Vol.2015-IOT-30 No.12
2015/7/4

Overlay Cloud で構成する Web サービスバックアップサイト構築例

横山 重俊[†1]　政谷 好伸[†1]　吉岡 信和[†1]　合田 憲人[†1]

Web サービスのバックアップサイトを Overlay Cloud アーキテクチャで構成する事例について紹介する．具体的には，アーカイブ系コンテンツサービスを構成する Web サービス全体のサービスバックアップを実現する．提案方式の特徴は，対象とする Web サービスのコンテナ化とそれらを運用するシステム基盤自体のコンテナ化という二重のコンテナ技術の適用により，クラウドを跨るシステム構成を取ることで高い可用性確保と運用省力化の両立を目指す点である．本稿では，ユースケース分析に沿った方式設計と実装により評価環境を構築，評価した結果を報告する．本評価の結果，対象とするクラウドを特に選ばないバックアップサイト構築を実現できる見通しを得た．

A Case Study on Web Service Back-up site deployments on Overlay Cloud Architecture

Shigetoshi Yokoyama[†1], Yoshinobu Masatani[†1], Nobukazu Yoshioka[†1], Kento Aida[†1]

The Inter-Cloud is a promising approach for recent various demands for cloud computing. However, building the Inter-Cloud environment requires expert knowledge and high skills for network/server administration and not feasible for every application developer. This paper presents Virtual Cloud Provider (VCP), the middleware to automatically build a set of virtual resources on the Inter-Cloud in deploying web service back-up site for digital archive services.

1. はじめに

国立情報学研究所（以下，NII）ではアカデミックインタークラウド構想[1]に基づきアカデミックコミュニティ向けクラウド基盤の構築を検討している．アカデミックインタークラウド構想で想定している主なユースケースは，サービスバックアップ，共同研究環境提供，分散データ処理の３つである．これらのユースケースを実現するためには，一般的には，アカデミックコミュニティクラウド内のリソースだけではなく，プライベートクラウド側のリソース，そしてパブリッククラウド側リソースに跨ったクラウド連携が必要になる．

クラウド連携の方法としてアカデミックインタークラウド構想では，Overlay Cloud アーキテクチャ[2]で構成する広域に分散した Virtual Cloud を提供する Virtual Cloud Provider の実現を前提としている．本稿では，上記ユースケースのうちサービスバックアップの一例としてアーカイブ系コンテンツサービスを構成する多数の Web サービスをバックアップするサイト構築例について紹介し，この Virtual Cloud Provider の当該ユースケースへの適用方法と適用性評価について述べる．以下，第二章で Virtual Cloud Provider に対する要求分析について述べ，第三章で実現方式，第四章では構築した評価環境について説明する．また，第五章では，評価結果について報告し，第六章でまとめる．

2. 要件分析

2.1 Virtual Cloud Provider の定義

Virtual Cloud Provider（以下，VCP）は，特定のクラウドサービスやデータセンター基盤に依存せず，複数の異なるクラウドに存在するリソースを統合し，共通のホスト環境イメージをコンテナ形式で提供する仕組みである[3]．これらのコンテナ型ホスト（以下，ベースコンテナ）の集合体は，要求に合った仮想プライベートネットワークが利用可能な状態で提供され，対象アプリケーションの要件に応じた様々な形式の「Virtual Cloud」を構成する．Virtual Cloud Provider が複数のクラウドサービスやデータセンター基盤に跨ったインフラ資源を一元的に管理する機構を持つことにより，アプリケーションの運用・管理者は基盤毎の利用方法の違いやリソースの存在場所などを直接的に意識することなく，Virtual Cloud の構築を行うことができその上でコンテナ化されたアプリケーション（以下，アプリケーションコンテナ）を実行することができる．

2.2 想定するユースケース

アカデミックインタークラウド構想で想定する代表的なユースケースを表１にまとめる．これらのうち，共同研究基盤と分散データ処理についての適用性については別稿[2]で述べたので，本稿ではサービスバックアップについての VCP の適用性について説明する．

