様式Ｃ－１９

科学研究費助成事業（科学研究費補助金）研究成果報告書

平成 25 年 4 月 22 日現在

機関番号：１２１０２
研究種目：基盤研究（Ｂ）
研究期間：2010～2012
課題番号：２２３０００００６
研究課題名（和文）：仮想計算環境のためのプログラミングシステム・フレームワークに関する研究
研究課題名（英文）：A Study on a Programming Framework for Virtual Computing Environments

研究代表者：
加藤　和彦（KAZUHIKO KATO）
筑波大学・システム情報系・教授
研究者番号：90224493

研究成果の概要（和文）：本研究では，系統的で拡張性に富んだクラウドコンピューティングシステム構築のためのプログラミングシステム・フレームワークの研究開発を実施した．粒度の小さな機能コンポーネントを提供し，それらをスクリプティングで組み合わせることでシステムを構成した．また，仮想マシンにコンポーネントの集合体を封じ込め，迅速かつ大規模で高堅牢性を有するシステム管理を可能とした．さらに，システム監視を含めた自律分散的な管理を可能とした．

研究成果の概要（英文）：We have built a programming framework for cloud-computing environments. We took a structural and extensible design approach by offering fine-grained functional components, which were encapsulated by virtual machines and combined together by scripting to make cloud systems in a rapid, scalable, and robust way. We also provide autonomous decentralized management of the clouds by offering self-monitoring and regulation capabilities.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
２０１０年度	5,900,000	1,770,000	7,670,000
２０１１年度	4,600,000	1,380,000	5,980,000
２０１２年度	3,300,000	990,000	4,290,000
総　計	13,800,000	4,140,000	17,940,000

研究分野：ソフトウエア
科研費の分科・細目：ソフトウエア
キーワード：オペレーティングシステム，分散システム，仮想マシン，クラウドコンピューティング

１．研究開始当初の背景

今日，インターネット上のシステム構築技術として，クラウドコンピューティング（以下 CC と略す）と総称されるシステム構築技術が，産業界から学術界まで，広く社会の興味を集めている．CC は，従来は組織内のサーバマシンやクラスタマシンによって構築していた情報システムを，インターネットの「向こう側」に集約し，情報インフラを共有することを可能にすることによって，情報インフラの有効利用，オンデマンド利用，スケーラビリティ等を簡便に提供することを可能にする．CC は，当初は一般エンドユーザを対象とした「パブリッククラウド」という形で利用が広まったが，企業内の大規模エンタープライズシステムを CC の考え方で構築しようとする試みも最近，見られるようになり，「プライベートクラウド」と称され，その形態も SaaS, PaaS, IaaS と広がりを見せている．

本研究代表者らは，JST CREST「情報社会を支える新しい高性能情報処理技術」領域にて研究課題「自律連合型基盤システムの構築」を 2003 年 10 月～2009 年 3 月において実施した．この研究では，広域ネットワーク環境上の分散システム技術と，仮想マシン技術を統合的に用いるアプローチを推し進めた．物理マシンを仮想マシン層によって抽象化し，仮想マシンの生成・起動・消滅・移動等の操作を，インターネット上の複数拠点に配備したクラスタマシンにまたがって動作させることを可能とした．そしてさらにこれらの機能を利用して，一拠点のサーバ環境では達成できないサービス提供技法を開発した．一つのサービスを，複数拠点の仮想マシンを提供することによって，サービス需要に応じてスケールアウト／スケールインさせ，計算機ハードウェアやネットワークに障害が発生しても，協調する仮想マシンがそれを自律的に発見，調整して補う動作をすることにより，サービスを持続する（サステーナブルとする）技術を開発した．CC という語は，2006 年頃から米国で使われ始めた語であるが，前述の CREST 研究の中で研究していた技術は，CC を構成する基盤技術の一種であると考えられる．本研究提案は，これまでの 6 年間に及ぶ研究経験に基づき，世界の CC 研究の動向も踏まえた上で，CC 環境を広く普及させるための基盤作成を目指して構想したものである．

Google, Amazon, Salesforce.com を始めとして，CC 的な商用サービスが始まっているが，CC のためのソフトウェア体系はいかなるものにすべきか，未だ確立はしておらず，筆者の研究グループによるこれまでの研究も含めて，試行錯誤を重ねながら，「作り込み」を重ねることによってボトムアップ的にその在り方を探ってきたのが現状である．Google 社の Google File System (GFS), MapReduce, BigTable（およびそれらのオープンソースソフトウェア(OSS)実装である Hadoop）等，あるいは，Amazon 社の Dynamo は，技術的な内容が公開された CC 基盤を構成する要素技術であって，CC システム全体を統括するものではない．また Salesforce.com 社も含め，それらの会社が商用提供する CC 機能は，技術的な詳細が公開されておらず，両社が CC のためのソフトウェア体系の構築に成功しているかどうかは不明である．

