージレベルおよびそれを用いたアプリケーションの連携は比較的容易であると考えられる．

6.2.5 検証１に関するまとめ

実践的なPBLに必要な教育クラウドの要求条件を分析し，実現のためのアーキテクチャとその機能を提案した．実際に200台の計算機からなる教育クラウドを構築した．

また，本システム上でクラウド基盤の改変を行う実証評価を実施した．結果として，本システムを利用することで，クラウド基盤の改変までが実際に行えることを示した．例として実施した仮想マシンの起動の高速化に着目した基盤の改変では,20倍以上の高速化が可能であった.また，クラウド連携ではパブリッククラウドである Amazon Web Service との間での仮想マシンイメージ相互変換を実施することで示した．
本システムを使ってさらにPBLを数多く実施して，本システムの評価を継続的に行なうことが必要であると考える．

第4章で述べた設計方針に対応して本プロトタイプ構築を通して検証できたものは単一データセンタにおける以下の設計方針3，4，5，6についてである．

クラウド間の分離については，各クラウド基盤のハードウェア構成を固定にすることでネットワーク設定により実現した．クラウド構築の簡単化は各クラウド基盤の初期化ツールとして構成管理ソフトウェア puppet を活用し開発し，実運用した．LDAPサーバと各クラウド基盤の連携についてはクラウド基盤側の認証部分を改造し認証の共有化を実施した．クラウド間連携については大規模データの保存をdrupalベースで開発したCMSにより実現し,それとクラウド側のObject Storageの間でSave/Resotore機能を開発した．

本節で説明したプロトタイプの位置づけは以下のようになる．また，検証対象は図6-7の通りである．

(1) 検証評価したい項目

OSSによる実用的マルチクラウド基盤の実現

- 品質確保
- クラウド共通認証基盤
- クラウド共有オブジェクトストレージ
- クラウド間ネットワーク分離
- クラウド基盤自動デプロイ

(2)検証評価方法

OSSによるマルチクラウド基盤を使ったサービス運用

- OSSによる教育クラウド　edubase Coudの構築/運用
- LDAPによるユーザ管理
- Drupalベースのアーカイブサービス
- ネットワーク手動設定
- Puppetとshellスクリプトによるデプロイツール開発

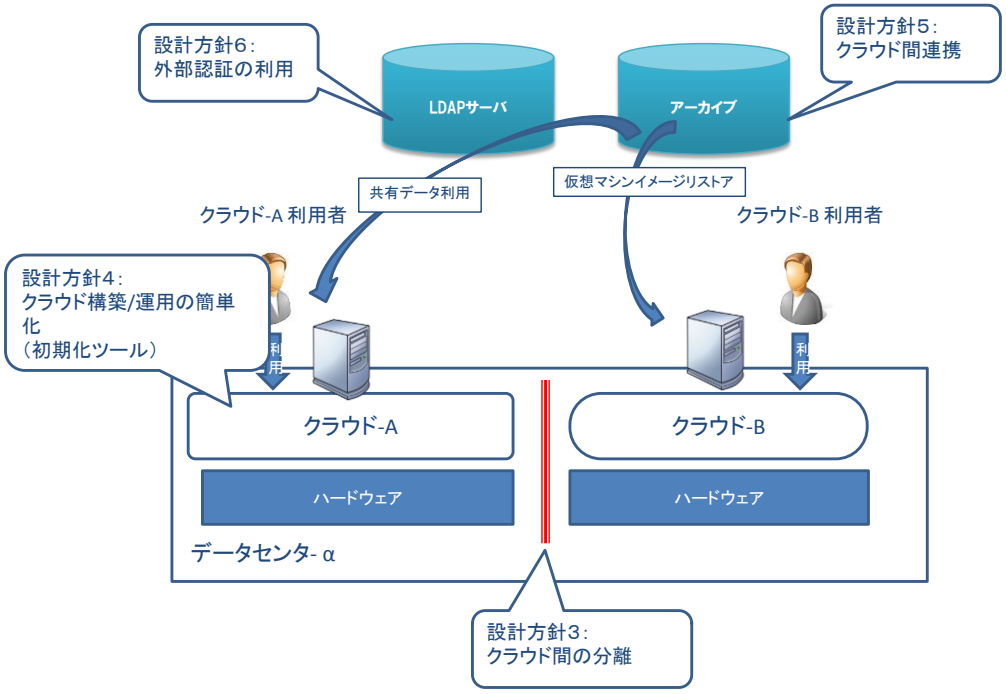


図 6-7　検証 1 の対象項目

6.3　研究クラウド gunnii （検証 2）

第 4 章のアカデミッククラウド設計方針に従って，その 1 つの実装例である NII の研究クラウドプロトタイプシステムを構築し，NII 研究者向けにクラウドサービスを提供している．

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設計方針としてまとめた．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，研究クラウド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．

(6.3-1) 高速分散処理可能な大容量クラスタの動的構築

常設でなくても良いけれど大規模実験を実施してデータ収集を行うためのある一定期間，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．

(6.3-2) 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，その都度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．

(6.3-3) 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．

(6.3-4) 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．

(6.3-5) 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．

以下にその開発の背景，特徴と実装について述べる．

6.3.1 研究クラウド gunnii の特徴

6.3.1.1 開発の背景

情報システムの設備使用率向上と利用コスト低減をもたらすクラウド技術は，「情報システムの所有から使用へ」のパラダイムシフトを牽引し，適用範囲のさらなる拡大が期待されている．従来の IaaS では，サーバ仮想化技術を利用して複数の利用者を一台の物理マシンに収容する方法が採られてきた．このため，

- 仮想化技術のオーバヘッドにより，マシン性能が低下する．
- 他利用者のマシン使用状況により，マシン性能が変動する．

といった課題があり，高性能・安定性能が要求される科学技術研究分野での利用には，限界があった．前節までで説明したクラウド基盤ソフトウェア dodai は，仮想化技術を活用した IaaS と同様にセルフサービス・オンデマンドで，物理マシンを貸し出すことを可能にしている．このため，

- マシン性能の高速性が求められる大規模計算
- マシン性能の安定性が求められるソフトウェア性能の評価・ベンチマーク

などの科学技術研究分野に対し，クラウド技術の適用範囲を広げることができる．

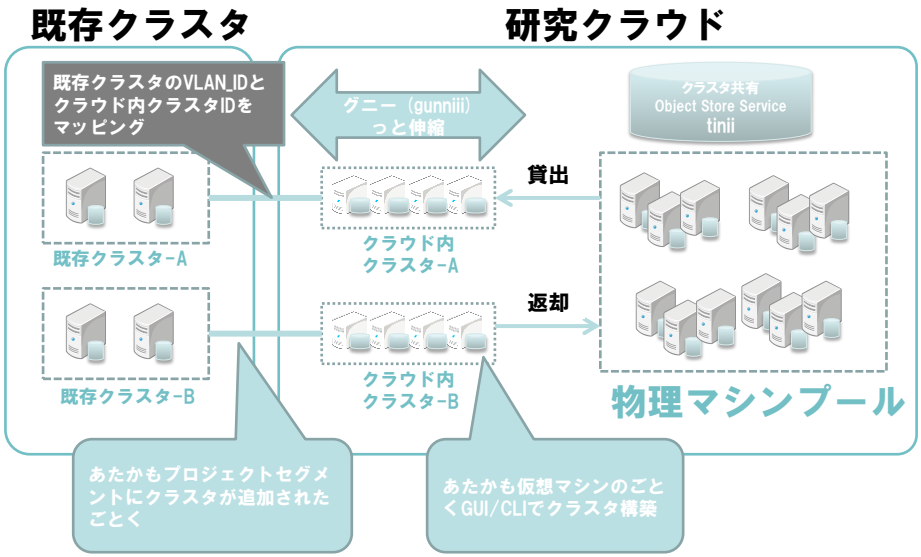


図 6-8 研究クラウド概要

6.3.1.2 特徴

① セルフサービス・オンデマンド

セルフサービス・オンデマンドで，物理マシンをウェブブラウザーからマシン数や OS などの情報を投入するだけで，物理マシンの設定（OS インストール，ネットワーク設定など）が自動実行され，短時間で物理マシンの貸し出しが受ける．マシン返却時には，物理マシンの初期化（ハードディスク消去など）が自動実行され，再利用によるセキュリティ問題を回避する．物理マシン貸し出しのアプリケーションインターフェースは，オープンソースのクラウド基盤として定評のある AWS EC2 のインタフェースと共通化している．このため，計算システムの開発時には柔軟性の高い仮想マシンクラウドを使い，実利用時には高性能の物理マシンクラウドを使う，といった使い分けが簡便に可能となる．

② 研究室ネットワーク内の既存計算機リソースとの統合

利用者は，貸し出された複数の物理マシンをクラスタと呼ぶ単位で分類し，クラスタごとに異なるネットワークに所属させることができる．所属させるネットワークは，既存の研究室ネットワークとするもできるため，貸し出しされた物理マシンを，あたかも手元にあるかの様にシームレスに利用することが可能である．

クラウド基盤ソフトウェア dodai では，柔軟なネットワーク構成を，物理的な配線変更を行うことなく，オンデマンドで提供するために，OpenFlow 技術を活用している．

③ ミドルウェアの自動インストールにより直に使える計算機環境を提供

貸し出された物理マシンには，クラスタ単位に一括して，大規模計算に使われる Hadoop や Sun Grid Engine などの研究用ミドルウェアなどを自動インストールすることができる．このため，短時間で計算機環境を増強して，研究効率の向上がはかれる．

④ クラスタ共通のストレージサービス

オブジェクトストレージサービスや NFS を用いてファイルを共有できる．

⑤ 学認で認証

認証に学認を使用する．学認にログインしていれば追加の認証なしに研究クラウドを利用できる．

設計方針 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．

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．このことにより設計方針 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．

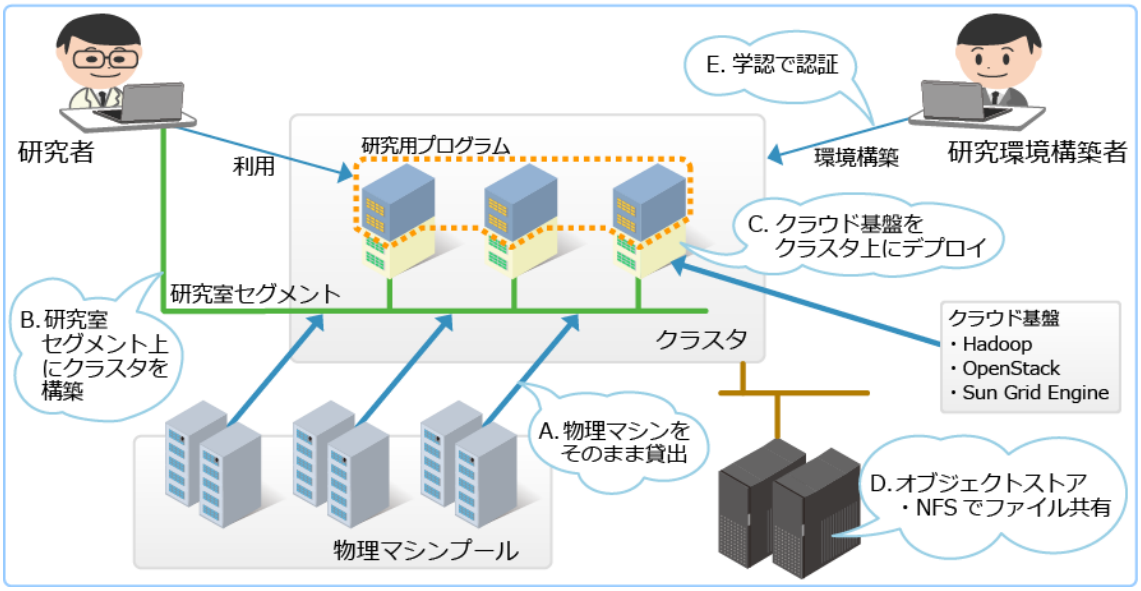


図 6-9 研究クラウドの特徴

6.3.2 研究クラウドシステム

研究クラウドシステムの構成を，サーバ，ネットワーク，ストレージに分けて説明する．

サーバ

(1) コンピュートサーバー

研究者の利用用途によって，最適な計算機を提供するために，複数の種類の計算機をコンピュートサーバーとして利用できる．具体的には，以下の 2 種類のコンピュートサーバーを用意した．

- コンピュートノード　計算能力が高く内蔵ディスクの容量が少ない．
- ストレージノード　内蔵ディスクの容量が多い．

(2) ユーティリティサーバ

ユーティリティサーバは，以下の理由から，用途ごとに独立したサーバを用意した．

- ユーティリティサーバがシステムの性能上のボトルネックとならないようにする．
- ユーティリティサーバの運用管理を容易にする．
- コンピュートサーバーとは接続するセグメントが違うことから，必要なセグメントにだけ接続する構成とする．

ネットワーク

本システムでは，NII に既設の L3 スイッチに接続する新規システムとして設計することで，既存のネットワークの運用への影響を最小とした．

本システムでは以下の理由から，ネットワークセグメントをその用途により分割した．

- セキュリティの観点から，流れるデータの種類ごとに別々のセグメントを用意する．
- セキュリティの観点から，管理用のセグメントとサービス提供用のセグメントを分ける．
- サービスレベルの異なるデータは別々のセグメントに分ける．
- ネットワーク機器の停止が直接サービスの停止につながるセグメントに関しては二重化構成とし，ネットワークが停止してもサービスは停止しないセグメントに関してはシングル構成と

する.

- 必要とされるネットワーク帯域が異なるトラフィックは,適切なネットワーク速度のセグメントに収容する

具体的には,以下の 8 つのセグメントから構成される.各セグメントの特徴を以下に記述する.

(1) サービスセグメント

サービスセグメントは，研究者が実行するアプリケーションに関する通信を行うためのセグメントである．OpenFlow スイッチおよび OpenFlow コントローラによってクラスタ用の閉域ネットワークを自由に切り出し，研究グループのネットワークと接続する機能を提供します．閉域ネットワークによりクラスタ間のネットワークの独立性を確保する．また，クラスタに構築するクラウド基盤では，その閉域ネットワーク内で自由に VLAN を利用できる.

ネットワーク機器の障害が直接サービスの停止につながるセグメントであるため，サービスを停止させないように二重化構成とする.

(2) ストレージセグメント

コンピュートサーバーとストレージの通信を行うためのセグメントである．通信量が多いため,サービスセグメントとは分離して別セグメントとする．ストレージの通信は通信量が非常に多く高速・高帯域である必要があるため，10Gbps のネットワーク装置で構成した.

(3) クラスタ管理セグメント

物理マシンのデプロイに関する通信，サーバの監視および設定ファイルのバックアップのために利用するセグメントである．デプロイとバックアップは通信量が多く，高速・高帯域である必要があるため，10Gbps で構成する．なお，設計の前提として，1 台の物理マシンのデプロイに関する通信のデータ量 5Gbytes を想定した.

(4) DMZ セグメント

研究クラウド運用者外からの運用システムへの不正アクセスを防ぐために，ファイヤウォールと DMZ セグメントを設置して，システムに対するアクセスを制限した.

(5) 運用システムセグメント

運用管理セグメントは，以下のユーティリティサーバを収納するための運用管理向けセグメントである.

- ユーティリティサーバ（クラスタインストーラ）
- ユーティリティサーバ（バックアップ）
- ユーティリティサーバ(運用)

(6) OpenFlow スイッチ制御セグメント

OpenFlow コントローラから OpenFlow スイッチを制御するための通信を行う専用のセグメントである．ネットワーク機器の障害が直接サービスの停止につながるセグメントであるため，サービスを停止させないために二重化構成とした.

(7) SAN セグメント

ストレージに関連する以下の機器を接続し，ストレージ関連の iSCSI プロトコル通信だけに利用するセグメントである.

- ストレージサーバ
- ストレージサーバ用ストレージ装置
- ユーティリティサーバ（バックアップ）
- バックアップ用ストレージ装置
- ユーティリティサーバ（運用）

(8) IPMI ネットワークセグメント

管理コンソール端末から電源操作を含む操作・運用をするための IPMI の通信を行うセグメントである．全てのサーバ機器の IPMI ポートを接続することで，IPMI によるサーバの一元的な管理を実現した．

ストレージ

以下の理由から，ストレージサーバとストレージ装置は別々の機器で構成し，ストレージサーバとストレージ装置は iSCSI プロトコルで接続する構成とした．

- 拡張性の観点から，ストレージサーバおよびストレージ装置を増設した場合でも，各ストレージサーバから各ストレージ装置を増設前と同様にマウントできる必要がある．
- コンピュートサーバーなどからストレージサーバへのアクセスプロトコルは NFS とする．

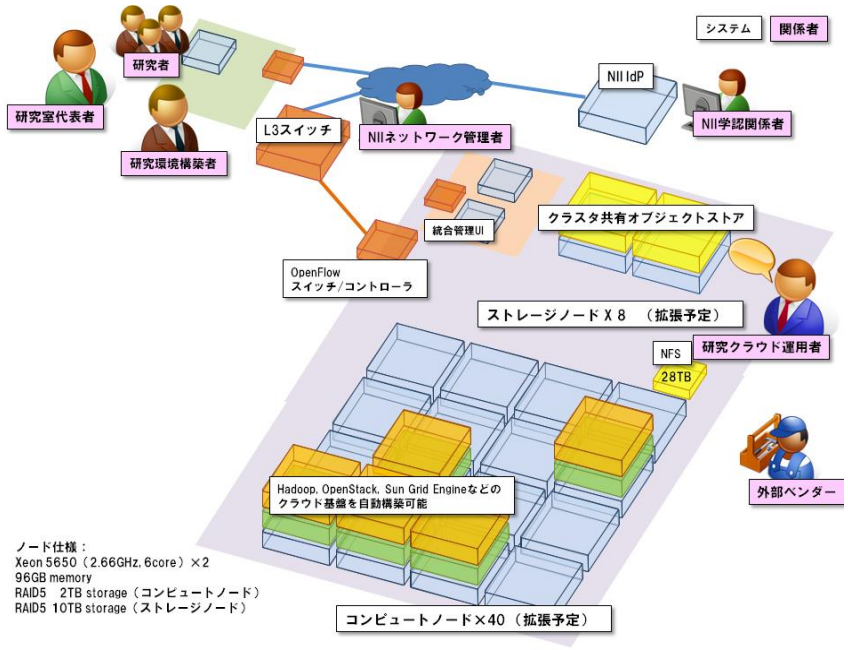


図 6-10 研究クラウドシステム構成

6.3.3 研究クラウドサービス

上記，研究クラウドシステム上の構築した基本的なサービスについて以下に説明する．

物理マシン貸し出しサービス

貸し出し方法

- セルフサービス：研究環境構築者がポータルサイトで必要な台数を投入
- アカウントごとに設定された制約の範囲内で借用

事前準備

- 研究クラウドポータルサイトへのアカウント登録（アカウント＝研究室）
- 貸し出しマシンは，利用者の所属する研究室の LAN に接続
- 貸し出しマシンに割り当てる固定 IP アドレスを用意
- 研究室 LAN のファイヤウォールを適切に設定

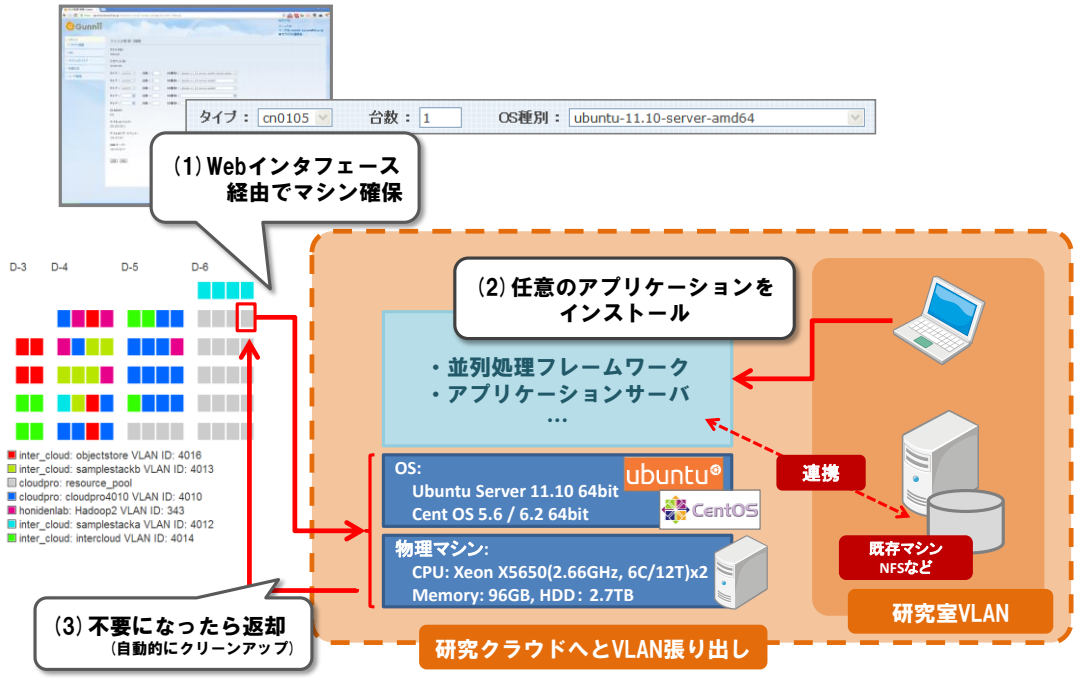


図 6-11 物理マシン貸し出し　利用イメージ

クラウド基盤（ミドルウェアなど）一括導入サービス

サービス概要

- 貸し出しされた複数の物理マシンに，クラウド基盤（ミドルウェアなど）を一括してインストールするサービス
- 提供する基盤，ミドルウェアは 2012 年 7 月現在，以下のとおり．