†1　国立情報学研究所
National Institute of Informatics

表１．ユースケース一覧

Table 1. Use Cases

ユースケース	概要	求められる機能
サービスバックアップ： コンテンツサービスの定期停電対応	サービス運用環境の短期間の計画的な停止に対応し，バックアップサイトを任意のクラウド上に開設する．	異種クラウドを跨ぐシステム拡張・縮退
	退避系についてもwritable動作での運用を可能とする．	アクセス経路の切り替え
		DBMSクラスタリング
		分散ファイルシステム
サービスバックアップ： コンテンツサービスディザスタリカバリ (DR)	災害などにより突発的に現行システムが被害を受けた場合でも，別環境でサービスを継続する． 任意のタイミングで切り替え可能なバックアップサイトを並行運用する．	広域分散環境上で以下を実現する．
		DBMSクラスタリング
		分散ファイルシステム
		障害監視
		自動フェイルオーバー
共同研究基盤	大学・研究機関に分散した計算資源を有機的に結合し，シームレスな共同研究基盤を提供する．	大学・研究機関などの計算資源の（一時的）統合管理
		モバイル連携(L2)
		動的構成変更（自律協調configuration）
		（大規模分散ストレージ・リポジトリ）
分散データ処理	大量に発生するデータを適切に処理するために，分析・監視機能をデータ発生源に広域分散運用する．	ネットワークトラフィックのビッグデータ分析
		ストリームデータ処理
	ネットワークに広域分散させた環境上でLocation Awareなクラスタを構成する．	多数のコンテナを統合的に扱う仕組み
		リソース配置の最適化
		ネットワーク構成（経路・機能）の最適化

具体的には，下表のようなサービスバックアップに関する運用形態を VCP 上で構築できることを示す．

表２　運用形態一覧

Table 2. Service Management Contexts

単一サイト内HA	同一のDC，クラウドサービス，ネットワークセグメント内のネットワークの信頼性・品質が高い環境においてHA機能を提供する．
インタークラウドHA	DCやクラウドを跨ぐ環境においてHA機能を提供する．ネットワークの信頼性や品質は必ずしも高くなく，遅延やパケットロスが発生する可能性があること，マシンスペックが環境毎に異なることを想定する必要がある．
計画保守・法定停電対応	スケジュールに合わせて運用環境を他のDCやクラウドに移動させる．移動先へのデータコピーの所要時間がスケジュールに間に合うように計画する必要がある．
DR対応	突発的な災害や大規模障害時に運用環境を他のDCやクラウドに移動させる．元の運用環境のデータが消失することに備えて常に移動先に最新のデータを保持しておく必要がある．

表３　要求レベル一覧

Table 3. Service Management Levels

	フェイルオーバー	RTO（復旧時間目標）		RPO（復旧時点目標）
	トリガー方式	障害検出時間	復旧時間	
単一サイト内HA	障害監視・検出	数分	数秒から 5分	ゼロ データ損失無し
インタークラウドHA	障害監視・検出	数分	数秒から 5分	ゼロ データ損失無し
計画保守・ 法定停電対応	スケジュール	自動検出不要	停止時間ゼロ	ゼロ データ損失無し
DR対応	手動 スイッチオーバー	自動検出不要	数時間以内	一日分まで許容

2.3 運用要件

運用者へのヒアリングによると，アーカイブ系コンテンツサービス運用において以下の課題が顕在化している．

- 各 Web サービスについて，OS やアプリケーションのアップデート作業を個別に行う必要があり，サーバ管理の作業負荷が高い．
- Web サービス毎の負荷に差がある状況下において，1 サービスあたり固定的なリソース割り当てのため，効率的なリソース利用ができていない．
- ソフト / ハードシステムの維持管理・保守費用が高額である．
- 計画保守，法定停電などに伴うシステム停止への対応作業が定期的に発生し運用稼働を圧迫する．

これらの課題を解決するため，アーカイブ系コンテンツサービスを VCP 上で実現する際には，以下のような運用性に関する要件がある．

(1)　アプリケーションバージョンアップの省力化

複数サーバへのバージョンアップ・デプロイ作業をサーバ自体に変更を加えることなく，(コンテナ) イメージのソフトウェアによる置き換えなどにより省力化する．

(2)　OS パッチ適用の省力化

複数サーバへの OS やソフトウェアパッケージの更新適用作業をサーバ自体に変更を加えることなく，(コンテナ) イメージのソフトウェアによる置き換えによる省力化する．

(3)　計画保守・停止作業の省力化

事前に計画された法定停電や機器のリプレースなどによるサービス停止に対応し，代替システムへの切り替え・切り戻し作業をソフトウェアにより円滑に行う．

(4)　性能保証の自動化

負荷が上昇した場合，それに追従できるようにスケールアウト，スケールアップの必要性をソフトウェアにより自動的に判断し，実行する．

(5)　フェイルオーバーの自動化

システム監視によって現用系と待機系で二重化された環境における障害発生時の切り替えをソフトウェアにより自動的に行う．

(6)　資源利用の最適化

サーバリソースの負荷状況を監視することにより，特定のサーバだけが過負荷にならないようにアプリケーションの配置状態の自動制御, ロードバランシングを行う. また，リソースに余裕がある状況において CPU/メモリのオーバーコミットをサポートし，インフラコストの削減に寄与する．