  クラウド基盤　OpenStack

  分散処理ミドルウェア　Hadoop

  分散処理ミドルウェア　Sun Grid Engine

事前準備

- 研究クラウドポータルサイトへのアカウント登録（アカウント＝研究室）
- 研究環境構築者が，研究クラウドポータルより，物理マシンの貸し出し

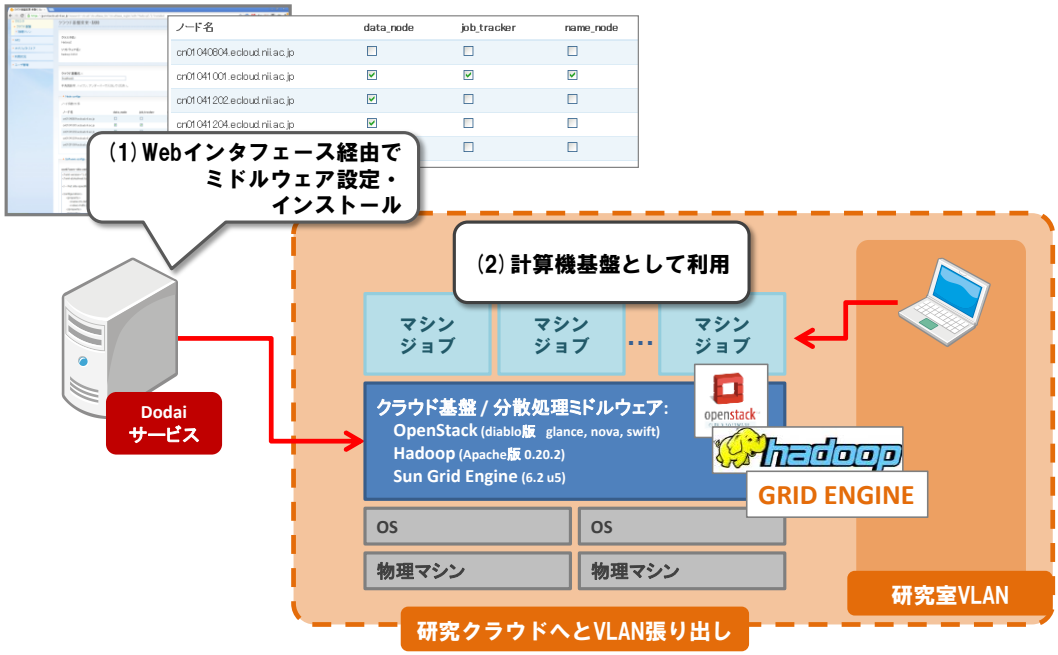


図 6-12 クラウド基盤一括導入　利用イメージ

オブジェクトストレージサービス

サービス概要

- ウェブ （HTTP）経由で利用できる汎用データ（オブジェクト）保管庫
- Amazon S3 と同様のサービス
- HDD のバックアップなどを一時的に保管する場として提供
- 研究クラウドはもとより，NII ネットワークのどこからでも利用可能

事前準備

- 研究クラウドポータルサイトへのアカウント登録（アカウント＝研究室）
- 研究環境構築者が，研究クラウドポータルサイトからオブジェクトストレージ利用者登録

6.3.4 Dodai-compute の評価

リソースの割り付けの柔軟性を実現するためには，割り付け，開放がソフトウェアから簡単に行なえ，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，その割り付け，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，予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“設計方針 1：クラウドへのリソース割り付けの柔軟性”，“設計方針 2：ベアメタルな性能”および“設計方針 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．

但し，デ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いけ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けれど 4TB 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．

6.3.5 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 ハンズオンセミナーを実施し用いて 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ー内での利用には適用可能な時間であった．[32]　さらに，図 6-13 に示すように The University of Alabama では OpenStack の構築に dodai-depploy が活用されていて利用者向けマニュアルが作成されている．

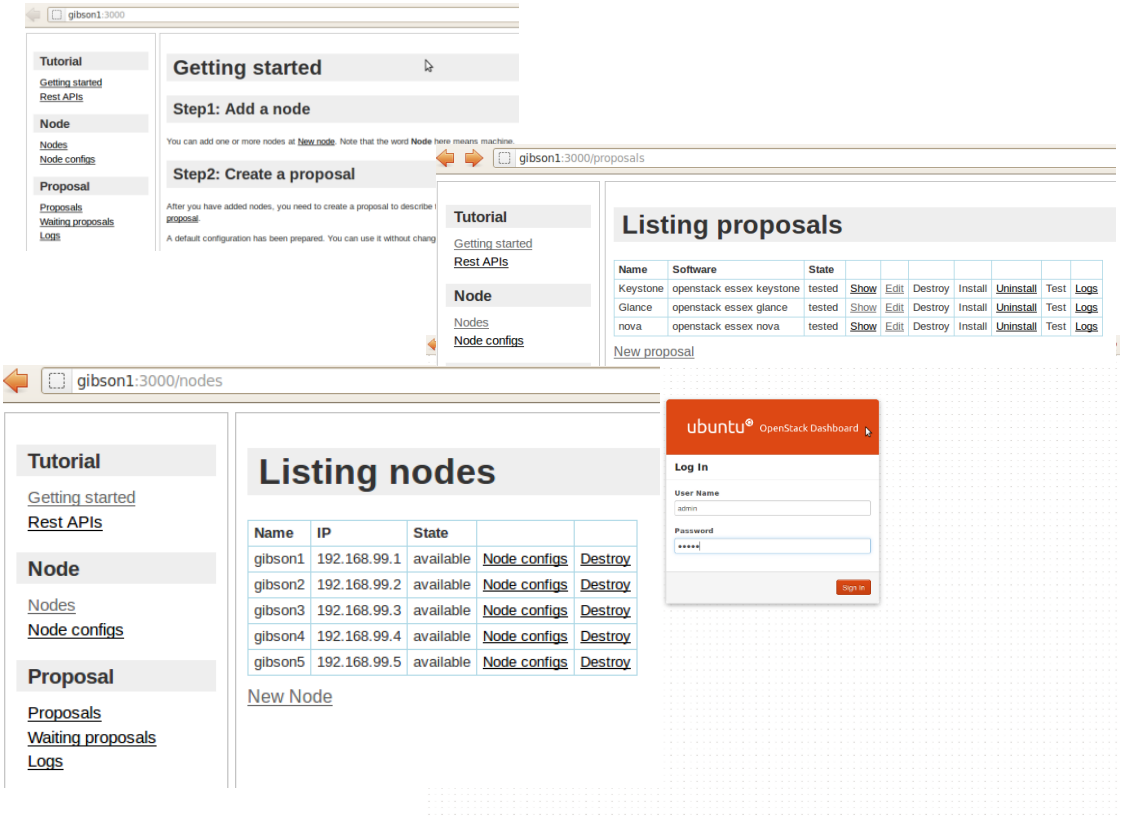


図 6-13　The University of Alabama での利用例（GUI）
(http://cloud.cs.ua.edu/wiki/index.php/Openstack_Essex)

なお，現在 dodai-deploy が deploy ソフトウェアとしてサポートしているものを図 6-14 にまとめる．

	ubuntu 10.10	ubuntu 11.04	ubuntu 11.10	ubuntu12.04/ CentOS6.2
OpenStack Folsom(Compute, Glance, Keystone, Swift) Compute includes Nova, Horizon, Quantum, Cinder				:)
OpenStack Essex(Nova, Glance, Swift, Keystone)				:)
OpenStack Diablo(Nova, Glance, Swift)	:)	:)	:)	
hadoop 0.22.0	:)	:)	:)	:)
sun grid engine 6.2u5	:)	:)	:)	

図 6-14　dodai-deploy がサポートしているソフトウェア

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 が持つ予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か限り並列 deploy 実行を指示する機構の効果で，図 6-15 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．但し，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 を使い切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“設計方針 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．

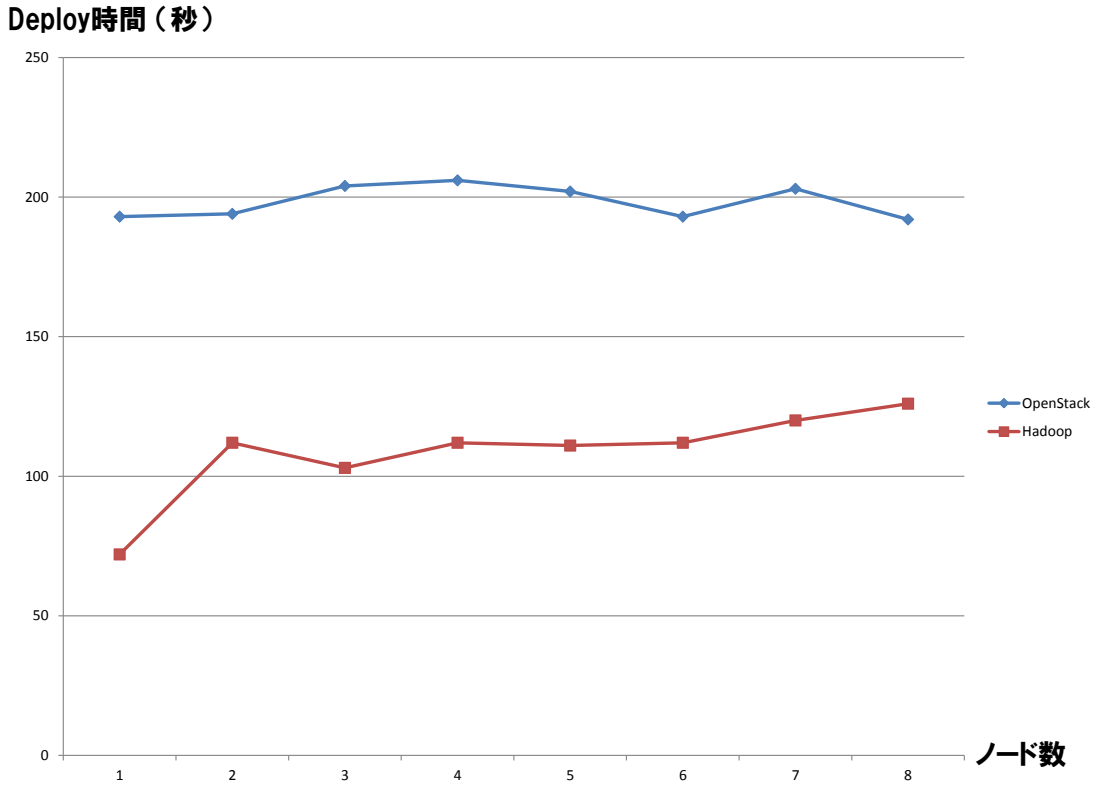


図 6-15　Dodai-deploy の性能評価結果

6.3.6　検証 2 に関するまとめ

第 4 章で述べた設計方針に対応して本プロトタイプ構築を通して検証できたものは単一データセンタにおける以下の設計方針 1，2，3，4，6 についてである．ベアメタルクラスタの構築については PXEboot および IMPI というコモディティハードウェアが持つ標準的な機構を組み合わせることで制御可能な仕組みを dodai-compute として OpenStack Compute の拡張として実現した．この際，OpenStack Compute の持つ API をベアメタル対象でも維持した．クラウド間の分離につ

いては OpenFlow の閉域網で実現することで，各クラウド基盤内で自由に VLAN-ID や MAC アドレスを付与できる仕組みを作った．既存クラスタと研究クラウド内のクラスタを L2 接続するために OpenFlow Controller 側に既存クラスタに対応する VLAN-ID と該当クラスタの閉域網に対応するリージョン ID を紐付けた．

クラウド基盤構築時の性能をノード数増加に対して押さえるために dodai-deploy の各ノードでの並行実行を支えるため deployment の全体制御を行う MCollective を導入した．この MCollective に deploy するソフトウェアコンポーネントの依存関係を入力することで実行可能で最大の並列度を達成した．

本節で説明したプロトタイプの位置づけは以下のようになる．また，検証対象は図 6-16 の通りである．

(1) 検証評価したい項目

OSS による実用的ベアメタルクラウドとその上へのクラウド基盤動的構築

- セルフサービスでのベアメタルクラスタ貸し出しとその上でのクラウド基盤構築
- 既存クラスタとベアメタルクラウド内クラスタの L2 接続
- 認証基盤との連携

(2) 検証評価方法

dodai によるベアメタルクラウドとクラウド基盤動的構築サービスの構築/運用

- dodai と swift による研究クラウド　gunnii の構築/運用
- 学認連携による認証

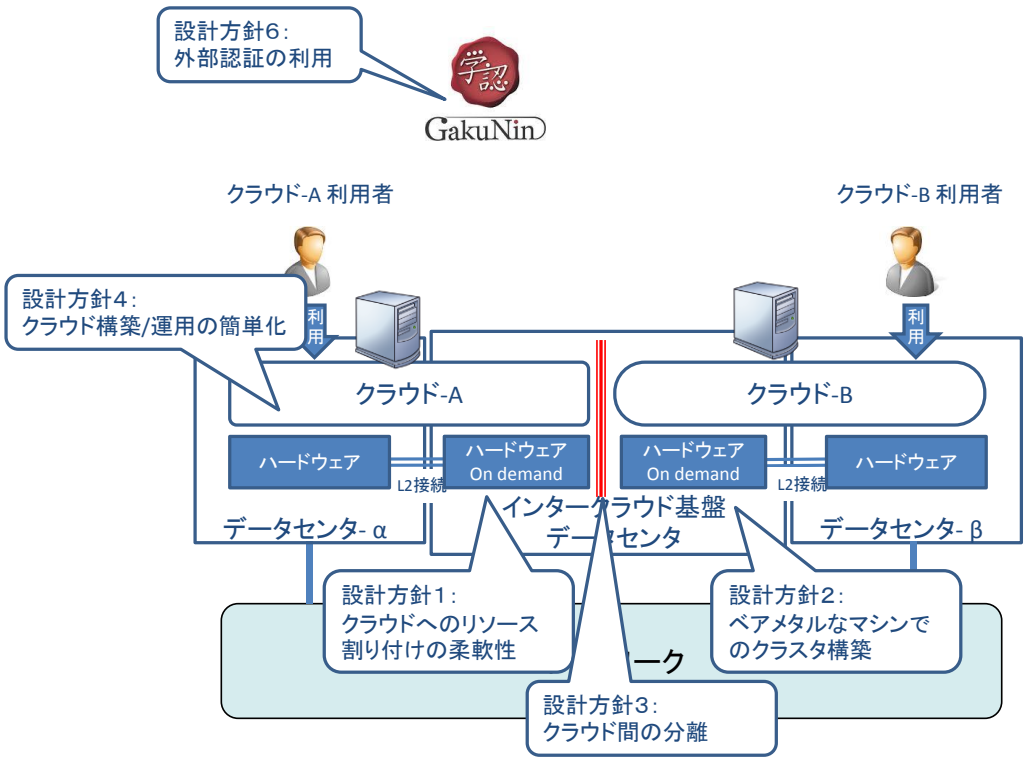


図 6-16 検証 2 の対象項目

6.4　地域分散オブジェクトストレージ　colony（検証 3）

Colony プロジェクト[33] では SINET をターゲットとしたオブジェクトストレージシステムの展開を検討する中で，データセンタ内などの LAN 環境での利用を前提として設計されている swift[28]を，SINET のような広域網（WAN 環境）で相互接続された複数サイトにより構成されるシステム上で実利用可能とすることを目指してきた.

その目的のために， swift に追加した機能の概略は以下の通りである.

- ring の zone 情報から，サイトの配置を判断可能とする.
- proxy-server に WAN 経由での PUT をさせず，自サイトのみにレプリカを作成させ，非同期で本来のレプリケーションサイトへコピーする.
- GET 時も proxy-server は自サイトへのアクセスを優先させる.
- GET 時にオブジェクトキャッシュサーバを経由させ，WAN 経由の通信を減らす.

前提として proxy-server がストレージサーバに対するオブジェクトの書き込み（PUT）や読み込み（GET）を行う際に，WAN 回線を経由することで処理速度は低下する．これは一般に WAN の回線速度は LAN と比較して低速であるか，あるいは帯域が十分であっても，遅延は減らないためである.

6.4.1　基本アイデア

swift がオブジェクトのレプリカ配置を決定する際には，ring への問い合わせを行い，結果として分散された複数の zone が得られる．この複数の zone へレプリカがそれぞれ配置される.

最初に得られた zone が機能していない（マシンダウン，ネットワーク障害など）場合には，代替として handoff という代替の zone が採用される.

仮に「自サイト」以外のすべての zone が機能不全であれば，swift は自サイト内の zone のみをレプリカ配置先として選択することになる.

Colony では，これを意図的に行わせることによって proxy-server のアクセス先は「自サイト」内に限定され，LAN スピードでの高速アクセスが可能となる.

PUT 時に自サイトに一時的に集中して配置されたレプリカは，object-replicator の機能により，本来の配置先に非同期に移動される.

本来の配置へのレプリケーションが完了するまでの間，サイト A とサイト B 上のオブジェクト A の内容は同一でない．しかし，これは通常の PUT 実行中，複数の zone へのレプリカ配置処理中において，処理が完了している node と完了していない node とが混在している状態と同一であるため，本機能追加による固有の問題とはならない．また，レプリカ配置の変更以外は，通常の swift の機能を変更していないため，差異や固有の問題点は最小限度である.

6.4.2　zone へのサイト情報の付与

Swift の ring 情報は，どのストレージサーバも距離的な観点からは区別がない．広域化対応のため，swift に「自サイト」「他サイト」の概念を加えるために，proxy-server から見て近距離(LAN 内)にある zone と，遠距離(WAN 経由)となる zone を区別できるようにする必要がある.

それには，ring の中に含まれている zone 情報を利用する．zone 情報は ring の中でユニークな十進数であるが，これをサイト情報を示すものとして扱う.

具体例を挙げると，zone 名を 101，102，201，202，203，301，302 と付与されたノード群がある．ある proxy-server は自己を 100 であると認識する．これは設定ファイルで設定する．100 を始点としそこから 199 までの範囲にある zone 名を，proxy-server は自サイトに含まれるノードであると判断する．この場合 101，102 が自サイトとなる．

また，別のサイトにある proxy-server は自己を 200 とし，始点から 399 までが自サイトであると判断する．この proxy-server からは 201，202，203，301，303 が自サイトである．

この方式であれば ring の構造に手を加えることなく，zone 名の命名ルールのみでサイト情報を持つことができる．

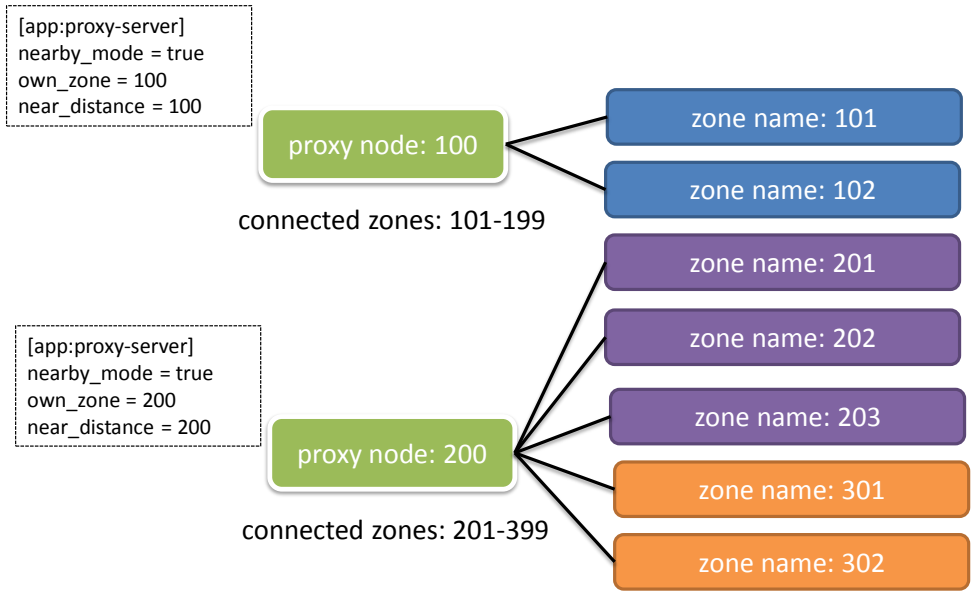


図 6-17 zone へのサイト情報付与

6.4.3 PUT 処理

Swift はすべてのレプリカ配置先への処理が完了するまで，クライアントに対し PUT の処理完了を伝えない．レプリカ配置先に WAN 経由の zone が選ばれた場合，長時間かかる処理の完了を待ち続けることになる．

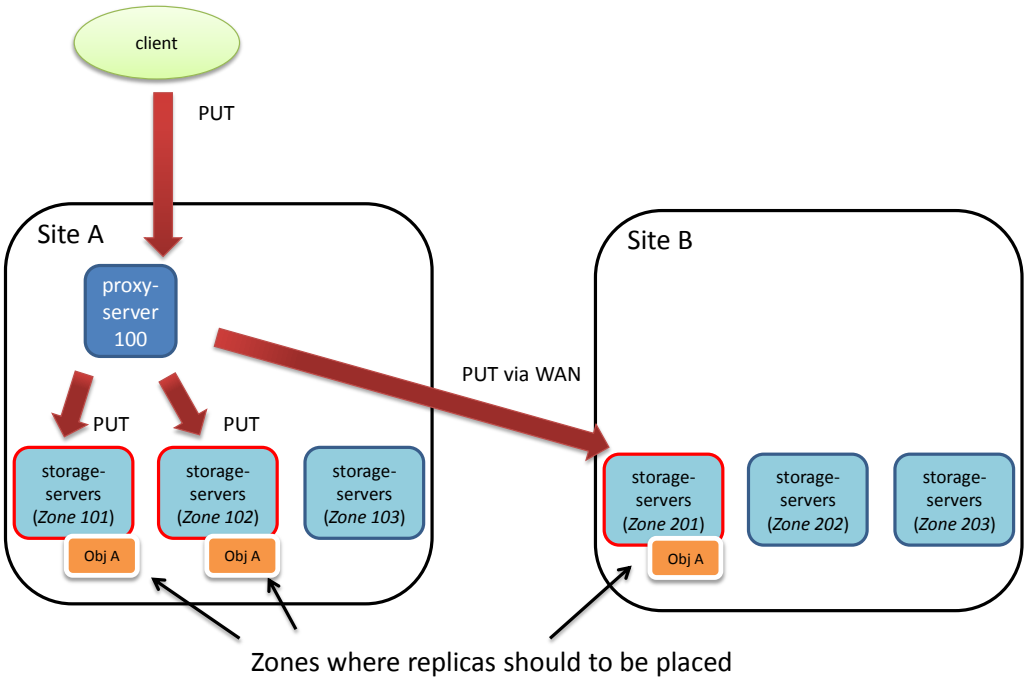


図 6-18 通常 swift での PUT 処理

Colony では，PUT 処理を自サイト内のストレージサーバのみに対して行う．ネットワーク的に近距離のノードに対しての通信となるため処理は高速になる．（図 6-19）

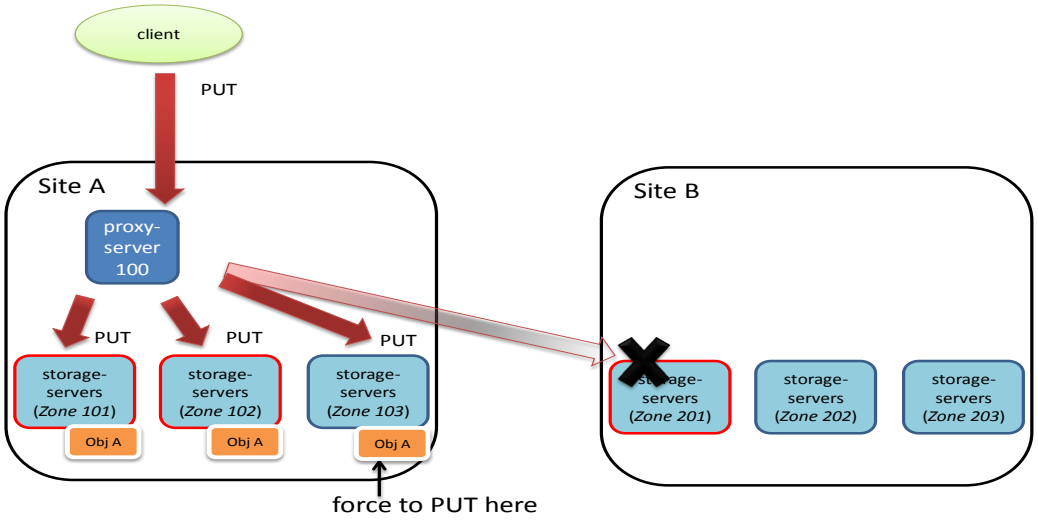


図 6-19 広域化対応された PUT 処理

しかし，PUT 処理直後はオブジェクトが本来のレプリカ配置先とは異なり，自サイトに偏った配置となっている．これは ring の指示する本来の配置とは異なった状態である．

この偏りは object-replicator によって是正される．object-replicator はレプリカを本来の配置先にコピーし，そうでないレプリカを削除することで，最終的には本来のレプリカ配置に直される．これは object-replicator 本来の処理である．object-replicator の処理が完了すると，本来の位置へのレプリケーション配置が完了する．（図　6-20）

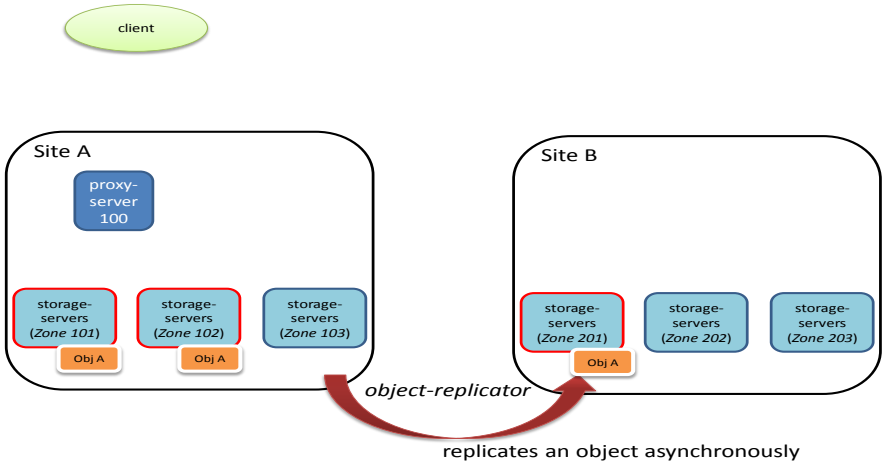


図 6-20　Object-replicator による非同期でのレプリカ配置

6.4.4 GET 処理

Colony は，自サイト内のストレージサーバを優先してオブジェクト取得を試みる．そしてそこになければ本来のレプリカ配置先を探すという挙動を取る．これは，GET 対象のオブジェクトが以下のいずれかの状態にあることを想定し，探索をおこなうものである．

1. 本来のレプリカ配置先にまだ展開されていない
2. 本来のレプリカ配置先に展開済み

対象のオブジェクトが自サイトでの PUT 処理直後（まだ object-replicator に展開されていない）であれば，オブジェクトは自サイトのストレージサーバにある確率は高い．object-replicator がレプリカの展開を完了していても，自サイトに配置されているレプリカ先を優先するので（自サイトが本来のレプリカ配置先に含まれていれば），WAN を経由することは避けられる．そのため，まず自サイトを探索する．（1 のケース）

自サイトになければ，本来のレプリカ配置先を探索する．（2 のケース）

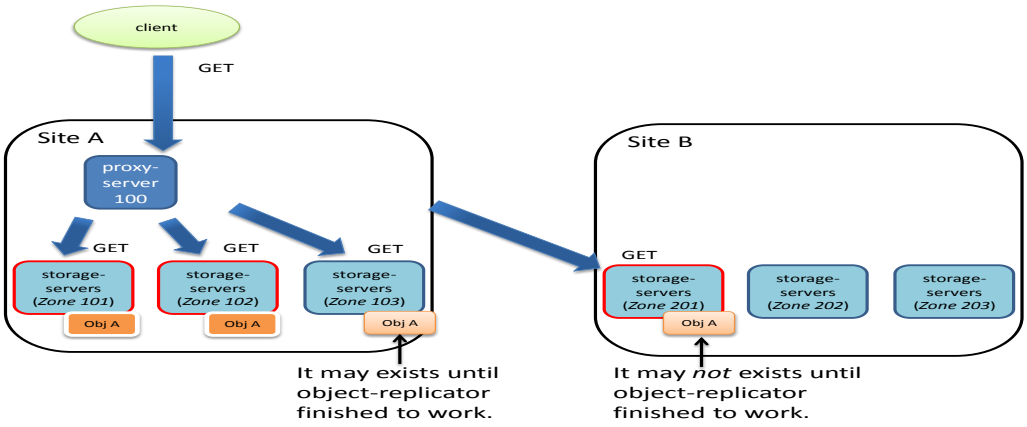


図 6-21 広域化対応された GET 処理

GET は自サイトを優先するようにしているが，オブジェクトのレプリカ配置先によっては自サイトにレプリカがない場合もあるだろう．その場合にはサイトをまたいだ（WAN を経由した）アクセスが発生する．これを緩和するために，キャッシュ機能を持ったリバースプロキシーサーバとの連携機能を追加することで，性能を向上させる．

6.4.5 実験と計測結果

本章では，考察・検討した機能を実装し，性能計測した結果を説明する．

6.4.5.1 実装

Swift-1.4.8 の以下のコードに対し機能の変更と追加を行った．

- swift/proxy/server.py：proxy-server 本体
- swift/common/ring/ring.py：ring への問い合わせ機能

6.4.5.2 実験環境

下図が実装の結果を確認するための実験環境の概要である．一ツ橋と北大はそれぞれで「サイ

ト」として扱われる．サイト内はLANによってサーバ間が接続されているが，サイト間はWAN回線(SINET)によって接続されている．両サイトとも proxy-server と，ストレージサーバ（account-server，container-server，object-server，およびその他のサーバ）で別のマシンに分けられており，1つのストレージサーバには3つの zone が設置されている．

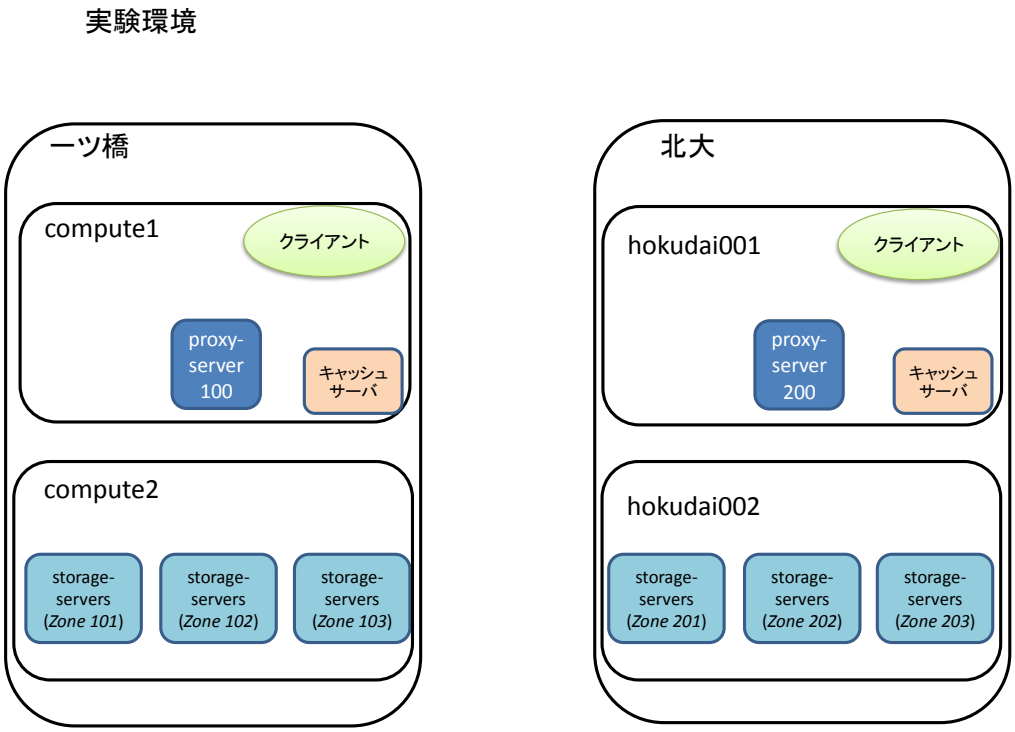


図 6-23 実験環境

6.4.5.3 計測結果

計測は proxy-server と同じマシン上のクライアントから，テスト用のオブジェクトの PUT もしくはGETを5回ずつ行って，秒あたりのアップロードバイト数もしくは秒あたりのダウンロードバイト数を取得している．クライアントは”curl”を用いたスクリプトである．

PUTについてはパッチ適応前と適応後の比較を行った．

GET についてはパッチ適応前と適応後，適応後にはキャッシュサーバ連携機能（proxy-serverの middleware 実装と，dispatcher によるものの2種）との比較も行った．

テスト用に用いたオブジェクトの名称は以下の表の通りである．各オブジェクトのファイルサイズは名前が示している．また“swift-get-nodes”コマンドを用いて，オブジェクトのレプリカがどのサイト上のノードに配置されるかを，事前に確認してある．表の中でサイト名の列の数字は，3つあるレプリカが各サイトにどのような配分で配置されるかと示している．

表中の”1.txt”,”5.txt”は，特定のサイトにレプリカが偏って配置されるオブジェクトである（サイズ1GB）．他の条件との比較のためにGETでの計測内容に含まれている．

表 6-3 評価用オブジェクト

	一ツ橋	北大
	1	2
1M.dat	2	1
10M.dat	1	2
100M.dat	2	1
1G.dat	2	1
1.txt (1G)	3	0
5.txt (1G)	0	3

そのほかに，テストの実施に当たっては，以下の条件を整えている．

PUT

- ストレージサーバ上ではメインのサーバプロセス以外は起動しない
- リクエストの実行は 10 秒間隔
- テスト開始前にデータはすべて削除

GET

- リクエストの実行は 10 秒間隔
- テスト開始前に squid のキャッシュデータはすべて削除

6.4.9.4 PUT 処理

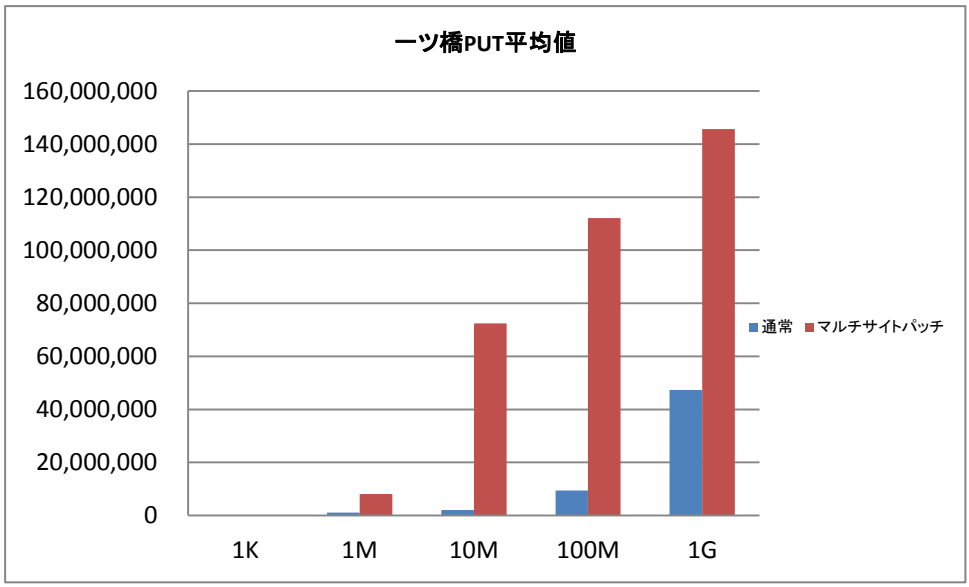


図 6-24-1 PUT 処理性能

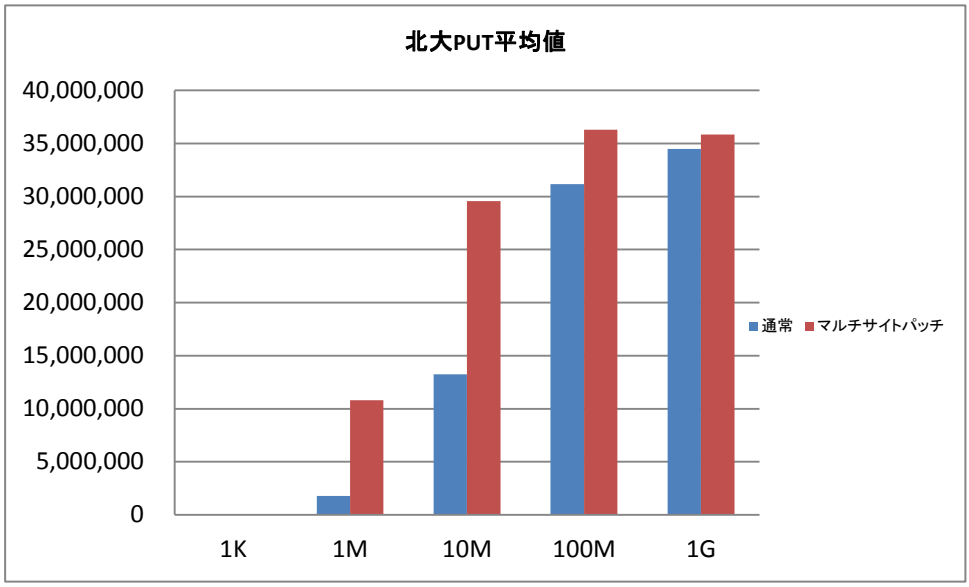


図 6-24-2 PUT 処理性能

マルチサイト対応により，PUT 先は自サイト内に限定されるため，サイト内の LAN 速度に応じたスピードとなる．一ツ橋での効果が大きいのに対して，北大での効果はわずかである．これは，北大ではサイト内とサイト間との速度差があまりないためである．