(7)　データバックアップ

データのバックアップ処理を設定したスケジュールに従って自動的に行う．

3. VCP 実現方式

VCP で構築する運用形態に設定された要求レベルを満たすための実現方式について，以下の３つの側面から検討した．

- データ保全
- システム監視
- 性能保証

3.1 データ保全

データの消失に備えるために，データの種類や特性に合わせた保全方法を選択する必要がある．

3.1.1 保全対象データの種類

HA クラスタにおいて保全対象とすべきデータの種類と主な特徴は以下のとおりである．

(1) システムファイル

OS/ミドルウェア/アプリケーションの実行可能ファイル（システム設定情報やバージョン状態管理を含む）

(2) 固定的なデータ

稼働中に変化しないデータ（サイト設定やデザインなど）

(3) 追加・更新が発生するデータ

レギュラーファイル（利用者がアップロードしたコンテンツなど）と DB データファイル

(4) Web サービス毎に独立したデータ
アクセスログ等

3.1.2 データの保全方法

(1) マスター/スレーブ（プライマリ/バックアップ）型保全
冗長なリソース間で一貫性を保ちながらデータをスレーブ側のストレージ領域へ複製する．

(2) マルチマスター（複数プライマリ）型保全
マルチサイト/クラウドを跨ってデータを冗長分散保持する．複数ホストからネットワーク経由でデータを共有し，論理的に構造化された 1 つの名前空間で透過的なアクセスを可能とする．

(3) 定期バックアップ保全
広域分散オブジェクトストレージなどの長期保存ストレージを活用した定期データバックアップを実施する．

3.1.3 検討結果

検討の結果，データの種類毎の保全方法は各運用形態に対応して以下のようにまとめた．

表 4 保全方法一覧
Table 4. Maintenance Procedure

	単一サイト内HA	インタークラウドHA	計画保守・法定停電対応	DR対応
システムファイル	internal registry	public registry or internal registries(※1)	image copy	public registry
固定データ	internal registry or shared disk	public registry or 分散object store	image copy or 分散object store	public registry or 分散object store
可変データ	shared disk or [A-S] replication (同期型) [A-A]分散FS	[A-S] replication (同期型) [A-A] 分散FS	定期backup or replication	定期backup or replication
アクセスログ	[A-S/A-A] サイト内収集サーバ or [A-S] replication	外部の収集サーバ	サイト内収集サーバ （切り戻し時に回収が必要）	サイト内収集サーバ （切り戻し時に回収が必要）

A-S: Active-Standby クラスタ/A-A: Active-Active クラスタ

3.2 システム監視

システム監視は運用性要件を満たすために以下のような目的で行われる．

① システムの故障や状態をリアルタイムに検知し，システム管理者や運用者に通知する．また，フェイルオーバーの要否を自動的に判断する．

②リソース情報などの実績情報を取得，蓄積し，分析できるようにする．

(1) 監視項目と対象

- 死活監視
  サーバ（VM/コンテナ），ネットワーク機器
- プロセス監視
  サーバ（VM/コンテナ）
- リソース（性能）監視
  サーバ（VM/コンテナ）
- ネットワーク監視
  LAN/WAN (DC 間/クラウド間)

(2) 監視方式とツール

- 死活監視
  クラスタ管理と連携した方式による死活監視を行う．
  Serf（簡易的なノード死活監視）
  Consul（サービス定義にヘルスチェックを指定可）
  corosync/Heartbeat + Pacemaker
  Keepalived
- エージェント監視
  各監視対象にインストールされたエージェントが監視を行う．システムの様々な稼働情報（主にリソース監視）を監視項目として設定でき，ネットワークがダウンしていても監視対象サーバ内で監視が継続される．
  Munin，MRTG，Sensu，Zabbix など
  （クラウドサービス型）New Relic，Datadog，Mackerel
- リソース情報蓄積
  各監視対象にインストールされたエージェントがリソース情報の時系列変化を記録する．
  コンテナのホスト側からの監視
  Docker の場合は，コンテナ内からは自コンテナのリソース使用状況（CPU 使用率，メモリ使用量，LoadAverage など）を通常のコマンド等では取得できないため，コンテナ毎のリソース状況はホスト側からモニタリングする．

3.3 性能保証

過負荷状態によるサービスレベル低下を防止するための対策を行う．リソース監視によって過負荷状態を検知し，自動的にそれに追従できるようにリソースを制御することでの影響範囲を極小化する．過負荷対策として以下のような方式がある．

(1) 流入制御
コネクション数を制限し，上限以上なら「Sorry サーバ」へアクセスを振り向ける．

(2) リソース分割
Web/AP 系，DB 系のような機能によるサーバ分割（論理/物理）を行う．

(3) 同時実行数制御
同時実行可能な処理やトランザクションの上限数を設定しておく．

(4) リソース再構成
過負荷時にスケールアウトなどによって構成を変更し，システム全体のリソースを最適化する．

- スケールアウト
  Web/AP サーバは「ステートレス」であればノード数を増やすことは容易である．セッション情報をステートとして持つ場合はセッション同期の実施が必要である．DB サーバの場合は，分散型 DB 機能を使う必要がある．
- スケールアップ
  クラウドサービスが提供するインスタンスタイプの変更，または Docker コンテナの「CPU Shares」パラメータ

による制御を行う. コンテナに対応する cgroup 内の CPU 数，メモリ割り当て量は動的変更可能だが，Docker の場合は, 現状 docker run コマンドオプションによる設定(コンテナ起動時の指定）のみがサポートされている.