6.4.9.5 GET 処理

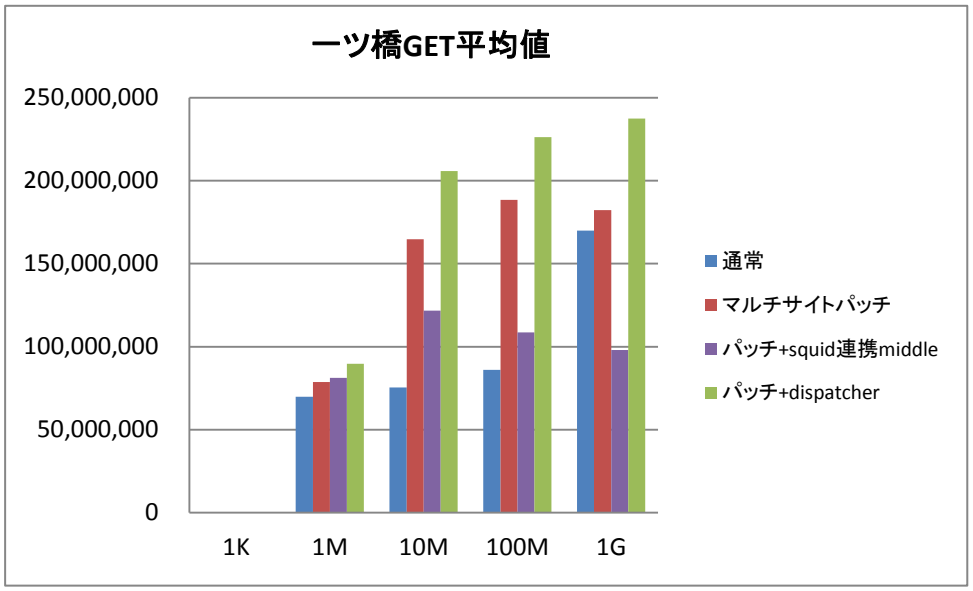


図 6-25-1 GET 処理性能

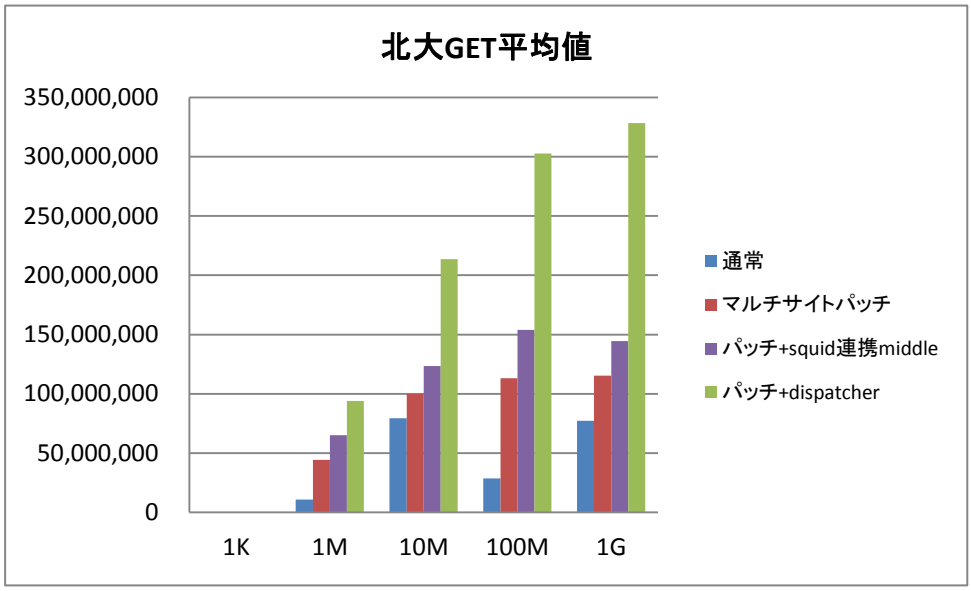


図 6-25-2 GET 処理性能

マルチサイト対応では，まず自サイト内のストレージサーバから優先してオブジェクトを得ようとする．そのためオブジェクトがサイト間で分散されている場合では，必ず自サイト内サーバへのアクセスとなり，通常の proxy-server と比べ結果は変動しなくなる(標準偏差の値の低下)．

しかし，オブジェクトが一方のサイトに偏って配置されている場合では，マルチサイト対応に特別の効果はなく，常にサイト外へのアクセスを強いられるため低速となる(1.txt,5.txt)．この状況を緩和するために，キャッシュサーバ連携機能を導入する．

キャッシュサーバ連携では，あるオブジェクトに対する最初のアクセスは連携なしの場合よりも遅くなるが，初回以降のアクセスでは proxy-server へのアクセスはなく，クライアントへはキャッシュサーバのキャッシュが渡されるため，高速化が期待できた．

まず，proxy-server の middleware として実装したものの性能を確認する．結果としては，一ツ橋環境では squid によるキャッシュの効果よりも，キャッシュなしの結果のほうが高速であった．北大環境ではこれが逆転しキャッシュは効果を発揮している．

これは一ツ橋環境では LAN の高速性が squid の処理を上回ったのであり，北大環境では LAN が低速なため squid のキャッシュの方が高速に働いたということだと思われる．リバースプロキシーによる高速化は環境により効果が変動するものだと云える．

6.4.6 colony と swift の比較

これまでの広域化対応と通常の実装を比較し，広域化対応のために通常の swift より失われた機能や特長はないかを改めて確認する．

表 6-4 colony と swift の比較

	Colony	Swift
クライアントから見た性能	PUT: 自サイトへのアクセススピード（LAN スピード）となる GET: 自サイトへのア	バックエンドで WAN 経由のストレージサーバへのアクセスが含まれていた

	Colony	Swift
	クセスを優先. 最新オブジェクトが自サイトになければWAN経由のアクセスとなる. オブジェクトキャッシュ機能により，初回以降はLANスピードとなる	場合，アクセススピードは低下する
データの一貫性	クライアントから見てアクセス可能なレプリカが同一内容であることを保証する	すべてのレプリカが同一内容であることを保証する
データの可用性（障害耐性）	定数分のレプリカは常に確保されている.	定数分のレプリカは常に確保されている.
データ容量	オブジェクト容量 × レプリカ数 (通常と変わらず)	オブジェクト容量 × レプリカ数
運用	ring の zone 情報をサイトの情報を反映するように管理する	-

データの一貫性に関しては，自サイトにレプリカを保持している為，システム内の暫定的な状況として，ring の本来の配置と違う node に配置されることになるが，GET 時に最新のオブジェクト更新時間を各 node から取得し，最新オブジェクトを取得する事で一貫性を保つような処理であれば問題はない.

データの可用性に関しては，ゾーン単位では複数のレプリカが存在することで，十分な可用性は保持されているが,サイト単位ではレプリカの配置先が 1 つのサイトに集中する可能性がある. この問題を解決するためには，１サイトに１ゾーン，もしくは 1 サイトにサイト数より少ないゾーン数で ring を構成するような運用対処か，ring の構造自体を変更する必要がある.

今まで検討した自サイトにオブジェクトを一時的に配置する案は swift 高速化のために有用であるが，上記のような運用対処を行った場合などのレプリカがサイト間で分散配置されるケースでも，可用性向上のために，本来の配置に速やかに移す必要がある. よって，object-replicator の高速化を検討する必要がある.

colony の広域化対応では，フロントエンドである proxy-server の機能・挙動は変更されているが，バックエンドであるストレージサーバに対する変更はないためである.

ストレージサーバの挙動は通常の swift とまったく同じであるため,『(proxy-server の行う) レプリカの偏った配置とその収束までの時間経過』による影響を除いて，差異や特有の問題点は存在しない.

6.4.7 検証 3 に関するまとめ

swift のマルチサイト対応についての基本的なアイデアと，それを適用した場合の swift の詳細な挙動について検討を行った．そのアイデアを実際に実装し，実験環境にてオブジェクト操作(GET/PUT)の性能測定を行い，想定環境において通常の swift と比較して十分な性能向上が見込めることが確認できた．

マルチサイト対応の基本アイデアの有用性は確認できたが，これまでの考察の中でも触れているように，このアイデアの中では object-replicator が非常に重要な役割を負っている．恣意的に配置したレプリカを本来のサイトへ配置するのが，object-replicator だからである．object-replicator には可能な限りの高速動作が求められる．object-replicator には追加すべき機能はないが，現状の実装よりも高速に動作するような改良を加えることが，マルチサイト対応のためにも重要である．

第 5 章で述べた設計方針に対応して本プロトタイプ構築を通して検証できたものはデータセンタを跨いで以下の設計方針 5，6 についてである．

設計方針 5 を満たすために，OpenStack Storage swift をネットワーク上の Object Server の位置を意識できるように改造した．重要なのは zone 毎のネットワーク位置を示す指標を作成することであるが，改造を最小にするために zone 名に距離を示す仕組みを導入した．設計方針 6 を達成するために OpenStack の ID 管理モジュール keystone を shibboleth 対応とするため改造した．本節で説明したプロトタイプの位置づけは以下のようになる．また，検証対象は図 6.4-24 の通りである．

(1) 検証評価したい項目

地域分散オブジェクトストレージの実現

- 地域分散オブジェクトストレージの利用性向上
- 地域分散オブジェクトストレージの可用性向上
- レイテンシ―に強い地域分散オブジェクトストレージ
- 遠隔オブジェクトアクセスに関する実用的な性能確保

(2) 検証評価方法

OpenStack Object Storage swift のネットワークアウェア化

- Dispatcher アーキテクチャによる dropbox 的ビューの実現
- ゾーンを意識したレプリケーション実装
- 非同期レプリケーション
- キャッシュ機構の dispatcher への埋め込み

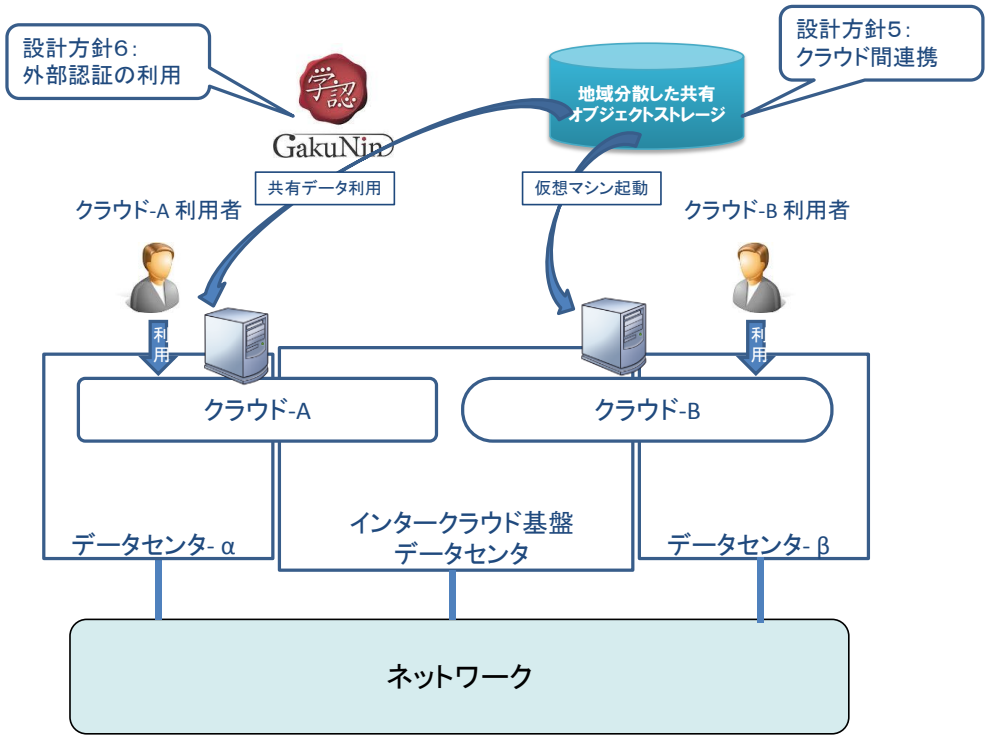


図　6-26 検証 3 の対象項目

6.5　　インタークラウド基盤実験システム（検証 4）

SINET を経由したクラウド基盤の延伸およびその上でのアプリケーション実証検証を行うため，インタークラウド基盤実験システムを構築した．

この上でのアプリケーション実証検証として，災害時や法定停電時に活用できるクラウドマイグレーションを選択し，そのアプリケーション開発と実験を行った．

6.5.1　　インタークラウド基盤実験システム構成

研究クラウドを構築した際に用いた基本アーキテクチャを踏襲したアーキテクチャとする．各拠点には dodai-compute を使ったベアメタルマシンの動的配備サービスを構築する．そのサービスにより動的な各拠点でのクラスタを構築した後，大学側のクラウドの属するネットワークを SINET-4 の VPLS 機能を使って該当拠点のインタークラウド基盤に接続する．この際，そのネットワークに対応する VLAN-ID を VPLS 機能の VLAN-ID マッピング機能を使って，インタークラウド基盤側に出てくる VLAN-ID との動的マッピングが可能である．このインタークラウド基盤側に出てくる VLAN-ID とインタークラウド基盤側で OpenFlow Controller が管理する該当閉域網とを図 6-27 のように紐づける ．

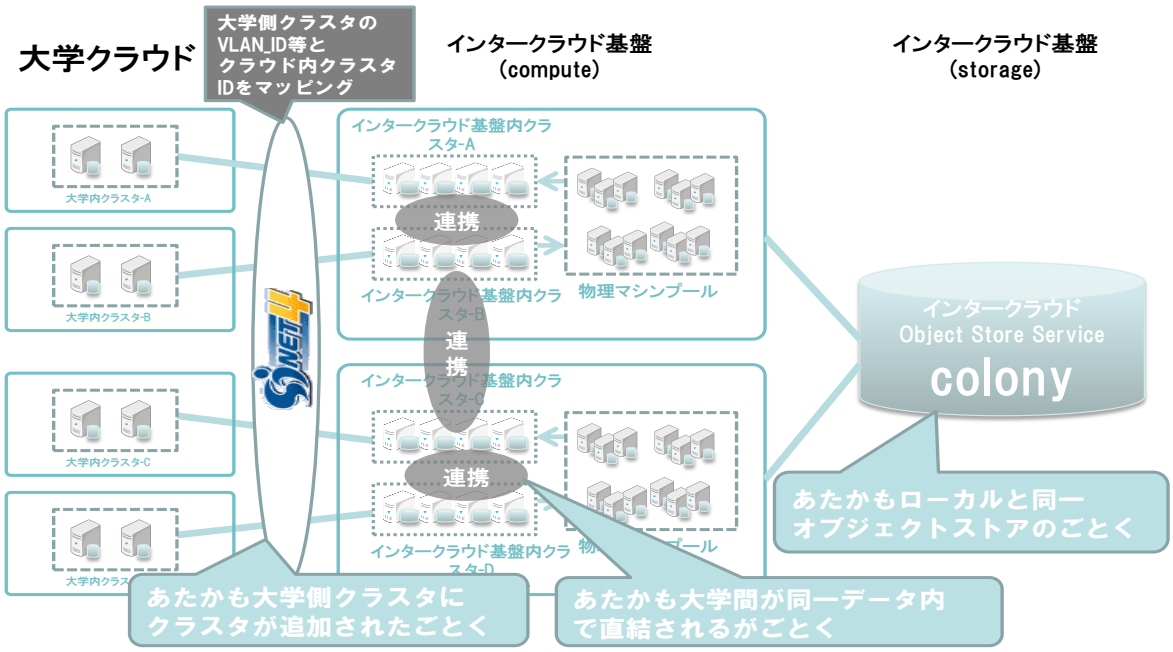
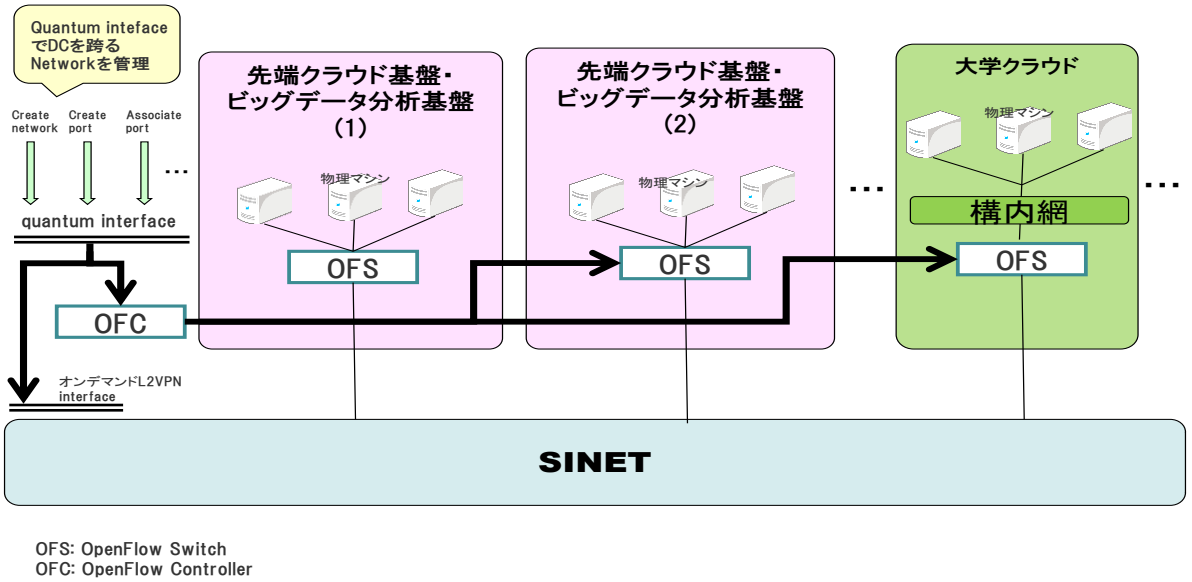


図 6-27　拠点を跨る動的配備アーキテクチャ概要

具体的な実装としては各拠点の存在する OpenFlow Switch 全体を管理する OpenFlow Controller に対して dodai-compute が OpenFlow Network サービスの標準インタフェースである quantum 経由で呼び出す. この際, quantum plug-in の拡張により SINET　L2 オンデマンド機能を使える. (図 6-28)



OFS: OpenFlow Switch
OFC: OpenFlow Controller

図 6-28　拠点を跨る動的配備実現方法（OpenFlow 活用方法）

各大学側の VLAN-ID は大学側の管理下にあるためインタークラウド基盤側で決めることはできない．場合によっては異なる大学で同一の VLAN-ID を使ってインタークラウド基盤側にアクセスしてくることがあり得る．このため SINET に入って来るパケットに付与されている VLAN-ID を図 6-29 に示すように，SINET 内においてインタークラウド基盤側で管理しているユニークな VLAN-ID に変換する処理が必要となる．この変換指定は申請ベースで実施すうることもできるし，また SINET の持つオンデマンド L2 用 API 経由でも設定することができる．

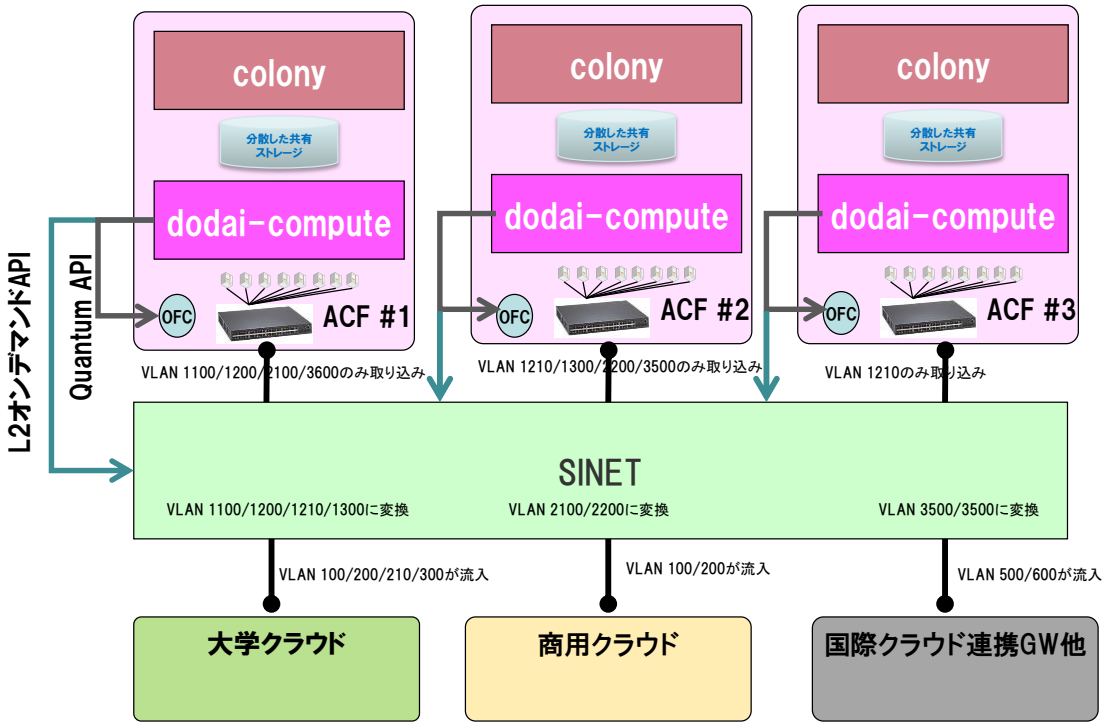
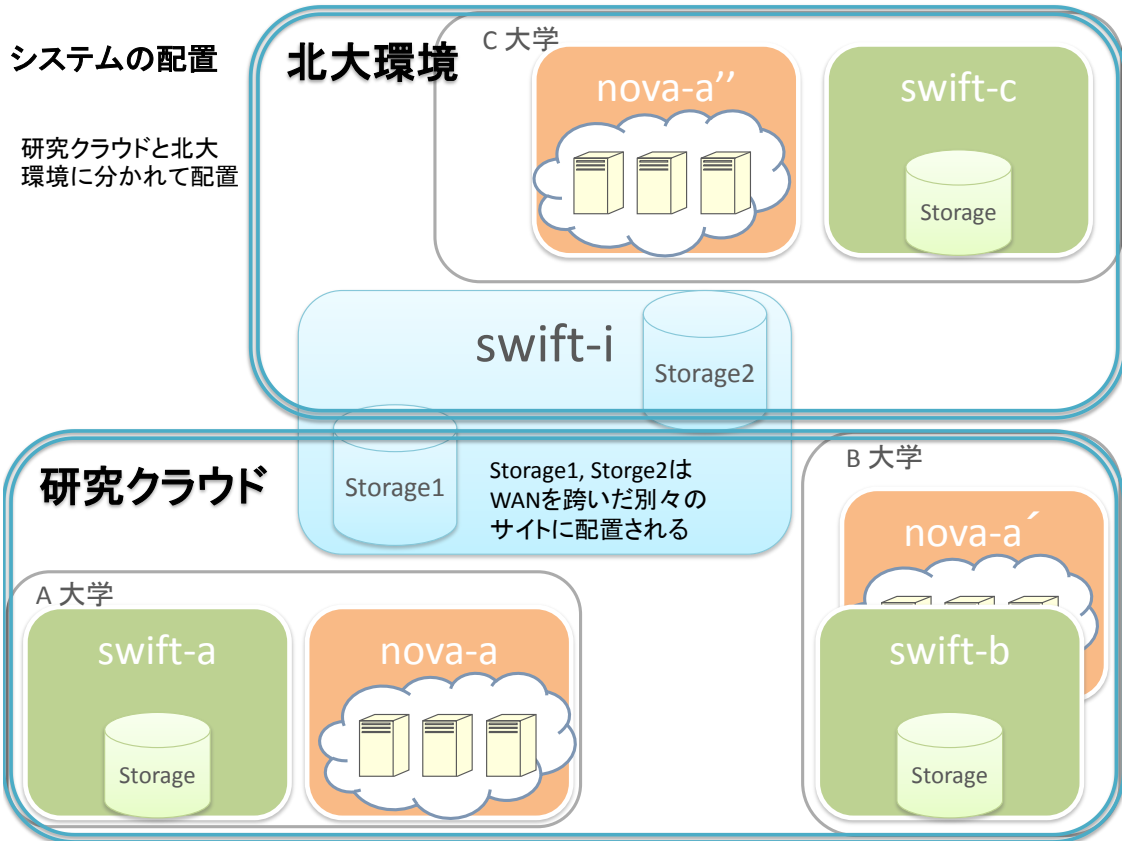


図 6-29　拠点を跨る動的配備実現方法（VLAN-ID 活用方法）

この実装方法に従っている限り，クラウドへ届く VLAN-ID 付きのパケットについて OpenFlow の閉域網との接続をする部分は基本アーキテクチャと同一であるため，問題無く動作する．　今後，インタークラウド基盤プロトタイプを構築した際に，再確認する．

上記方針に従ったインタークラウド基盤実験システム構成を図 6-30 に示す．



システム構成マシン一覧

- 研究クラウド

swift-a	nova compute	cn01061002	10.1.4.30
		cn01061204	10.1.4.31
swift-b	nova compute	cn01061004	10.1.4.32
		cn01061203	10.1.4.33
swfit-i		sn01030401	10.1.4.34
		sn01030602	10.1.4.35
		sn01030601	10.1.4.36

- 北大環境

swift-c	nova compute	hokudai001	10.1.4.132
swift-i		hokudai002	10.1.4.133

図 6-30　インタークラウド基盤実験システム構成

6.5.2 クラウドマイグレーション実験ユースケース

インタークラウド基盤実験システム上でのアプリケーション実証検証として，災害時や法定停電時に活用できるクラウドマイグレーションを選択し，そのアプリケーション開発と実験を行った．以下が実験の対象としたユースケースである．

ユースケース名

クラウド基盤全体をサイト A からサイト B へマイグレーションする

概要

サイト A で運用しているクラウド基盤（openstack)を，設定や登録されているデータはそのままに他のサイト B にマイグレーションさせる．

これは，法定点検による停電や災害などにより，元のサイト A でクラウド基盤の運用が困難となった場合を想定している．マイグレーションのための手間や時間を削減するため，マイグレーションは可能な限り自動化することが求められる．

アクター

サイト A プラットフォーム
サイト B プラットフォーム
サービス利用者
サービス運用者

事前条件:

- サイト A プラットフォーム上で稼働する複数のサーバマシンが連携して動作し，サービスを構成・提供している．
- サーバマシン群はサイト A のクラウド基盤（openstack）上に構築されている．
- サイト A とサイト B をまたがった広域化対応 swift が置かれている．
- 広域化対応 swift 上にはサービス停止直前までのサービス運用継続のための永続化データ（マシンイメージ，サービスの設定情報，ボリュームデータを指す）がバックアップされている．

メインフロー:

1. サービス運用者がサイト A でのクラウド基盤サービス継続不可を判断する．
2. サービス運用者が DNS の変更を行う．
3. サービス運用者がサイト A 上のサービスおよびクラウド基盤サービスを停止させる．
4. サービス運用者が，広域化対応 swift 上に置かれた永続化データを元にマイグレートを行い，サイト B プラットフォーム上でサイト A と同等のクラウド基盤を復元する．
5. サービス運用者がサイト B 上で，サービスを再開する．

6.5.3 マイグレーション要件

0penstack 環境のマイグレーション（環境移行）を行うにあたり，以下の 6 点が移行元と移行先で同一であるか，もしくは同意義であることが満たされなければならない．

1. マシン，ネットワーク構成

2. 設定ファイル内容
3. データベース内容
4. ボリュームデータ
5. マシンイメージ
6. カーネル上のネットワークデバイスの状態

1.は物理的なマシンとネットワークのスペック・台数，それらを接続するネットワークトポロジー等の構成が，移行元と移行先で同一でなければならないということである．

2.は設定ファイルの記述内容が同一でなければならないことを意味する． ただし，設定ファイル内にはIPアドレスやホスト名,それを含むURLなどの環境固有の情報が含まれているため，それらが移転先の環境において，意味として移転元と等価になっていなければならない． つまり，IPアドレスは同じものを付与できないが，それがシステムの構成や機能の上で同等の機能となる値であればよいということである．

3.はソフトウェアの使用する内部データベース内のデータが同一でなければならないことを意味する．これも2と同様に機能的に等価でなければならない．データの内容は現在の一時的な状態を記録しているものと，登録・変更・削除などの継続的なメンテナンスを受けつつ，永続的に使用されるものなどがある．永続的なデータはマイグレーションの対象であるが，一時的な状態の記録は対象ではない．

2,3 の移行を実現するには，設定ファイルやデータのコピーと必要な部分の取捨選択，部分的な書き換えを行う必要がある．

4.はボリューム（インスタンスに接続されるブロックデバイス）のデータが同一でなければならないことを云う．ボリュームはデータベース上の記録と，ボリュームの実体そのものからなる．マイグレーションを実現するには，ボリュームの実体をサイト間で可搬性のある形で持ち，移行元と移行先の間でコピーできることが前提となる．

5.はマシンイメージのバックエンドストレージとして，移行元・移行先から同様にアクセス可能なswiftを使用することを前提とするため，2,3に含まれるマシンイメージ管理の設定・データともに移行が実施されるものとする．

6.はopenstackの仮想ネットワーク管理機能により作成されるnetfilter，ブリッジデバイス，VLAN等の状態が同一でなければならないことをいう．これらは，2,3の移行が成功すれば自動的に設定されるが，注意点として仮想ネットワーク管理機能(nova-network)は，停止してもこれらの設定を削除しないため，マイグレーション前後の古い状態と新しい状態で設定が衝突する恐れがある．このため，マイグレーション手順の中にマシンのリブートによるネットワーク状態の初期化が必要となる．

6.5.4 マイグレーション対象情報

openstack computeの設定ファイル，DB等構成情報の中で，以下の情報がマイグレーションに必要である．

- ユーザ情報
- プロジェクト（テナント）情報

- キーペア（ssh 鍵情報）情報
- セキュリティグループ（フィルタリング情報）情報
- 内部利用ネットワーク設定
- マシンイメージ
- ボリューム

仮想マシンに割り当てる公開 IP アドレスである floating ip に関するデータの移行は行わない．これは移行先の環境に固有の情報が必要であり，自動化できないためである．

6.5.5 マイグレーション実施の流れ

マイグレーション実施の流れを以下に記述する．なお，移行先も nova,keystone,glance のデプロイは完了しているものとする．openstack 環境が複数台で構成されている場合には，移行元と移行先の対応し合うマシンで下記手順を行う．

移行元

1. 全インスタンスからボリュームを detach する．
2. 全インスタンスを停止する．
3. 設定ファイルのコピーを取得する．
4. データベースのダンプを取得する．
5. ボリュームの停止手順を実施し，ボリュームファイルを swift にアップロードする．
6. 設定ファイル，データベースダンプを移行先へコピーする．

移行先

1. 移行元からコピーした設定ファイルを，各々所定のディレクトリに置く．
2. 設定ファイル中の変更が必要な部分を書き換える．
3. データベースを初期化する．
4. 移行元からコピーしたデータベースダンプを，リストアする．
5. データベース中で変更が必要な部分を更新する．
6. マシンをリブートする．
7. ボリュームファイルを swift からダウンロードする．
8. ボリュームの認識手順を実施する．
9. インスタンスを起動する．
10. インスタンスにボリュームを attach する．

6.5.6 検証 4 に関するまとめ

NII 千葉分館に構築したアカデミッククラウドプロトタイプシステム実装と SINET VPLS 機能を使って一ツ橋，西千葉と北海道大学間に形成した実験システムを使って，提案した基本アーキテクチャを評価した．第 4 章で述べた設計方針に対応して本プロトタイプ構築を通して検証できたものはデータセンタを跨いで，設計方針 1,2,3, 4, 5, 6 についてである．

本節で説明したプロトタイプの位置づけは以下のようになる．また，検証対象は図 6.5-4 の通りである．

(1) 検証評価したい項目

6.2-6.4 の知見を活用して総合的なクラウド連携基盤を構築/検証すること

- 地域分散したクラウド基盤へのリソース割り付けの柔軟性
- 地域分散環境でのベアメタルなマシンクラスタ構築

- 地域分散環境でのクラウド間分離
- 地域分散環境でのクラウド構築/運用の簡単化
- 地域分散オブジェクトストレージとの連携
- 外部認証の利用

(2) 検証評価方法

dodai と colony および SINET VPLS 接続サービスの連携によるインタークラウド基盤実験システムの構築とそれを用いたクラウドマイグレーションの実現

- SINET 側の VLAN-ID と OpenFlow 閉域網の動的マッピング
- Dodai-deploy を SaaS 化した Install as a Service による遠隔サイトでのクラウド基盤構築
- dodai/colony の学認連携

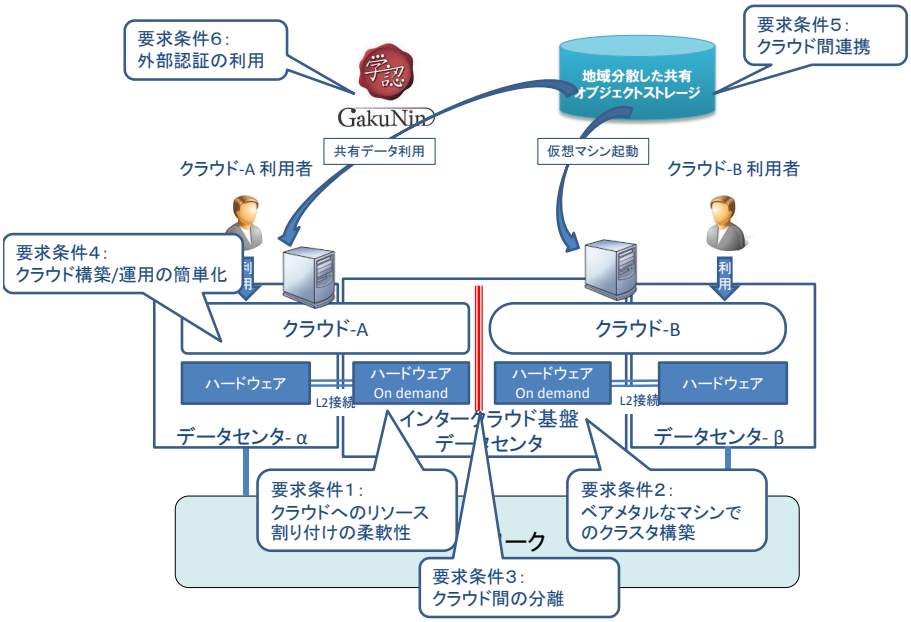


図 6-31　検証 4 の対象項目

第7章　関連研究

7.1　クラウド連携アーキテクチャに関する関連研究

7.1.1　Future Grid

大規模並列・分散システムの研究用に作られたテストベッドである Grid5000 上に構築されたハイパフォーマンス・グリッドテストベッドである．再生可能な実験のための管理機能を提供する． アプリケーションまたは分散/グリッド/クラウドミドルウェアの実験を簡単に定義，実行，そして繰り返し行えるように構成されている．

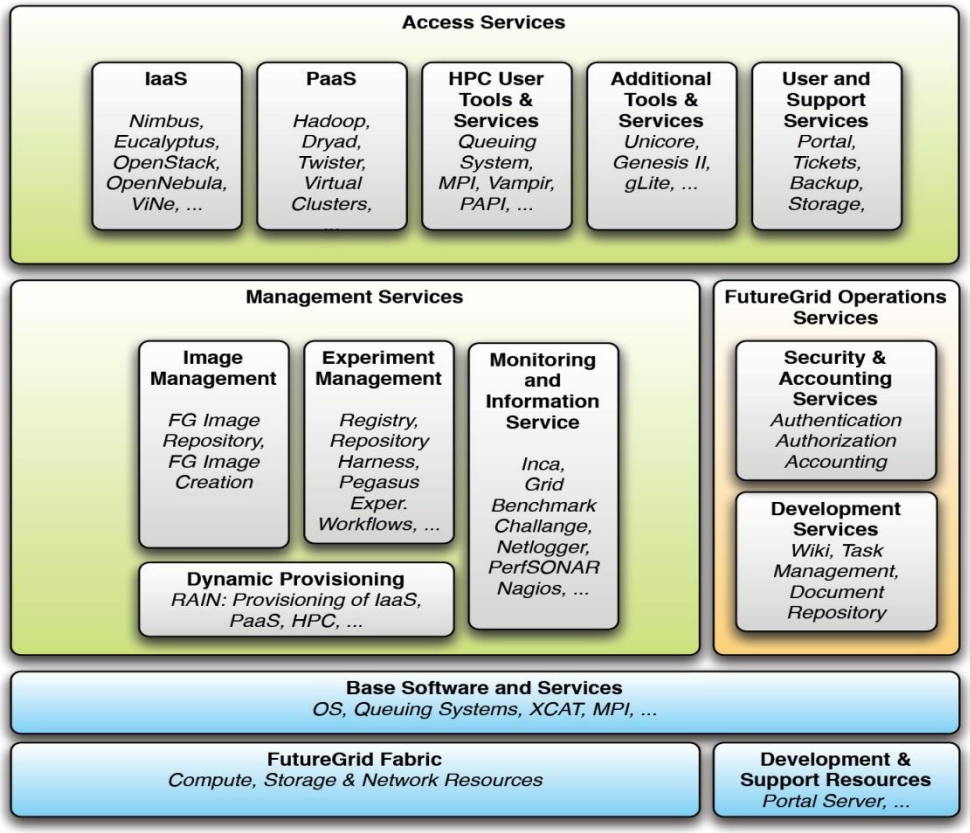


図 7-1　FutureGrid のアーキテクチャ

Future Grid はテストベッド提供を目的としており，各機関のクラウドとの連携を目指したものではない．

7.1.2　OpenCirrus

クラウドコンピューティングのシステムレベルの研究促進を目的として構成されたコンソーシアムである．クラウド向けのオープンソーススタックおよび API を開発し，そのスタックに沿ったサイト群をテストベッドとして提供する．OpenCirrus は各サイトの標準スタックを規定することでの連携を目指しており，各サイトのクラウド基盤をそのまま活用することを考慮していない．

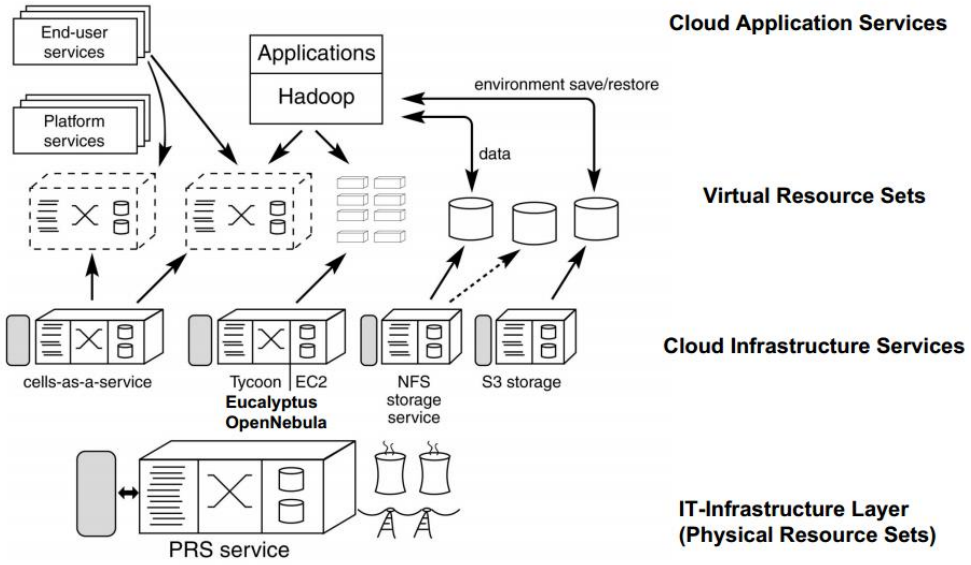


図 7-2　OpenCirrus のソフトウェアスタック

7.2　コンピュートリソース配置に関する関連研究

7.2.1　Zoni

Zoni[34]は，Open Cirrus サイトのソフトウェアキテクチャの基盤サービスとして，クラスタ内の物理リソースの管理を行い，ベアメタルサービスアクセスを提供する．

以下は Zoni の主な 5 つの機能である．

- 隔離
  - 複数のミニクラスタがお互い干渉せずに共存可能とする
- アロケーション
  - ユーザに物理リソースをアサインする
- ドメイン内での主要なソフトウェアのプロビジョニング
  - 特定 OS のインストールまたはブート
- デバッグ
  - 割当られたノードのデバッグ（コンソールアクセス）
- 管理
  - 境界外のサーバの管理（リモートパワーコントロール）

Zoni は，以下のコンポーネントで構成されている．

- DHCP サーバ
- PXE サーバ
- HTTP サーバ
- イメージストア（NFS）
- DNS サーバ（オプション）
- 設定可能なスイッチ（レイヤー2，VLAN サポート）
- 管理とデバッグを可能にするリモートハードウェアアクセスメソッド
  - IPMI/iLO/DRAC

- IP-addressable PDU

Zoni は物理マシンを OpenCirrus の想定している上位スタックにそのまま貸し出すことになり，その上にクラウド基盤を自動構築し，さらには既存のクラスタとの L2 接続までを考慮されていない．

7.2.2 Mesos

Mesos[35]は，クラスタ管理ツールであり，分散型アプリケーションまたは複数フレームワークからのジョブ要求を振り分け処理するものである．

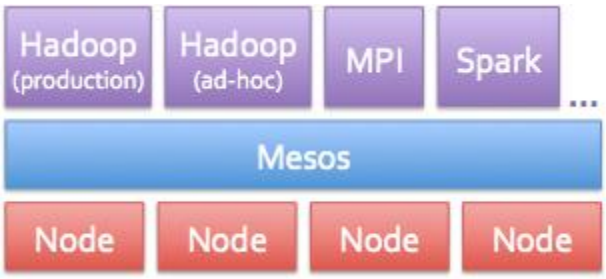


図 7-3　Mesos 概念図

Mesos を利用することにより，以下が可能となる．

- Hadoop，MPI，Spark，その他のフレームワークを，ノードを動的に共有したプール上で実行することができる．
- 同じクラスタ上で複数の Hadoop インスタンスを動作させ，製品生産と実験，あるいは Hadoop の複数のバージョンでさえも隔離することができる．
- 長期間継続するサービス（例：Hypertable や HBase）を，バッチアプリケーションやその間の共有リソースとして，同じノード上で動作させる．