(5) ネットワーク帯域制御

特定の通信に対する性能を確保するために，ネットワークの優先制御/帯域制御を行う.

4. Web サービスバックアップサイト構築例

前章までの検討結果に基づいて，VCP 上へのアーカイブ系コンテンツサービスの構築・運用を想定した評価環境を構築した.

4.1 構築時の要件

① クラウドや DC 基盤環境による資源の違いを意識することなく，どのホストでも容易に Web サービスを開設できること.

② Web サービスの可用性を向上できること.

Level-1: 自動的な再立ち上げによる障害復旧

Level-2: 待機系への切り替えによる障害復旧

Level-3: Active-Active (Multi-Master) 構成による冗長化

③ 確保した資源全体の負荷バランスを平準化し，効率的な資源利用ができること.

Level-1: 稼働中の IaaS ホスト間での Web サービスの移動・再配置

Level-2: 稼働する IaaS ホストの増減による Web サービスの移動・再配置

Level-3: Web サービスのスケールアウトによる負荷分散

④ バージョンアップのための作業負荷を軽減できること.

OS 環境 (共有ライブラリ, Apache, PHP, MySQL 等)

CMS アプリケーションを対象とすること.

4.2 評価環境実装

今回構築したアーカイブ系コンテンツサービスが稼働する基盤環境は，図 1 に示す要素で構成される.

① Web/App Node

Web サービスコンテナを配備するためのベースコンテナとして機能する. DBMS 接続を中継する Proxy モジュール，ネットワークファイルシステムをマウントする Client モジュールが組み込まれ，アプリケーションと Storage/DBMS の間でロードバランシングとフェイルオーバーを行う.

② Storage Service

分散共有ファイルシステム・サーバ機能を持つベースコンテナ群によって汎用的なネットワークストレージ機能を提供する．Web サービス毎にファイルシステムボリュームが分割され，アプリケーションからはネットワーク経由でマウントして利用する．データ保全と性能保証に関する要求レベルに対応して，Storage Service の実装方式を変更した場合でも，アプリケーション側では特に意識することなく透過的なアクセスを可能とする.

③ DBMS Service

DBMS Replication または Cluster 機能を持つベースコンテナ群によって汎用的な DBMS 機能を提供する. アプリケーション要件に対応した DB サーバコンテナ・イメージを持ち込んで利用することを許容し，DB Node 内においてそれらをアプリケーション単位で複数起動することができる．Storage Service と同様，DBMS の実装方式を変更した場合でもアプリケーション側からのアクセス手段には影響を与えない.

④ Utility Node

外部からの HTTP アクセスに対してリバースプロキシ/ロードバランサ機能を提供する．その他，Web/App Node に配備されたアプリケーションコンテナの稼働状態や設定情報を集約し，フェイルオーバーやスケールアウトの要否を判断する.

⑤ 外部ネットワーク接続用ルータ

Public Network とアーカイブ系コンテンツサービスの Private Network(VLAN)との間を接続する NAT ルータとして機能する.

4.3 動作シーケンス

4.3.1 バックアップサイト構築

各 Web サービスのデータは，定期的に広域分散ストレージへのバックアップが行われる．このデータを利用し，現用システムと異なるクラウド基盤へのサービスの引き継ぎを実現する．バックアップデータが用意できれば，クラウド基盤の種類に依らず新規にサービス運用環境を構築できるため，同じ手段によって DR を実現することも可能である．ただし，RPO（復旧時点目標）はバックアップの実行頻度に依存する.

【処理フロー】

① アーカイブ系コンテンツサービス管理者は，現用システムの機能を新規クラウド環境へ引き継ぐことを決める.

② アーカイブ系コンテンツサービス管理者は，必要があれば，クラウド基盤上に VCP をインストールする.（VCP 自体もコンテナイメージとして配布される.）

③ アーカイブ系コンテンツサービス管理者は，VCP に対してアーカイブ系コンテンツサービスの構築を指示する.

以下，VCP によって次の一連の処理が実行される.

④ システムを構成する各要素に対応する IaaS ホスト・リソースを確保する.

- クラウド基盤のリソース確保は VCP がクラウド基盤に対応した API を呼び出すことで実現する.
- 対象アプリケーションの「リソース要求記述」に確保すべき IaaS ホストの種類や数量，ネットワーク構成が記述されている.
- リソース要求記述は移行前のクラウド基盤の利用状況から類推可能なものであるため，アーカイブ系コンテンツサービス管理者が記述して VCP に設定しておく.