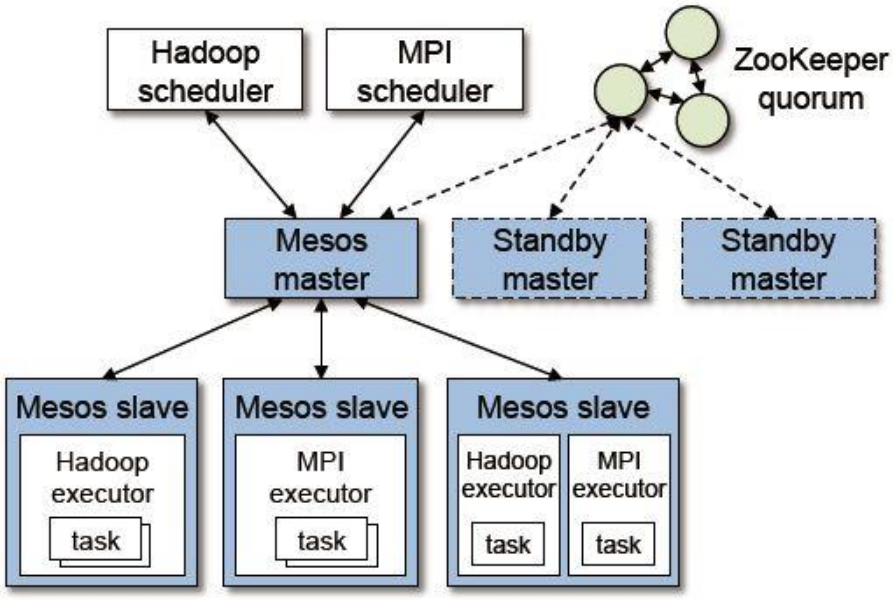


図 7-4　Mesos のアーキテクチャ概要図

（Hadoop と MPI の二つのフレームワークを動作させている）

図 7-4 では，Mesos の主要なコンポーネントが示されている．Mesos は，各クラスタノード上で走るスレーブデーモンを管理するマスタプロセスと，これらのスレーブ上でタスクを実行するフレームワークで構成される．

マスタは，「resource offer」を使って，フレームワーク全体にわたってきめ細かい共有を実施する．各 resource offer は，複数のスレーブ上のフリーのリソースのリストである．マスタは，フェアシェアリングや優先度などの組織的なポリシーにより，各フレームワークにいくつリソースを提供するかを決定する．多様なインターフレームワークアロケーションポリシのーセットをサポートするべく，Mesos では，組織が，プラガブルなアロケーションモデルを介して自身のポリシーを定義できるような仕組みとなっている．

Mesos 上で実行される各フレームワークは，次の二つのコンポーネントで構成される．一つ目は，マスタに登録し，リソースを提供される「scheduler」で，もう一つは，フレームワークのタスクを走らせるためにスレーブノード上で起動される「executor」プロセスである．マスタが各フレームワークにリソースをいくつ提供するかを決定する一方で，フレームワークの scheduler は，提供されたリソース中のどれを利用するかを選択する．フレームワークが提供されたリソースを受け入れると，フレームワークは，Mesos にそれらのリソース上で立ち上げたいタスクの詳細を送る．

Mesos は物理マシンをジョブで区切って分割使用する方式で，我々の方式に比べてリソース割り付けの粒度が細かく，その分利用効率が高い．反面，高セキュリティの情報を扱うのが難しいという欠点がある．

7.2.3　その他関連プロジェクト

その他関連プロジェクトとして，VCL[36]，crowbar[37]，MAAS[38]，juju[39]，Lucie[81,82]がある．

Dodai-compute と同様にベアメタルを扱えるソフトウェアとしては，VCL，crowbar，MAAS 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で物理マシンを扱うため 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 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．Lucie はグリッド環境構築の効率化を目的として，複数拠点に分散配置されたクラスタの効率的な管理を行うために開発されたものである．クラウド基盤を構築するためにも適用できる可能性はあるけれど，クラウド基盤特有の，その上に仮想ネットワークを構築するために必要なクラスタ間のネットワーク分離が考慮されていないため，クラウド基盤間で利用する VLAN-ID のレンジや MAC アドレスなどを事前に調整する必要が出るため実用的な適用には向かない．また，Lucie はサイトを跨った並列インストールには対応しているけれど，インストレーションの際に puppet のみを用いているため，さらに細粒度の各サイト内のノード群への並列インストールへは対応していない．

7.3 クラウドストレージに関する研究

7.3.1 Storage Resource Broker(SRB)

Storage Resource Broker(SRB)[89] は，San Diego Supercomputer Center(SDSC) が開発しているデータグリッドシステムである．SRB は，Resource Broker の名前の通り，資源へのアクセスの仮想化を行い，分散する異機種環境にある資源への仲介を行う．SRB は，メタデータを用いて資源への仲介をしている．SRB では，ユーザが識別しやすいデータ属性であるファイル名と，SRB システム内のユニークな識別子をマッピングする．マッピング情報とデータの物理的な位置情報は，MCAT(Metadata CATalog)サーバと呼ばれるメタデータ管理サーバ上に保存される．

SRB のメタデータは，Dublin Core メタデータと科学技術の専門分野のメタデータから構成されている．

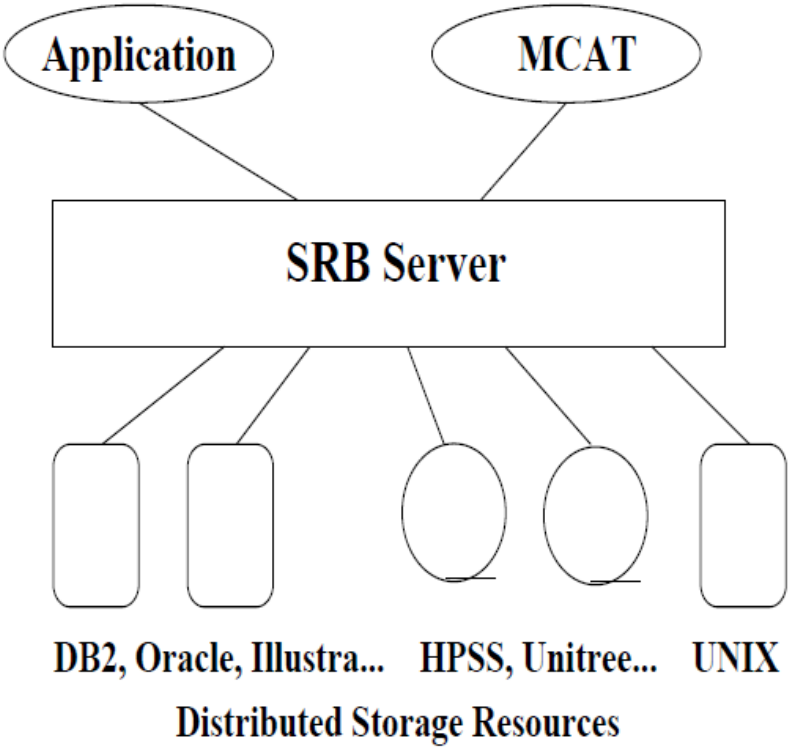


図 7-5　SRB の位置づけ

7.3.2　Gfram

Gfarm[90] は，産業総合研究所グリッド研究センタが開発している広域ネットワーク環境におけるグリッドファイルシステムである．Gfarm は，farm は，NFS では想定されていない大規模な分散ファイルシステム構築を目的に設計され，グローバルファイルシステムを提供する．

Gfarm では，データを実データとメタデータに分離して管理する．

MDS(Meta Data Server)は Gfarm ファイルシステムのメタデータを管理する．ホスト上では，メタデータサーバの gfmd と，LDAP サーバ(slapd)，PostgreSQL サーバ(postmaster) などのバックエンド・データベースで構成されている．

I/O Server は，Gfarm ファイルシステムのデータ記憶領域を提供する．

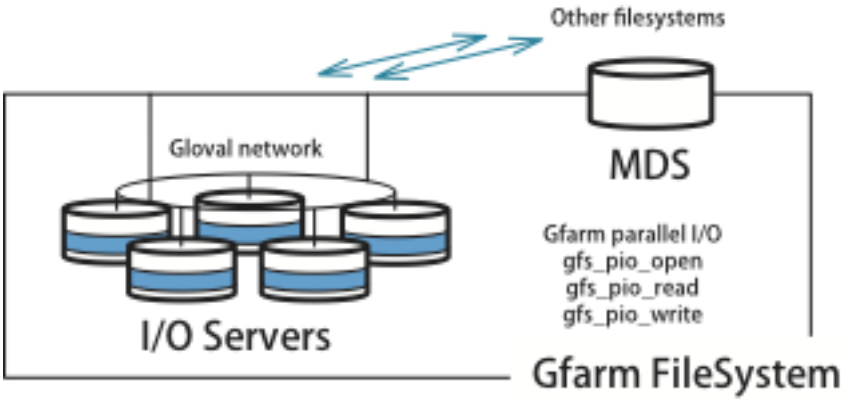


図 7-6　Gfarm のアーキテクチャ

SRB も Gfarm もともにファイルシステムの広域化で保存場所を保持しているメタデータ管理サーバを必要としており，その広域網上の配置に工夫が必要である．一方，提案の広域オブジェクトストレージについては，保存場所についての認識のためのメタサーバを必要としないため，オブジェクトストレージシステムの構成変更への柔軟性が高い．

第8章　　議論

提案したアプローチとそれに基づくアーキテクチャを適用することで，従来のアプローチで課題であった標準化策定およびそれに合わせたソフトウェア追加開発という実現上の困難性やフェデレーションポータルによる最大公約数的機能のみの提供でクラウド連携の限られた有効性を克服することができた．すなわちクラウド連携の実現性も有効性も高いクラウド連携によるアカデミックコミュニティクラウドの実現に目途がついた．

本章では実現性および有効性について前章までに述べた内容をまとめ上記主張を検証する．

【実現性】

ベアメタルマシンの起動・制御に関するすでに普及している標準仕様（PXEboot，IPMI）のみを前提としたクラウド連携アーキテクチャについて実装し，実用的な想定性能を達成することを確認した．　さらにこの際開発したソフトウェアを dodai/colony というオープンソースプロジェクトを通じて，公開した．このため我々の評価環境と同等汎用サーバを用いれば同様のクラウド連携が実現可能である．

但し，現状 dodai の実装の際には特定の OpenFlow Controller を前提とした実装となっているため，この部分をさらに抽象的なネットワークインタフェース（例えば quantum）を使って改善すれば，さらに実現性は向上する．

【有効性】

各クラスタ上にクラウド基盤を動的にデプロイするため，それぞれのクラウド基盤が持つ個別機能は全て利用可能であるし，利用に最適なデプロイの構成が定義できる．実際に実装したデプロイ対象クラウド基盤は OpenStack，Eucalyptus，Hadoop，GridEngine の 4 種であるけれど，対象クラウド基盤をさらに増やすための手順も確立した．このため，このアプローチによるクラウド連携の有効性はポータルによるそれよりも高いと言える．

以下に実現性および有効性に関する議論についてまとめる．

8.1　　実現性

実現性に課題を持つ標準化によるアプローチでは，具体的には実現までにかかる時間とコストが問題であった．提案アプローチとアーキテクチャについてこれら二つの視点からまとめる．

8.1.1　　実現までに要する時間

提案アプローチおよびアーキテクチャを適用したアカデミックコミュニティクラウドを，以下のステップを踏むことで，実現できる．

（1）ステップ 1

提案アプローチの特徴である動的にクラウド基盤を構築するためには技術的な課題以外にソフトウェアライセンス上の課題を解決しておく必要がある．すなわち，ライセンス条件がきびいしいソフトウェアを用いたクラウド基盤を用いることはこの動的構築を行う妨げとなる．一つの回避方法がオープンソースソフトウェアの利用によるクラウド基盤を用いることである．一方，オープンソースソフトウェアの利用によるクラウド基盤については実用に耐えるレベルのソフ

SRB も Gfarm もともにファイルシステムの広域化で保存場所を保持しているメタデータ管理サーバを必要としており，その広域網上の配置に工夫が必要である．一方，提案の広域オブジェクトストレージについては，保存場所についての認識のためのメタサーバを必要としないため，オブジェクトストレージシステムの構成変更への柔軟性が高い．

第8章　　議論

提案したアプローチとそれに基づくアーキテクチャを適用することで，従来のアプローチで課題であった標準化策定およびそれに合わせたソフトウェア追加開発という実現上の困難性やフェデレーションポータルによる最大公約数的機能のみの提供でクラウド連携の限られた有効性を克服することができた．すなわちクラウド連携の実現性も有効性も高いクラウド連携によるアカデミックコミュニティクラウドの実現に目途がついた．

本章では実現性および有効性について前章までに述べた内容をまとめ上記主張を検証する．

【実現性】

ベアメタルマシンの起動・制御に関するすでに普及している標準仕様（PXEboot，IPMI）のみを前提としたクラウド連携アーキテクチャについて実装し，実用的な想定性能を達成することを確認した．　さらにこの際開発したソフトウェアを dodai/colony というオープンソースプロジェクトを通じて，公開した．このため我々の評価環境と同等汎用サーバを用いれば同様のクラウド連携が実現可能である．

但し，現状 dodai の実装の際には特定の OpenFlow Controller を前提とした実装となっているため，この部分をさらに抽象的なネットワークインタフェース（例えば quantum）を使って改善すれば，さらに実現性は向上する．

【有効性】

各クラスタ上にクラウド基盤を動的にデプロイするため，それぞれのクラウド基盤が持つ個別機能は全て利用可能であるし，利用に最適なデプロイの構成が定義できる．実際に実装したデプロイ対象クラウド基盤は OpenStack，Eucalyptus，Hadoop，GridEngine の 4 種であるけれど，対象クラウド基盤をさらに増やすための手順も確立した．このため，このアプローチによるクラウド連携の有効性はポータルによるそれよりも高いと言える．

以下に実現性および有効性に関する議論についてまとめる．

8.1　　実現性

実現性に課題を持つ標準化によるアプローチでは，具体的には実現までにかかる時間とコストが問題であった．提案アプローチとアーキテクチャについてこれら二つの視点からまとめる．

8.1.1　　実現までに要する時間

提案アプローチおよびアーキテクチャを適用したアカデミックコミュニティクラウドを，以下のステップを踏むことで，実現できる．

（1）ステップ 1

提案アプローチの特徴である動的にクラウド基盤を構築するためには技術的な課題以外にソフトウェアライセンス上の課題を解決しておく必要がある．すなわち，ライセンス条件がきびいしいソフトウェアを用いたクラウド基盤を用いることはこの動的構築を行う妨げとなる．一つの回避方法がオープンソースソフトウェアの利用によるクラウド基盤を用いることである．一方，オープンソースソフトウェアの利用によるクラウド基盤については実用に耐えるレベルのソフ

トウェア品質を持つことを検証する必要がある．
オープンソースソフトウェアで実用的なクラウド基盤を構築する必要がある．
筆者らは教育クラウド　edubase Cloud に用いるクラウド基盤をオープンソフトウェアにより構築し，2 年間を超えて実用運用することで，この実用性を検証してきた．
（2）ステップ 2 （設計方針 1,2,3,4 の実現）
本研究で提案しているアーキテクチャのハブとなるインタークラウド基盤を第 5 章で説明した設計方針 1，2，3 ，4 を満たす形で実現する．
これは研究クラウド　gunnii およびそれを SINET に接続したプロトタイプで実現できている．しかも，NII の研究クラウドとして実装し，クラウドサービスを 2012 年 7 月より開始しその実現性は検証できている．
（3）ステップ 3（設計方針 5 の実現）
各大学のプライベートクラウドの延伸としてステップ 1 で実現したインタークラウド基盤内に動的に構築できる．この際，このクラウド基盤の上にマシンイメージをインタークラウドストレージサービスから呼び出し起動することで起動時間の短縮が実現できる．
これは第 7 章のインタークラウドストレージサービスの実現とインタークラウド基盤実験システム上のクラウドマイグレーションの実施により確認できた．
目標とした 20,000 コアには満たないけれど 1500 コア規模のインタークラウド基盤が構築できることと，それがスケールする方式であることを確認した．このため目標規模のインタークラウド基盤が構築可能である見通しが得られた．また，2,000 コア規模のクラウド基盤およびその上の仮想マシン起動までを 1 時間以内に目標性能については，クラスタ確保まで 10 秒以内（デフォルト OS 指定の場合），　クラウド基盤構築（OpenStack の場合）5 分以内，仮想マシン立ち上げ　4 分以内，を達成しトータルで 10 分以内の立ち上げが実現できる見通しを得た．すなわち，技術的には本論文で提案しているアーキテクチャを用いることで第 1 章で述べた規模および性能目標を満たすアカデミックコミュニティクラウドでのクラウド連携向けインタークラウド基盤を構築できることが分かった．
（4）ステップ 4（設計方針 6 の実現）
各大学のプライベートクラウドの延伸としてステップ 1 で実現したインタークラウド基盤内に動的に構築できる．この際，インタークラウド基盤のアクセス制御のための ID 管理は学認により実施できることで利便性を持たせることができる．
これは研究クラウド　gunnii での学認対応やインタークラウド基盤実験システムでの学認対応で実現でき，実用に供している．
以上のステップ経てアカデミックコミュニティクラウドに向けたプロトタイプ構築を通じ，提案アプローチおよびアーキテクチャの実現性の高さを検証できた．すなわち，技術的にはすでにアカデミックコミュニティクラウドにおけるクラウド連携は実現できており，後は実際のインタークラウド構築基盤を構築すれば，各大学からの学認による認証を経た利用ができる状態にある．具体的にクラウド連携の対象とできるクラウド基盤は現在 Eucalyptus, OpenStack, Hadoop, GridEngine であるが，これは現在 dodai-deploy がサポートしているものという制限であり，この対象クラウド基盤を今後必要に対応して増やして行くことは容易である．
すなわち，図 8-1 に示すように提案アプローチとアーキテクチャについては技術的な実現性は確認されており，普及のためのインタークラウド基盤センタは 2013 年夏に完成予定でそのプロトタイプ開発に着手している．後は実証実験を実施し，その有効性への賛同を増やしつつ普及する期間が 1-2 年程度必要であると考えられる．一方，標準化アプローチにおいては仕様設計につ

いては完了しているプロジェクトもあるものの技術検証や実装に着手しているプロジェクトは無く後何年先にアカデミックコミュニティクラウドで採用できるものが出てくるか不明である．ポータルを前提としたアプローチについてはすでに実用レベルのサービスがあり活用が進んでいる．

クラウド標準化を前提としたアプローチ	ユースケース分析，機能要件分析	インタフェース仕様書		
ポータルを前提としたアプローチ	各種商用サービス，商用パッケージ，OSS企画	各種商用サービス，商用パッケージ，OSS設計	商用サービス，商用パッケージ，OSS実装	特定のクラウド基盤向け商用マルチクラウドサービス
提案アプローチ	ユースケース分析，機能要件分析	アーキテクチャ設計	プロトタイプ実装	テストベッド利用から実施中

図 8-1 実現性（実現までに要する時間）

8.1.2 実現までに要するコスト

ベアメタルマシンの起動・制御に関するすでに普及している標準仕様（PXEboot，IPMI）のみを前提としたクラウド連携アーキテクチャについて実装し，実用的な想定性能を達成することを確認した．さらにこの際開発したソフトウェアを dodai/colony というオープンソースプロジェクトを通じて，公開した．このため我々の評価環境と同等汎用サーバを用いれば同様のクラウド連携が実現可能である．すなわち，提案アプローチとアーキテクチャを採用すれば，オープンソースソフトウェアと汎用ハードウェアを用いたシステムインテグレーションを実施することで実現できるため，特別な追加開発など実施する必要がない．クラウド連携の対象とできるクラウド基盤は現在 Eucalyptus，OpenStack，Hadoop，GridEngine であるが，これは現在 dodai-deploy がサポートしているものという制限であり，この対象クラウド基盤を今後必要に対応して増やして行く必要があれば，そのパッケージの依存関係に関する情報とテストスクリプトを追加するコストのみである．但し，現状 dodai の実装の際には特定の OpenFlow Controller を前提とした実装となっているため，この部分をさらに抽象的なネットワークインタフェース（例えば quantum）を使って改善すれば，さらに実現性は向上する．一方，標準化によるアプローチでは第 3 章で述べた全ての設計方針に関する標準対応を各クラウド基盤側で実施する必要があり，そのコストは

設計方針 x クラウド基盤数になる．さらに，これの開発はクラウド基盤側の変更や標準化案の変更の毎に発生する．

また，ポータルによるアプローチでは対象とするクラウド基盤毎の差分吸収のための開発が必要となる．

8.2 有効性

有効性に課題を持つポータルを利用するアプローチでは，実現できる機能（クラウド基盤機能，クラウド基盤連携機能，ユースケース実現）に問題があった．提案アプローチとアーキテクチャについて実現できる機能の視点からまとめる．