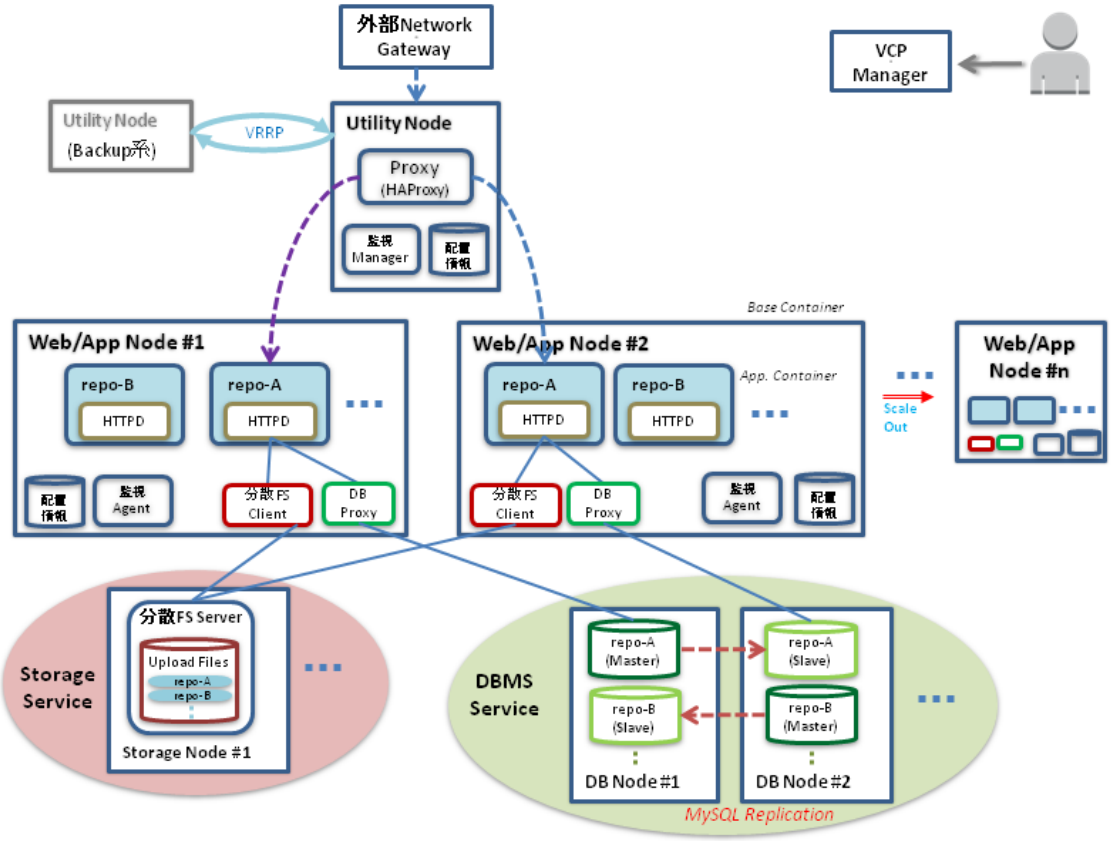

図1 アーカイブ系コンテンツサービス構成例
Figure 1. Service Configulation

⑤ 各 IaaS ホスト上で VCP が提供する基盤機能を担うベースコンテナを起動し，構成する．
- 分散共有データストレージサービス（複数 Node のクラスタ構成）
- DBMS サービス（複数 Node のクラスタ構成）
- Web/App Node
- Utility Node（リバースプロキシ/ロードバランサ機能）
- ベースコンテナ間を仮想ネットワークにより接続
- 配置・設定情報共有用分散 KVS クラスタを構成

⑥ 全 Web サービスについて，以下の処理を行う．
- データボリューム（ストレージ，DBMS）初期化
- バックアップデータリストア
- Web サービスコンテナ起動
- リバースプロキシ設定

4.3.2 障害検知・自動復旧

各 Web サービスの稼働状態を定期的に監視し，故障/停止等の障害を検知した際にサービスの自動回復を行う．

【処理フロー】

① Utility Node から現在稼働中の Web サービスに対して HTTP によるヘルスチェックを一定時間間隔で実行する．

② Utility Node のヘルスチェックプログラムは，特定の Web サービスの異常を検出し，分散 KVS 内のコンテナ稼働状態を「稼働中」から「停止」に更新する．

③ Web/App Node において当該 Web サービスの稼働すべきコンテナの個数が不足していることを検出し，自ノードで当該サイトのコンテナが起動していなければ新たにコンテナを起動する．

④ コンテナ起動時に，当該 Web サービスのデータストレージ領域のマウント，および DBMS への接続パラメータ更新を行う．

⑤ 新たに Web サービスを配備した Web/App Node において，分散 KVS 内のコンテナ稼働状態を「停止」から「稼働中」に更新する．

⑥ Utility Node のリバースプロキシ設定更新プログラムは，復旧した Web アービスのコンテナ稼働情報(稼働先 Web/App Node の IP アドレス, Port 番号) の変化を検出し，設定に反映する．

⑦ 当該 Web サービスに対する外部からのアクセス経路が回復し，サービスが再開される．

4.3.3 負荷分散

アクセス数やデータ量の増大などに対応し，特定の Web サービスをスケールアウトすることでアクセス負荷を分散

する．
【処理フロー】
① VCP の性能監視により，特定の Web/App Node で稼働している Web サービスの応答性能の低下が閾値に達したことを検知する．監視結果を元に，予め設定された閾値との比較により判断する．
② VCP が各 IaaS ホストの空きリソース状況に基づいて，高負荷の Web サービスを特定し，そのスケールアウト先となるホスト決定する．
③ VCP によって Web サービス用コンテナのスケールアウト処理が行われる．

4.3.4 システムアップデート

リポジトリシステムの OS アップデートを共通のコンテナイメージの置き換えによって容易に実現する．最終的には全 Web サービスがアップデート対象となるが，様子を見ながら順次対象を増やし，万が一アップデートによって問題が発生した場合には更新前の状態に戻すことができる．
【処理フロー】
① アーカイブ系コンテンツサービス管理者は，Web サービス用コンテナイメージの更新作業をメンテナンスサーバ上で行う．
(ア) 現バージョンの Web サービス用コンテナイメージを Registry から取り出す．
(イ) Web サービス用コンテナを起動し，OS ライブラリ，Web サーバ，DB サーバ等のソフトウェアアップデートを実行する．
(ウ) 更新したコンテナイメージを Commit し，Registry へ push する．
② アーカイブ系コンテンツサービス管理者は，各 Web/App Node で稼働中の Web サービス用コンテナのアップデートを VCP へ指示する．
③ VCP は，各 Web/App Node 上で Web サービス用コンテナイメージの最新版を Registry から取得する．
④ VCP は，全ての Web サービス用コンテナを 1 コンテナずつ再起動する．
⑤ アップデートにより問題が発生した場合，アーカイブ系コンテンツサービス管理者は Web サービス用コンテナのロールバックを VCP へ指示する．

5. 評価

前章で述べた評価環境を利用し，アーカイブ系コンテンツサービスを構成する Web サービス全体のサービスバックアップというユースケースへの運用面での VCP 適用性検証を行った．

5.1 ホスト環境構築方式

(a) サーバリソース割り当て

アーカイブ系コンテンツサービスの構築先となるクラウドや DC 基盤から仮想サーバ，またはベアメタルサーバを確保する．リソース確保の手段はクラウド環境によって異なる API 等が提供されるため，実際に利用する基盤に合った方式を選択することになる．本構築では，NII のベアメタルクラウド環境の事前確保済みのベアメタルマシン，および AWS EC2 インスタンス[4]の 2 種類のクラウド環境でそれぞれ動作検証を行った．AWS については，Vagrant Provider プラグイン[5]を用いてリソース確保とサーバ起動，および必要なソフトウェアのインストールまでの一連の構築作業を自動化できることを確認した．本システムは複数台のノードから構成され，それらの各ノードの要件（インスタンスタイプ，ディスク容量，配置先リージョン等）に基づいて設定ファイル(Vagrantfile)を記述し vagrant コマンドを実行した．
【課題】
インタークラウドでの構築を行うために，リソース要求記述に対する統一的なインターフェースが求められる．Vagrant は AWS 以外にも複数種類のクラウド基盤をサポートするプラグインが存在するが，本来，開発・テスト環境用の仮想マシンを管理する目的で使われることが前提となっている．このため，異なる Provider を混在させて一括適用することは不可能であり，構築先のクラウド基盤毎に個別に実行する必要がある．また，インタークラウドでの利用は基本的に想定されていない．

(b) 仮想ネットワーク作成

クラウド環境から確保したサーバ間で対象アプリケーション要件に対応したオーバーレイ方式の仮想ネットワークを作成する．本システムでは，NII クラウド内のネットワーク制御システム(AICN)[6]のコマンドラインツールを用いて各ホスト上で実行可能なシェルスクリプトを生成している．対象アプリケーションの要件に基づいて「仮想ネットワーク構成情報」と「割り当て済みリソース情報」の 2 種類の設定ファイルが必要である．
- 仮想ネットワーク構成情報 … 仮想ネットワーク方式，Subnet/Gateway 設定，Firewall 設定など
- 割り当て済みリソース情報 … ホスト IP アドレス，使用する NIC，VLAN タグなど

【課題】
インタークラウドでの構築を行うために，ネットワーク構成情報とサーバリソース要求記述に対する統一的なインターフェースが求められる．現状の設定ファイルの記述形式は AICN の実装による独自形式であるため，依存関係を持つサーバリソース割り当て処理と一括して実行することができない．また，各ホストにおいて SSH などによるリモート接続が可能な状態を構築しておくことが前提となっており，アカウントや鍵情報の管理も独自に行う必要がある．