8.2.1 クラウド基盤機能の有効性

提案アプローチでは，クラウド基盤を動的に構築するためにネイティブな機能をそのままサポートできる．そのため，各クラウド基盤のエコシステムとして形成されている各種ユーティリティ機能もそのまま利用できる．一方，ポータルを利用するアプローチでは対象とするクラウド基盤の持つ機能の最大公約数的機能を定義して，それのみが提供可能になる．従って各クラウド基盤のエコシステムとして形成されている各種ユーティリティ機能は利用できない．標準化によるアプローチでは標準化されたクラウド基盤機能は全て活用できる．

8.2.2 クラウド基盤連携機能の有効性

提案アプローチでは，第 5 章で述べた以下のインタークラウド基盤への設計方針は全て実現できる．

設計方針１(R1)：クラウドへのリソース割り付けの柔軟性
設計方針２(R2)：ベアメタルなマシンでのクラスタ構築
設計方針３(R3)：クラウド間の分離
設計方針４(R4)：クラウド構築/運用の簡単化
設計方針５(R5)：地域分散型オブジェクトストレージ
設計方針６(R6)：外部認証の利用

一方，ポータルを利用するアプローチでは各クラウド基盤とのネットワーク設定や認証連携については個別に解決れば連携が可能である．標準化によるアプローチでは標準化されたクラウド基盤連携機能は全て活用できる．

8.2.3 ユースケース実現に関する有効性

各クラスタ上にクラウド基盤を動的にデプロイするため，それぞれのクラウド基盤が持つ個別機能は全て利用可能であるし，利用に最適なデプロイの構成が定義できる．実際に実装したデプロイ対象クラウド基盤は OpenStack，Eucalyptus，Hadoop，GridEngine の 4 種であるけれど，対象クラウド基盤をさらに増やすための手順も確立した．このため，このアプローチによるクラウド連携の有効性はポータルによるそれよりも高いと言える．

基本アーキテクチャについては NII の研究クラウドとして実装し，クラウドサービスを 2012 年 7 月より開始しその有効性を示せている．具体的には，既存クラスタ内の NIS サーバや NFS サーバとの接続について，利用者側では何も，使っているリソースがクラウド内のものであるか，新規に購入後，直接自分のネットワークに接続した際の差が感じられないことを確認している．また，教育クラウドの#16 番目以降のミニクラウドを動的に構築し，さらにはその Node Contoller

数を動的に変化させて利用している．

また，6.5 で示した通り別の拠点のリソースを借りてクラウド全体をマイグレーションすることがコマンド一つで可能となっている．このことにより例えば法定停電の際のクラウド基盤サービス継続の実現の可能性を示せた．

具体的には，各クラウド基盤本来が持つ特徴的な機能提供ができない点を克服できることを第 6 章のプロトタイプシステム上の実験により U1-U4, U7 のユースケースを満たすことを検証した．

（１）急激な負荷増加に対する性能を保証（U1）

クラウド基盤(OpenStack)で急激に負荷が増加した場合，インタークラウド基盤内に設置した dodai-compute を利用し，インタークラウド基盤データセンタ内に必要な台数のベアメタルノードを確保し，それをクラスタ化し，自らのクラウド基盤を構成しているネットワークを L2 延伸する．この L2 延伸部分については事前に SINET 側に VPLS 申請し利用する VLANID を確保した．この際，ベアメタルを獲得するためのインタークラウド基盤のサービスを利用する際に学認 ID でサービス利用した．その後，dodai-deploy を利用し，既存のクラウド基盤を構成するノードと同一 L2 ネットワークに組み込まれたベアメタルノードに既存のクラウド基盤(OpenStack)のコンピュートノード用ソフトウェア nova-compute をデプロイメントし，それを既存クラウド基盤に組み込んだ．その拡張されたクラウド基盤上で仮想マシンを起動する際にも colony 内に保存したマシンイメージをした．

（２）遅延に対する性能保証（U2）

インタークラウド基盤内に設置した dodai-compute を利用し，インタークラウド基盤データセンタ内に必要なベアメタルノードを確保し，それをクラスタ化し，自らのクラウド基盤を構成しているネットワークを L2 延伸した．この L2 延伸部分については事前に SINET 側に VPLS 申請し利用する VLANID を確保した．この際，ベアメタルを獲得するためのインタークラウド基盤のサービスを利用する際に学認 ID でサービス利用した．その後，dodai-deploy を利用し，既存のクラウド基盤を構成するノードと同一 L2 ネットワークに組み込まれたベアメタルノードに既存のクラウド基盤のコンピュートノード用ソフトウェアをデプロイメントし，それを既存クラウド基盤に組み込む．colony を利用して，試験用仮想マシンに対応するマシンイメージスナップショットを地域分散型オブジェクトストレージに格納しておき，それをインタークラウド基盤内で起動した場合にレイテンシが小さくなることを確認した．

（３）災害や大規模故障発生に対する可用性保証，サービス継続（U3, U4）

クラウド基盤(OpenStack)の管理情報（ユーザ情報，クラウド基盤構成情報，マシンイメージスナップショット，ボリュームスナップショット）について colony を用いて地域分散型オブジェクトストレージに格納した．次に，インタークラウド基盤内に設置した dodai-compute を利用し，インタークラウド基盤データセンタ内にクラウド基盤サイズに合わせて必要なベアメタルノードを確保し，それをクラスタ化した．この際，ベアメタルを獲得するためのインタークラウド基盤のサービスを利用する際に学認 ID でサービス利用した．その後，dodai-deploy を利用し，既存のクラウド基盤を新規に構築したクラスタ上に再現する．この時，colony を用いて，前に蓄積してあったクラウド基盤構成情報を用いる．さらに再構築したクラウド基盤上に colony を用いてクラウド基盤のユーザ管理情報やマシンイメージスナップショット，ボリュームスナップショットをリストアする．該当仮想マシンを起動することでサービス再起動を行えた．

（５）特殊リソースを利用する計算処理（U7）

dodai-compute を利用し，特殊リソース近くにあるという想定のインタークラウド基盤データセンタに依頼して計算処理必要なベアメタルノードを確保し，それをクラスタ化し，自らのクラ

ウド基盤を構成しているネットワークを L2 延伸する．この際，ベアメタルを獲得するためのインタークラウド基盤のサービスを利用する際に学認 ID でサービス利用する．その後，dodai-deploy を利用し，既存のクラウド基盤を構成するノードと同一 L2 ネットワークに組み込まれたベアメタルノードに既存のクラウド基盤のコンピュートノード用ソフトウェアをデプロイメントし，それを既存クラウド基盤に組み込む．colony を利用して，利用者の計算処理に必要なマシンイメージスナップショットを地域分散型オブジェクトストレージに格納してき，新たに構築したクラウド基盤側で起動する際にこのスナップショットを利用した．

以上に加え，元々のアーキテクチャの特徴である各アカデミッククラウドを延伸するという特徴により元々のアカデミッククラウドの持つ機能はそのまま使えると言う有効性がある．また，L2 接続により遠隔地へのクラウド基盤の延伸であるということを活用することで，クラウド基盤内での仮想マシンのライブマイグレーションが容易に実現でき，これは遠隔地へのライブマイグレーションとなる．この遠隔地へのライブマイグレーションを活用することで負荷の動的バランシングやレイテンシの削減のための仮想マシン移動が実現可能となる．

また，第 6 章で述べたクラウド基盤全体の遠隔地へのマイグレーション時には仮想マシンで提供しているサービスをマイグレーション前に停止し，スナップショットを地域分散オブジェクトストレージサービスに格納して，マイグレーション先でスナップショットを再起動し，サービスを再開する必要があった．しかし，遠隔地へのライブマイグレーションとクラウド基盤全体のマイグレーションを組み合わせることでサービス無停止でのクラウド全体のマイグレーションが実現できる可能性がある．

	クラウド基盤機能	クラウド基盤連携機能	ユースケース実現
クラウド標準化を前提としたアプローチ	■ 標準化された機能の提供	■ 標準化された機能の提供	■ 急激な負荷増加に対する性能を保証(U1) ■ 遅延に対する性能保証(U2) ■ 災害や大規模故障発生に対する可用性保証，サービス継続(U3, U4)
ポータルを前提としたアプローチ	■ 各クラウド基盤の最大公約数的機能のみ提供可能	■ クラウド基盤間のネットワークは個別に設定が必要	各ポータルの実装に依存する
提案アプローチ	■ クラウド基盤のネーティブな機能をサポート ■ 既存ユーザ管理他ユーティリティ機能を利用可	■ リソース融通 (オンデマンドベアメタルクラスタ，クラウド基盤自動構築) ■ ネットワーク分離 ■ 分散オブジェクトストレージ (高速仮想マシンイメージ起動) ■ 認証連携	■ 急激な負荷増加に対する性能を保証(U1) ■ 遅延に対する性能保証(U2) ■ 災害や大規模故障発生に対する可用性保証，サービス継続(U3, U4) ■ 特殊リソースを利用する計算処理(U7)

図 8-2 有効性

第9章　結論

新しいアプローチによるインタークラウドを実現するためのプロトタイピングを行い，それが実際にこのアプローチでの実現の際に必要な機能を満たすことを確認すると同時に一つのユースケースである DR 向けのクラウドマイグレーションもこのプロトタイプ上で実現し，このアプローチの妥当性を確認した．その他クラウドバーストに関するユースケースも実際に実現し，NII の研究クラウドで利用している．この他のユースケースについても机上では実現できることが分かっているため，本アプローチは新しいアプローチとして先に進める価値がある．

また，本研究はネットワークおよび認証連携の進んだアカデミックコミュニティに関するクラウド連携のためのアーキテクチャを論じたものであるけれど，他のコミュニティについてもネットワークと認証連携が進み，同等のコンピューティングに対する要求が発言したコミュニティへの適用が可能となる．

本研究では従来の方法では困難であったクラウド型アプリケーションを他拠点のリソースを使っても継続実行を可能とすることを目的として新たなアーキテクチャを提案した．

第 1 章は，本研究の背景や目的について述べた．第 2 章では，クラウド基盤連携に関する代表的なユースケースについて述べた．第 3 章では，クラウドコンピューティングの分野における動向と，従来のアプローチが持つ課題について説明した．第 4 章で，この課題を解決する本研究で追求する新しいアプローチの提案を行った．さらに，このアプローチに沿ってインタークラウド基盤を構成する際に必要な設計方針を整理した．第 5 章では，この設計方針を達成するためのアーキテクチャを示した．第 6 章では，アーキテクチャの有効性と実用性を検証するために構成したプロトタイプについて説明した．第 7 章で，関連研究についてまとめ本提案の位置づけを述べた．第 8 章では，解決策として提案した，動的配備を行う際のアーキテクチャについて，実現性と有効性の観点から議論した．第 9 章で，研究を通じて得られた知見をまとめると同時に，提案手法の有効性および限界について示し，本論文を結んだ．

9.1　本研究の達成度

従来方式で課題であったクラウド連携に関する実現性と有効性の両方を同時に満たすアーキテクチャを提案し，議論の結果，アカデミックコミュニティクラウドを実現する際に使える実用的なアーキテクチャであると結論付けることができた．

9.1.1　実現性に関する議論

ベアメタルマシンの起動・制御に関するすでに普及している標準仕様（PXEboot，IPMI）のみを前提としたクラウド連携アーキテクチャについて実装し，実用的な想定性能を達成することを確認した．　さらにこの際開発したソフトウェアを dodai/colony というオープンソースプロジェクトを通じて，公開した．このため我々の評価環境と同等汎用サーバを用いれば同様のクラウド連携が実現可能である．

但し，現状 dodai の実装の際には特定の OpenFlow Controller を前提とした実装となっているため，この部分をさらに抽象的なネットワークインタフェース（例えば quantum）を使って改善すれば，さらに実現性は向上する．

9.1.2 有効性に関する議論

各クラスタ上にクラウド基盤を動的にデプロイするため，それぞれのクラウド基盤が持つ個別機能は全て利用可能であるし，利用に最適なデプロイの構成が定義できる．実際に実装したデプロイ対象クラウド基盤は OpenStack，Eucalyptus，Hadoop，GridEngine の 4 種であるけれど，対象クラウド基盤をさらに増やすための手順も確立した．このため，このアプローチによるクラウド連携の有効性はポータルによるそれよりも高いと言える．

9.1.3 議論の前提条件

上記議論は以下の前提条件のもとに実施されたという限界がある．

1．規模的制約

実利用の規模での運用まで含めた検証に基づくものではなく，プロトタイピング規模のシステムの上での実験結果に基づいている．

2．トポロジ的制約

インタークラウド基盤データセンタというアカデミックコミュニティクラウド全体のハブ的なハードウェア運用者の存在を前提としている．

3．スコープ的制約

現在，スコープを国内のアカデミックコミュニティに限定しているため SINET 接続を前提とした議論をしている．国際的なアカデミックコミュニティ連携や商用クラウドとの連携までスコープには入っていないという制約がある．

4．対象コミュニティの制約

本論文で提案したクラウド連携アーキテクチャを適用するコミュニティはアカデミックコミュニティとしている．このことでネットワークや認証基盤についての限定的な前提条件が仮定して議論が展開できた．アカデミックコミュニティ以外のコミュニティのためのコミュニティクラウド構築に，そのまま議論を適用できない．

9.2 学術的な貢献

日本のアカデミックコミュニティクラウドのように高速で L2VPN 機能を持つネットワークを前提と出来る場合には，既存のベアメタルマシンに対する起動・制御に関する普及している標準のみを前提にして，実現性と有効性のいずれも高くできるクラウド連携アーキテクチャを提案すると同時に，実装・評価し，それが実用的なアカデミックコミュニティ向けクラウド連携機構として使えることを示した．全体アーキテクチャについては，科学分野でのアプリケーションに適したアカデミックコミュニティクラウドアーキテクチャについてユースケース U7 に沿った国際会議発表を行った[14]．

全体アーキテクチャのみならずそれらを構成するコンポーネントレベルでもそれぞれ新規性のあるものとなっている．

9.2.1 既存クラスタを容易に拡張できるベアメタルクラウド

IaaS 向けオープンソースクラウド基盤で仮想化層(Hypervisor) を使って仮想マシン起動を実施している部分を改造し，標準インタフェースのみを用いて，物理マシンの管理・制御が行える

ようにしたCluseter as a Serviceについて国際会議発表発表を行った[21].

9.2.2 スケーラブルな自動デプロイサービス

デプロイするクラウド基盤の構造より事前に依存関係を抽出しておくことにより，デプロイ処理で並列動作可能な部分について最大限並列動作させる仕組みを MCollective と Puppet により実装し，スケーラビリティを確保することについて国際会議発表発表を行った [23].

9.2.3 ネットワークアウェアな地域分散型オブジェクトストレージサービス

広域ネットワーク上の位置情報を認識するようオープンソースオブジェクトストアシステムの，オブジェクト配置メカニズムを改善することとキャッシュメカニズムを導入することでクラウド基盤から利用するマシンイメージやデータへのアクセスの高速性を達成することについて国際会議発表発表を行った[22].

9.3 残された問題点

「9.1.3 議論の前提条件」で述べた制約に対応して提案アーキテクチャについて，以下の問題点が残っている.

9.3.1 実用的なサイズでの運用

第 1 章で目標として掲げた 20,000 コア規模のインタークラウド基盤を実運用する際に必要となる周辺技術やノウハウの蓄積がまだなされていない．これを克服するためには実際に上記規模の構築を目指した技術検証と実現後の運用技術の蓄積が必要である.

9.3.2 各アカデミッククラウド提供機関内のハードウェアリソース提供

各アカデミッククラウドとハブ的なインタークラウド基盤データセンタというハブ アンドスポーク型ではインタークラウド基盤データセンタのスケーラビリティに依存したアカデミックコミュニティクラウドのスケーラビリティとなる．アカデミックコミュニティクラウドのスケーラビリティをさらに高めるためには，Peer to Peer 型のトポロジにも対応する必要がある．すなわち各アカデミッククラウド提供機関内のハードウェアリソースを相互提供する仕組み作りが必要となる.

9.3.3 国際連携

国内にとどまらず国際的なアカデミッククラウド連携を実施する場合，SINET のネットワーク機能のみを前提にすることができない．この場合，L2 over L3 技術等の適用によりさらに適用範囲を広げる必要がある．この場合，本提案ではSINETが保証していた性能やQOSについて国際連携時にどう保証するのかという課題が残されている.

9.3.4 対応コミュニティ拡大

アカデミックコミュニティ以外のコミュニティ向けコミュニティクラウド構築に貢献するた

めには SINET や学認のみに依存しないアーキテクチャが必要となる．例えば，インターネット接続のみを前提としたアーキテクチャの開発が残された問題である．

9.4　本研究にかかわる研究/標準化技術の今後の展開

以下の順で今後本研究の成果を展開して行く予定である．

1．　実用的なサイズでのインタークラウド基盤の実装と議論を実施し，その運用性の確認を行う．

2．　オープンソース化したインタークラウド基盤ソフトウェアを普及することでクラウド連携の標準化を必要としないアカデミックコミュニティクラウド構築を推進する．

3．　インタークラウド基盤上に構築できるクラウド基盤の種類を順次充実させることで，さらに適用範囲の拡大をはかる．具体的にはビッグデータ基盤連携やスパコン連携などを対象とする．

9.5　今後の展望

今後は研究成果を実際の SINET と学認と連動するインタークラウド基盤の構成に活用することで，ネットワーク，認証基盤それにクラウド基盤が一体となった学術基盤を構成し，アカデミッククラウド利用者にとって使い易く，さらに投資対効果の高いサービス提供したい．(図 9-1)

技術的な主な問題は本研究を通じて解決したのだが実際にアカデミックコミュニティクラウドを実現するためには，関係者の合意，特に継続的財政支援に関する仕組み作りが最も大きなハードルになると推測される．このハードルを越えるためにも，基本的な仕組みに提供にとどまらず，さらに使い易い仕組み作りを通じて利用者コミュニティを構築して行く必要がある．

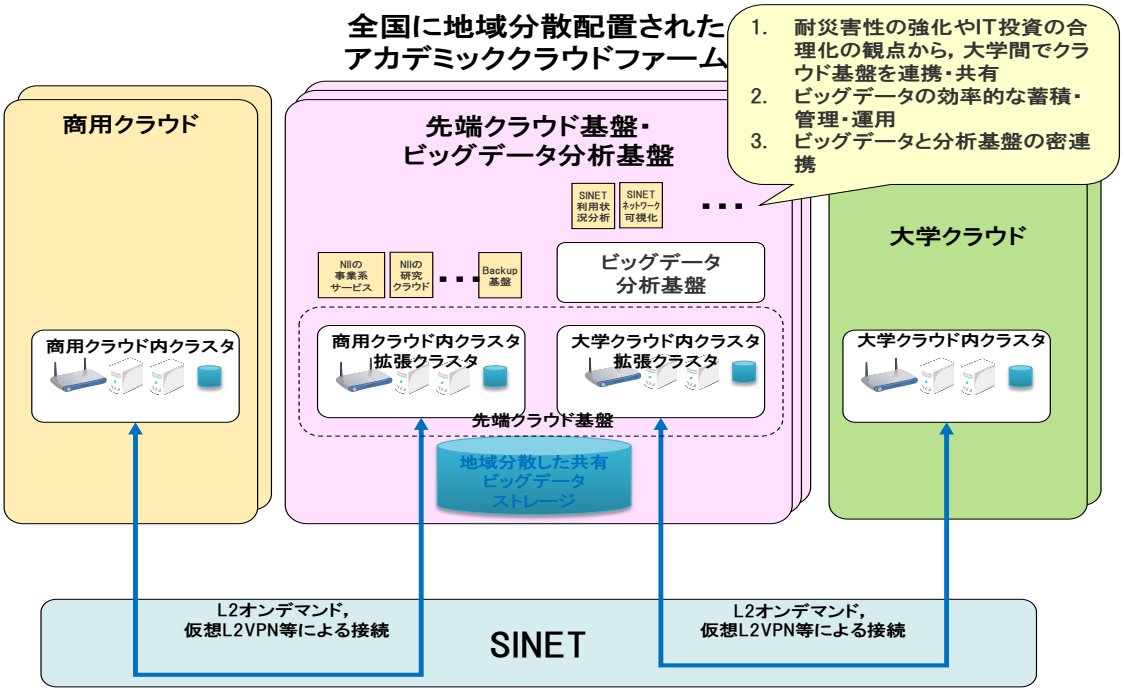


図 9-1 アカデミックコミュニティクラウドの実現に向けた取り組み例

さらには本研究の発展方向として動的クラウド基盤構築を一般化する提案アーキテクチャの進

化として，コンテクストアウェアアクセス制御に関する研究成果を活用した，Selfdeployable Software とその実行を支えるデータセンタ Kernel に関する研究がある．
前者は付録 A に示すセキュアコンテクストアウェアネス研究「動的なアクセス権変更をスケーラブルに実現するコンテクストアウェアアクセス制御方式 ACA^2 の提案」や[41],[42],[43]の研究成果を元に発展させる予定である．後者については本研究と上記研究を組み合わせることにより Data center kernel 側への設計方針を明確にすることで進行させる．