(c) ベースコンテナによるクラスタ構築

本システムでは，アーカイブ系コンテンツサービスを構成するホスト環境を「ベースコンテナ」として必要なソフ

トウェア群をコンテナ内に封入することにより可搬性を向上させている．

【課題】

インタークラウドのマルチホストを前提としてベースコンテナを管理できる仕組みが求められる．プロトタイプシステムでは，ベースコンテナのデプロイまでを Vagrant のプロビジョニング処理に含めることでクラウド環境へのサーバリソース割り当てから一括実行できるように自動化されている．しかし，コンテナ起動パラメータなどは独自実装のプロビジョニング・スクリプトに対して起動時に指定する形式となっており，汎用性に欠けている．

5.2 既存アプリケーション移植性

(a) サービス可搬構造

本システムでは，Web サービスの CMS ソフトウェアを含む Web サーバ環境をコンテナに封入したものを「アプリケーションコンテナ」と位置付け，ベースコンテナで構築されたホスト上に配置する．アプリケーションコンテナは共通のマシンイメージから生成され，永続的なデータをコンテナ内に保持しないことにより可搬性を確保し，必要に応じてスケールアウト構成でクラスタリング運用できることを想定している．

【課題】

アプリケーションの種類によっては，コンテナ形式にパッケージングするためのコストが高い場合がありうる．
例えば，永続化する対象となるデータ部分とアプリケーション部分が適切に分離できていないシステム構成の場合，コンテナ形式によるメリット（共通イメージのアップデート，スケールアウト構成など）が生かせないことになる．
本システムでは Web サービス系アプリケーションを対象としたが，他のタイプのアプリケーションに適用する際にはコンテナ方式が最適とは限らない．

(b) データ移行支援

アーカイブ系コンテンツサービスで実運用中のバックアップデータを共有ストレージと DBMS へリストアするために使い捨て前提の専用コンテナを用いる．リストア処理の要否は，Web サービス単位で分散 KVS に保存されたデータの状態から判断し，自動起動する．コンテナ内ではバックアップ先のオブジェクト・ストレージサービス(AICS)[7]からデータをダウンロードし，リストア処理完了後にコンテナ自体を廃棄する．

【課題】

対象のアプリケーション毎にリストア処理を実装する必要があるが，バックアップデータの読み込み，書き込み方式について統一的なインターフェースで実装できれば，より可搬性・運用性が高められるメリットがある．

5.3 HA 機能

(a) データ冗長保持

分散共有ファイルシステム・サーバ機能，DBMS Replication またはクラスタ機能を持つベースコンテナ群により汎用データストレージサービスを備えている．これにより，利用者がアップロードしたコンテンツデータなどのシステム稼働中に変化が生じるデータを保全し，可用性の向上を図っている．

- 分散共有型ストレージサービス

Web サービス毎にファイルボリュームを分割し，GlusterFS[8]で実装した分散分散共有型ストレージサービスに格納する．各アプリケーションからはネットワーク経由でマウントして利用する．

- DBMS サービス

アプリケーション要件に対応した DB サーバコンテナ・イメージを持ち込んで利用することができる．

【課題】

- 共有ストレージと RDBMS を対象としているが，アプリケーション要件しだいで NoSQL なども利用されることを想定しておく必要がある．
- Gluster FS の性能面での懸念があり詳細な検証を要する．プロトタイプシステムの構成において，1 サーバのボリューム数が増大した際に新規ボリューム作成(volume start)に時間（20～30 秒程度）がかかることが判明している．

(b) 負荷分散

アクセス数やデータ量の増大などに対応し，特定の Web サービスをスケールアウトすることでアクセス負荷を分散する．

本システムでは，Web サービス毎のスケールアウトの要否を運用者が判断し，オンデマンドにコマンドを実行する．その結果，ロードバランサとして動作している HAProxy 設定にも自動的に反映される仕組みを持つ．

【課題】

- 負荷やリソース使用状況の監視結果，ホスト環境の性能に基づくオートスケーリングに対応した仕組みが無い．
- アプリケーションコンテナの配置ロジックに関して，本システムではホストが自ノードにおいて当該 Web サービスが重複して起動しないことだけを条件としているため，リソース利用に偏りが発生しうる．自ノードに配備された個数やリソース状況に加え，他ノードの状況も加味した結果から自ノードで起動すべきかどうかを判断するなどの決定ロジックの実装が必要である．

(c) 障害検知・自動復旧

Web サービス毎の並列稼働数（稼働を維持すべきサーバ数）を予め設定しておき，Web サービス用コンテナの死活監視機能により障害を検知する．稼働状態はベースコンテナ上の分散 KVS に保存され，全てのノードから共有することができる．その情報を元に並列稼働数の不足状態を検知し，新規にコンテナを起動するといった自律制御を行う．