クラウドの基盤ソフトとして CaaS を今後 OS kernel に相当する Data center kernel としてモデル化することで，さらにアプリケーションから使い易いクラウド基盤として成長させて行きたい．（図 9-2）この際，既存の IaaS のようにサービスとして呼び出すと言う 1 方向の関係ではなく，インフラ側の状況を通知し，それに従った動きをアプリケーション側が動的に実施できる仕組みを基盤側に予め埋め込むアプローチを取る．例えば SINET の状況変化により利用するネットワークルートを動的に変化させるなど，コンテクストアウェアなアクセス制御の技術を適用したクラウド基盤構築を目標とする．

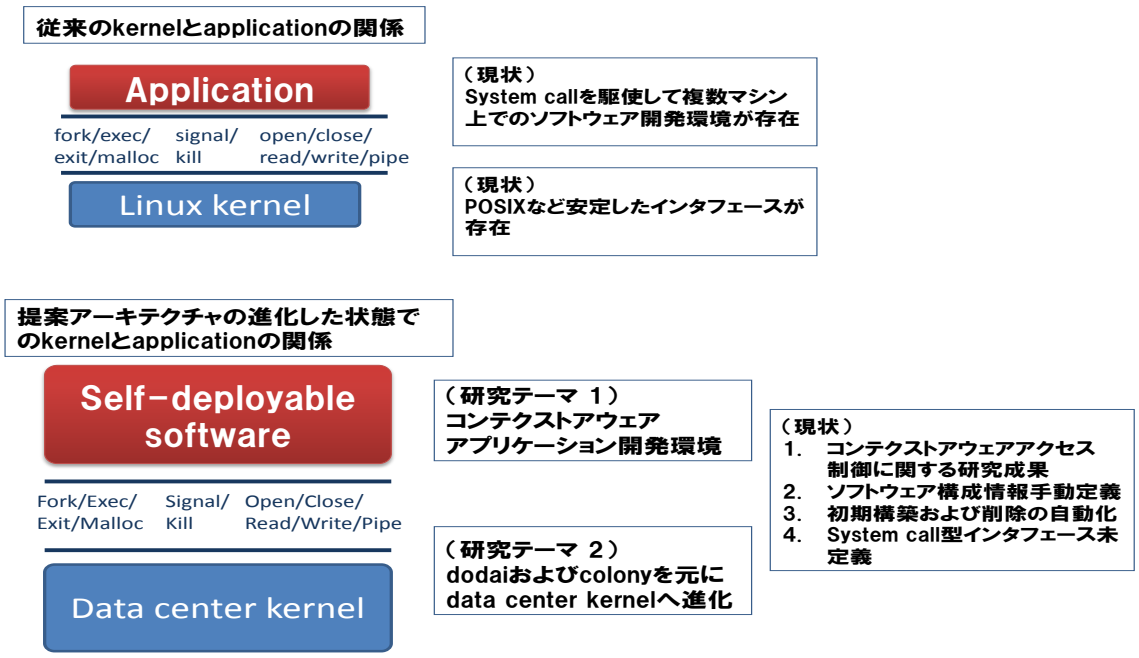


図 9-2 提案アーキテクチャの進化

謝辞

本研究は，多くの方々のご指導とご助力のもとに遂行されました．以下に特にお世話になった方々のお名前を記して感謝の意を表します．

本論文作成の指導教員である，総合研究大学院大学，国立情報学研究所の吉岡信和准教授に感謝致します．吉岡先生には，ご多忙中の中，研究の相談に乗っていただき，論文の訂正や研究の方向性など，きめ細やかな指導をして頂きました．こうして本論文を完成することができたのも，吉岡先生のご指導によるものです．ありがとうございました．

また，総合研究大学院大学在学中にセキュアコンテクストアウェアネスに関する研究について終始暖かい激励とご指導、ご鞭撻をいたいた山田茂樹教授に大変感謝しております．研究指導に留まらず，社会人および人間としてご指導頂き，多くのことを学ぶことが出来ました，

また，本博士論文の審査において貴重なご指導とご助言を頂いたました，総合研究大学院大学，国立情報学研究所 漆谷重雄教授，合田憲人教授，石川冬樹准教授に深く感謝いたします．本研究開始の動機となる，クラウド開発，運用に参加する機会をいただいた国立情報学研究所の本位田真一教授に心より感謝申しあげます．

さらに，教育クラウドと研究クラウドの運用に参加していただいた長久勝氏，谷沢智史氏，　西村一彦氏，それらの開発に参加いただいた桑田喜隆氏，中村竜也氏，山田大輔氏，落田征士氏，市村元信氏，志田隆弘氏，田中智文氏，羽深修氏に感謝いたします．

最後になりますが，いつも心の支えになってくれた大阪に住む母 弘子に心から感謝します．そして，四十路を過ぎてからの博士課程入学を快く承諾し，どのような状況においても応援してくれた，素晴らしい妻 和子にこの論文を捧げます．

本研究の成果が皆々様のご期待に沿うものとなるよう，今後さらに精進することを誓いますと同時に，ここに重ねて厚く謝意を表し，謝辞といたします．

参考文献

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, “Above the clouds: A Berkeley view of cloud computing,” University of California at Berkeley, Tech. Rep., February 2009. [Online]. Available: http://berkeleyclouds.blogspot.com/

[2] P. Mell and T. Grance, “The nist definition of cloud computing,” National Institute of Standards and Technology, vol. 53, no. 6, p. 50, 2009. [Online]. Available: http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc

[3] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging it platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (6) (2009) 599–616.

[4] Luis Rodero-Merino, Luis M. Vaquero, Victor Gil, Fermín Galán, Javier Fontán, Rubén S. Montero, Ignacio M. Llorente, From infrastructure delivery to service management in clouds, Future Generation Computer Systems 26 (8) (2010) 1226–1240.

[5] L. Vaquero, L. Rodero-Merino, J. Caceres, M. Lindner, A break in the clouds: towards a cloud definition, SIGCOMM Computer Communications Review 39 (1) (2008) 50–55.

[6] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm,“What’s inside the cloud? an architectural map of the cloud landscape,” Software Engineering Challenges of Cloud Computing, ICSE Workshop on, vol. 0, pp. 23–31, 2009.

[7] K. Keahey, M. Tsugawa, A. Matsunaga, J. Fortes, Sky computing, IEEE Internet Computing 13 (5) (2009) 43–51.

[8] Gabor Kecskemeti, Gabor Terstyanszky, Peter Kacsuk, Zsolt N’emeth, An approach for virtual appliance distribution for service deployment, Future Generation Computer Systems 27 (3) (2011) 280–289.

[9] K. Keahey, T. Freeman, and Contextualization: providing one-click virtual clusters, in: Fourth IEEE International Conference on eScience, IEEE, 2008, pp. 301–308.

[10] 吉岡信和,棟朝雅晴,本橋賢二,西村一彦,谷沢智史,横山重俊,アカデミッククラウド調査報告書 2012, (株)インプレス R&D, 2012.8

[11] SINET-4: http://www.sinet.ad.jp/

[12] 学認: https://www.gakunin.jp/

[13] GICTF: http://www.gictf.jp/

[14] Shigetoshi Yokoyama, Nobukazu Yoshioka: An Academic Community Cloud Architecture for Science Applications. SAINT 2012: 108-112

[15] B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, I. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, F. Galán, The reservoir model and architecture for open federated cloud computing, IBM Journal of Research and Development 53 (4) (2009) 1–11.

[16] Rochwerger, B., Galis, A., Levy, E., Caceres, J. A., Breitgand, D., Wolfsthal, Y., Llorente, I. M., Wusthoff, M., Montero, R. S., & Elmroth, E. (2009b). Management

technologies and requirements for next generation service oriented infrastructures. 11th IFIP/IEEE International Symposium on Integrated Management.
[17] S. Clayman, G. Toffetti, A. Galis, C. Chapman, “Monitoring Services in a Federated Cloud - The RESERVOIR Experience,” Achieving Federated and Self-Manageable Cloud Infrastructures, IGI Global, to appear 2012.
[18] B. Rochwerger, J. Tordsson, C. Ragusa, D. Breitgand, S. Clayman, A. Epstein, D. Hadas, E. Levy, I. Loy, A. Maraschini, P. Massonet, H. Munoz, K. Nagin, G. Toffetti, and M. Villari, “RESERVOIR - When one cloud is not enough,” IEEE Special Issue, Computing, 2011.
[19] Delta Cloud: http://deltacloud.apache.org/
[20] Scalr: https://scalr.net/
[21] Shigetoshi Yokoyama, Nobukazu Yoshioka, “Cluster as a Service for self-deployable cloud applications”, CCGrid 2012, pp.703-704, 2012.
[22] Shigetoshi Yokoyama, Nobukazu Yoshioka, Motonobu Ichimura, “Intercloud Object Storage Service: Colony,” Cloud Computing 2012.
[23] Shigetoshi Yokoyama, Nobukazu Yoshioka, "Dodai-Deploy: Fast Cluster Deployment Tool," icws, pp.681-682, 2012 IEEE 19th International Conference on Web Services, 2012
[24] Dodai:https://github.com/nii-cloud/dodai.
[25] Puppet:http://puppetlabs.com/
[26] MCollective:http://puppetlabs.com/mcollective/
[27] OpenFlow:http://www.openflow.org/.
[28] OpenStack:http://openstack.org/
[29] 横山重俊, 桑田 喜隆, 吉岡 信和, "IT 技術者育成のためのクラウドコンピューティング基盤", 教育システム情報学会 研究会, 2012.3
[30] Nobukazu Yoshioka, Shigetoshi Yokoyama, Yoshionori Tanabe, and Shinichi Honiden, “edubase Cloud: An Open-source Cloud Platform for Cloud Engineers,” SECLOUD '11 Proceedings of the 2nd International Workshop on Software Engineering for Cloud Computing, p.73, 2011.
[31] Shigetoshi Yokoyama, Nobukazu Yoshioka, Takahiro Shida, "Edubase Cloud: Cloud platform for cloud education", Software Engineering Education based on Real-World Experiences (EduRex),2012.6, pp.17-20
[32] Shigetoshi Yokoyama, Nobukazu Yoshioka, Takahiro Shida , “Cloud in a Cloud for Cloud Education”, Principles of Engineering Service Oriented Systems (PESOS), 2012 ICSE Workshop on Principles of Engineering Service-Oriented Systems pp. 63-64
[33] Colony:https://github.com/nii-cloud/colony
[34] Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A.D. Joseph, R. Katz, S. Shenker and I. Stoica, NSDI 2011, March 2011.
[35] Open Cirrus: A Global Cloud Computing Testbed by Arutyun I. Avetisyan, Roy Campbell, Michael T. Heath, Steven Y. Ko, Gregory R. Ganger, Michael A. Kozuch,

David O'Hallaron, Marcel Kunze, Thomas T. Kwan, Kevin Lai, Martha Lyons, Dejan S. Milojicic, Hing Yan Lee, Yeng Chai Soh, Ng Kwang Ming, Jing-Yuan Luke, and Han Namgoong, Computer, April 2010, pp. 35–43
[36] VCL: http://vcl.ncsu.edu/
[37] Crowbar:https://github.com/dellcloudedge/crowbar
[38] MAAS: https://launchpad.net/maas
[39] JuJu: https://launchpad.net/juju
[40] 横山重俊, 桑田 喜隆, 吉岡信和, ”アカデミッククラウドアーキテクチャの提案と評価”, to appear in 情報処理学会誌 情報処理ネットワークサービスと分散処理特集（10 pages), 2012 年 2 月
[41] 横山 重俊, 上岡 英史, 山田 茂樹, ”コンテクストアウェアアクセス制御方式のユビキタスサービスへの適用事例”, 電子情報通信学会モバイルマルチメディア通信研究会（MoMuC）技報 106(498, M) 19-24 2007 年 1 月
[42] Shigetoshi Yokoyama, Eiji Kamioka, Shigeki Yamada: An Anonymous Context Aware Access Control Architecture for Ubiquitous Services. MDM 2006: 74
[43] 横山重俊, 上岡英史, 山田茂樹, ” ユビキタスサービスに適したコンテクストアウェアアクセス制御方式の提案”, 電子情報通信学会技術研究報 105(565,M) 7-12 2006 年 1 月
[44] I. Foster, C. Kesselman, and S. Tuecke, “The anatomy of the grid: Enabling scalable virtual organizations,” International Journal of High Performance Computing Applications, vol. 15, no. 3, pp. 200–222, 2001.
[45] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar, S. Krishnakumar, D. Pazel, J. Pershing, and B. Rochwerger, “Oc´eano-SLA based management of a computing utility,” Proceedings of 6th IEEE/IFIP International Symposium on Integrated Network Management, pp. 855–868, 2001.
[46] M. Turner, D. Budgen, and P. Brereton, “Turning software into a service,” Computer, vol. 36, no. 10, pp. 38–44, October 2003.
[47] G. J. Popek and R. P. Goldberg, “Formal requirements for virtualizable third generation architectures,” Communications of ACM, vol. 17, no. 7, pp. 412–421, 1974.
[48] I. Foster and C. Kesselman, “Globus: A metacomputing infrastructure toolkit,” International Journal of Software Architecture, vol. 11, no. 2, pp. 115–128, 1997.
[49] P. Gardfj¨all, E. Elmroth, L. Johnsson, O. Mulmo, and T. Sandholm, “Scalable grid-wide capacity allocation with the SweGrid Accounting System (SGAS),” Concurrency and Computation: Practice and Experience, vol. 20, no. 18, pp. 2089–2122, December 2008.
[50] J. Skene, A. Skene, J. Crampton, and W. Emmerich, “The monitorability of service-level agreements for application-service provision,” in WOSP ’07: Proceedings of the 6th international workshop on Software and performance. ACM, pp. 3–14., 2007
[51] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” in Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation - Volume 6. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–

10.
[52] D. Bermbach, M. Klems, S. Tai, and M. Menzel, “Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs,” Proceedings of the IEEE Cloud 2011 - to appear, 2011.
[53] H. Weatherspoon and J. Kubiatowicz, “Erasure coding vs. replication: A quantitative comparison,” Revised Papers from the First International Workshop on Peer-to-Peer Systems, ser. IPTPS ’01. London, UK: Springer-Verlag, 2002, pp. 328–338. [Online]. Available: http://portal.acm.org/citation.cfm?id=646334.687814
[54] O. Goldshmidt, B. Rochwerger, A. Glikson, I. Shapira, and T. Domany, “Encompass: Managing functionality,” in Proceedings of 21th International Parallel and Distributed Processing Symposium (IPDPS 2007), March 2007, pp. 1–5.
[55] C. Chapman, W. Emmerich, F.G. Marquez, S. Clayman, A. Galis, “Software Architecture Definition for On-demand Cloud Provisioning,” Special Issue of Cluster Computing, 2011.
[56] Y. Song, Y. Sun, H. Wang, X. Song, An adaptive resource flowing scheme amongst VMs in a VM-based utility computing, 7th IEEE International Conference on Computer and Information Technology, CIT 2007, IEEE, 2007, pp. 1053–1058.
[57] A. McCloskey, B. Simmons, H. Lutfiyya, Policy-based dynamic provisioning in data centers based on slas, business rules and business objectives, IEEE/IFIP Network Operations and Management Symposium, NOMS’08, IEEE, 2008, pp. 903–906.
[58] Tiago C. Ferreto, Marco A.S. Netto, Rodrigo N. Calheiros, Csar A.F. De Rose, Server consolidation with migration control for virtualized data centers, Future Generation Computer Systems (May) (2011).
[59] J. Tordsson, R.S. Montero, R.M. Vozmediano, I.M. Llorente, Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers, 2010, in press (doi:10.1016/j.future.2011.07.003).
[60] A. Ali-Eldin, J. Tordsson, E. Elmroth, An adaptive hybrid elasticity controller for cloud infrastructures, 2011 (submitted for publication).
[61] D. Breitgand and A. Epstein, “SLA-aware Placement of Multi-Virtual Machine Elastic Services in Compute Clouds.” IBM Technical Report, H-0287, 2010.
[62] G. Toffetti, A. Gambi, M. Pezz`e, and C. Pautasso, “Engineering autonomic controllers for virtualized web applications,” Web Engineering (B. Benatallah, F. Casati, G. Kappel, and G. Rossi,eds.), vol. 6189 of Lecture Notes in Computer Science, pp. 66–80,
[63] Springer Berlin / Heidelberg, 2010.
[64] Google App Engine, http://appengine.google.com . (November 23, 2012)
[65] Windows Azure Platform, http://www.microsoft.com/azure/ .(November 23, 2012)
[66] Amazon Elastic Compute Cloud (Amazon EC2): http://aws.amazon.com/ec2/ (November 23, 2012)
[67] OpenQRM, “the next generation, open-source Data-center management platform”, http://www.openqrm.com/. (November 23, 2012)
[68] DMTF. Open Virtualization Format (OVF). Visited May 2011,

http://www.dmtf.org/standards/ovf.
[69] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, The Eucalyptus Open-source Cloud-computing System, Proceedings of Cloud Computing and Its Applications, Chicago, Illinois (October 2008)
[70] Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., Zagorodnov, D.: "The Eucalyptus Open-Source Cloud-Computing System." ,Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2009), Shanghai, China, May 18-May 21 (2010)
[71] Ranjan, R.: Coordinated Resource Provisioning in Federated Grids. Ph.D. Thesis, The University of Melbourne, Australia (March 2009)
[72] Ranjan, R., Liu, A.: Autonomic Cloud Services Aggregation. CRC Smart Services Report (July 15, 2009)
[73] Buyya, R., Ranjan, R., Calheiros, R.N.: Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities. , Proceedings of the 7th High Performance Computing and Simulation Conference (HPCS 2009), Leipzig, Germany, June 21-24. IEEE Press, New York (2009)
[74] Quiroz, A., Kim, H., Parashar, M., Gnanasambandam, N., Sharma, N.: "Towards Autonomic Workload Provisioning for Enterprise Grids and Clouds." , Proceedings of the 10th IEEE International Conference on Grid Computing (Grid 2009), Banff, Alberta, Canada, October 13-15 (2009)
[75] W. Li and L. Ping, "Trust model to enhance security and interoperability of cloud environment," Cloud Computing, pp. 69–79, November 2009.
[76] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Resource Leasing and the Art of Suspending Virtual Machines," High Performance Computing and Communications, 2009. HPCC '09. 11th IEEE International Conference on, pp. 59–68, June 2009.
[77] C. Hoffa, G. Mehta, T. Freeman, E. Deelman, K. Keahey, B. Berriman, and J. Good, "On the Use of Cloud Computing for Scientific Workflows," SWBES 2008, Indianapolis, December 2008.
[78] B. Sotomayor, R. Montero, I. Llorente, and I. Foster, "Virtual infrastructure management in private and hybrid clouds," Internet Computing, IEEE, vol. 13, pp. 14–22, September 2009.
[79] The Shibboleth system standards, http://www.openqrm.com.
[80] OpenID Authentication 2.0, OpenID Foundation, http://openid.net/specs/openid-attribute-exchange-2 0.html, 2007.
[81] 高宮安仁, 真鍋篤, 松岡聡. 「Lucie: 大規模クラスタに適した高速セットアップ・管理ツール」先進的計算基盤システムシンポジウム SACSIS2003 論文集, 情報処理学会, 電子情報通信学会, pp.365-372, 2003 年 5 月
[82] Takamiya, Y., Manabe, A. and Matsuoka, S., "Lucie: A Fast Installation and Administration Tool for Large-scaled Clusters", Proceedings of Symposium on Advanced Computing Systems and Infrastructures (SACSIS2003), pp.365–372

(2003).
[83] Baltic-avenue : http://code.google.com/p/baltic-avenue/
[84] Boardwalk :https://github.com/razerbeans/boardwalk/
[85] Fs3: http://fs3.sourceforge.net/
[86] Sinatra-s3: https://github.com/nricciar/sinatra-s3/
[87] Radosgw: http://ceph.newdream.net/wiki/RADOS_Gateway/
[88] Ceph: http://ceph.newdream.net/
[89] Storage Resource Broker: http://www.sdsc.edu/srb/
[90] Grid Datafarm: http://datafarm.apgrid.org/