【課題】

- 死活監視の方式が現状ではコンテナに対する ping 応答の

みである．他に Web サーバとそのバックエンドの DBMS，ストレージサーバ，ベースコンテナを含めた各階層での監視が必要である．他のツールやフレームワークを組み合わせて監視機構を構築する．

- コンテナ毎のリソース監視に対応できていない．
- ベースコンテナが稼働するホストの監視をクラウド基盤側に依存してもよいか，またその監視結果をどのように反映し活用するかなどの検討が必要である．

5.4 システムアップデート

(a) システムアップデート支援

本システムでは，アーカイブ系コンテンツサービスの Web サービス用共通コンテナイメージを Docker Registry で管理し，OS やライブラリ等のアップデートに対して共通イメージの置き換えによって対応している．Docker Registry によりバージョン管理されるため，万が一アップデートによって問題が発生した場合には更新前の状態に戻すことが可能である．「ローリングアップデート」方式（1Web サービスにつき複数のコンテナで構成しておき，当該 Web サービスを利用可能な状態を維持しながら 1 コンテナずつ順番に更新する）により，稼働中の全 Web サービスをアップデートすることも可能だが，本システムでは未実装である．

【課題】

Docker Registry 自体には以下のような機能が無いため，他のツールを組みわせるか自前で実装する必要がある．

- 認証機構

Docker Hub を利用するか，Private Registry の場合に例えば Nginx や Apache をリバースプロキシとして配置して Basic 認証を行うなどの方法がある

- ローリングアップデート機能

他の管理ツールを組み合わせてスケジュール管理ができることが求められる

- コンテナイメージ以外の情報のバージョン管理

複数コンテナによるアプリケーション構成情報やそのテンプレートなども含めて管理するには別途コードリポジトリ機能を組み合わせて利用する

- 複数拠点間でのイメージデータの冗長保持

Rsync などによりリポジトリデータを同期させる

6. まとめ

アーカイブ系コンテンツサービスを対象とし，そのサービスバックアップについての VCP の適用性のために，本ユースケースに対応する VCP 実現のために必要な機能要件の抽出と方式検討を行った．さらに，評価のためのシステム構築と評価を実施した．今後の展開として,VCP の実システムへの適用を踏まえた評価・検証および機能の拡充を以下の点について実施する必要があると考えられる．

- 汎用サービスとして実現するデータストア機能のアーキテクチャに関する性能面,運用面での評価・検証を行う．
- アプリケーションコンテナの配置決定ロジックを拡充する．
- 自ノードに配備された個数やリソース状況に加え,他ノードの状況も加味した結果から自ノードで起動すべきかどうかを判断する,などの決定ロジックを実装する．
- アプリケーションコンテナ,ベースコンテナの監視機能を拡充する．
- アプリケーションコンテナ単位の性能監視機能により，スケールアウト処理の自動判断（オートスケール）を可能とする．

筆者は，1950 年代後半に海運業界で起こったコンテナ革命により進展した新しいグローバルな輸送システムを発生相当する動きが，クラウドコンピューティングをはじめとするトレンドとして IT 業界に現在起こっていると感じている．

Overlay Cloud アーキテクチャは，IT 業界での新しいグローバルなソフトウェア管理システムの構成方法の一つとして，その可能性を実システムへの適用を通じて確認して行きたいと考えている．今後も適用例を広げ，その検討状況について進展状況を報告する予定である．

謝辞 VCP の評価環境構築，VCP 評価にご協力頂いたアスケード社の村田典幸氏，那須野淳氏に謹んで感謝の意を表する．

参考文献

1) 合田 憲人，横山 重俊，吉岡 信和: アカデミックインタークラウドの構想，電子情報通信学会 サービスコンピューティング研究会，信学技報 113(376) 1-6， 2014 年 1 月
2) 横山重俊，政谷好伸，吉岡信和，合田憲人: Overlay Cloud で構成するインタークラウド環境，電子情報通信学会 サービスコンピューティング研究会，信学技報 115(72) 1-6, 2015 年 6 月
3) Docker: https://www.docker.io/ (2015.6.1).
4) AWS: http://aws.amazon.com/ (2015.6.1).
5) Vagrant: https://www.vagrantup.com/ (2015.6.1).
6) Shigetoshi Yokoyama, Atsushi Matsumoto and Nobukazu Yoshioka: Network traffic optimization architecture for scalability in academic inter-cloud computing environments, International Conference Performance Engineering 2014, International Workshop on Hot Topics in Cloud service Scalability, 2014 年.
7) Shigetoshi Yokoyama, Nobukazu Yoshioka: An Academic Community Cloud Architecture for Science Applications, 12th IEEE/IPSJ International Symposium on Applications and the Internet, SAINT 2012, 108-112 2012 年
8) GlusterFS: http://www.gluster.org/ (2015.6.1).