CC 基盤ソフトウェアに関する研究は，前述のような米国企業が先行しており，世界的に見ても，大学での研究は，要素技術に関する論文発表はあるものの，実用性を狙ったレベルの研究は多くない．数少ないシステムの一つが，カリフォルニア大学サンタバーバラ校によって開発された Eucalyptus システムである．同システムは，Amazon EC2 互換を謳っており，OSS 実装として公開されているが，その内容はさまざまなソフトウェアの組み合わせによって，EC2 の動作を模したものであり，例えば，スケールアウトさせるためのさまざまなアルゴリズムの試験実装，当研究グループが開発したサステーナブルシステム機能，あるいは，省電力化を図るための管理機能を実装する等，根本的な機能改良を組み込もうとしたときに，簡単には行えない．

２．研究の目的

本科研費研究においては，これまでの約 5 年に及ぶ CC 基盤の開発経験に基づき，系統的で，拡張性に富んだ CC システム構築のためのプログラミングシステム・システムフレームワークに関する研究開発を行う．この研究開発は，三つのアイデアに基づいている．

第一のアイデアは，CC システムをコンポーネントの組み合わせ（スクリプティング）によって構成することである．従来の CC 構築ツールである GFS, MapReduce, BigTable のように，高水準なソフトウェアツールの集まりとして CC システムを作るのではなく，よりソフト粒度の小さな機能コンポーネント（サービスコンポーネントと呼ぶ）の集合体と，それら集合体を組み合わせる（スクリプティングする）ことによって CC システムを構成する．サービスコンポーネントは，従来の「ソフトウェアコンポーネント」（例えば Java 言語における JavaBeans）よりも大きな粒度で，Web サーバ，FTP サーバ，データベースサーバ等の，ネットワークサービスを提供できる規模のものである．従来の CC 構築ツールは，本研究で作成する機能コンポーネントの組み合わせにより構成可能となる．

第二のアイデアは，仮想マシン技術を駆使することにより，サービスコンポーネントの集合体を一つの環境内に封じ込め，迅速かつ大規模なスケールアウト／スケールインおよび高堅牢性を有するシステム管理（可搬性, deployment 容易性, 正常性の検査可能性, 修復性）を可能とするである．

第三のアイデアは，システムの動作状態を系統的に監視すると共に自律分散的な管理を可能とするモニタを各仮想マシン内と，コンピュータノード内に配置することである．このモニタ自体が，インタフェースが定められたモジュールの集合体により構成され，機能拡張可能となっており，前述のシステム管理を実現するために必要なさまざまな機能を柔軟に組み込むことができる．

このシステム設計は，既に成熟した技術となっている Unix 等の OS，仮想マシン技術（特に代表者らが以前研究していた SoftwarePot システムやサステーナブルシス

テム)，および，最近 Web サービスの分野で注目を集めている考え方である SOA (Service-Oriented Architecture)等を着想の源としている．現代の OS の拡張性は，ユーザプロセスによるそれと，カーネルモジュールによるそれの組み合わせで得られている．前述のサービスコンポーネントがユーザプロセスに，モニタ機能がカーネルモジュールに対応する．サービスコンポーネントという形式でサービス機能を標準化し，モノとして可搬にする手法は，SoftwarePot およびサステーナブルシステムという，当研究が過去 8 年に及ぶ研究で蓄積してきた特色あるアプローチである．この考え方はネットワークサービスの組み合わせで大規模ソフトウェアシステムを構築していくという SOA の主張とも高い親和性を持つ．そして，センサー機能と管理機能を有するモニタを仮想マシン内およびコンピュータノード内に配備することで，スケーラビリティ，堅牢性，保守性を有する自律分散アーキテクチャの実現を系統的に行うことを可能にしている．

本研究が目指す事は，一つの高機能な CC システムを作ることではなく，拡張性に富んだ CC 基盤の枠組みを作り出すことである．この研究成果によって，さまざまな性質をもった CC システムを作り出す事が可能となり，実験的な機能の実装も容易になり，そして CC システムの社会普及を促すことが期待される．前述のユーザサービス・コンポーネントはユーザプロセスに，システム・コンポーネントはカーネル内モジュールに対応する．ただし巨視的に見たときに構造的な対応はするが，より微視的にみれば，複雑度は大いに異なる．ユーザサービス・コンポーネントは，仮想マシンであり，内部にゲスト OS を包含し，そのゲスト OS 上で，ユーザサービスを実装するためのプログラミング言語の実行時システム，多種多様なライブラリやフレームワーク，Web サーバ，ストレージ管理機能等を内包し，一般に複数プロセスにて構成される．一方，システム・コンポーネントの各々の複雑度は，既存のカーネルモジュールのそれと同程度になると思われる．Unix 上で既存プログラムを組み合わせ，スクリプティング可能とすることによって，高い生産性と保守性を提供していることに着想のヒントを得て，系統的で，保守性・生産性が高い CC 基盤を構築することが本研究の特色で，独創的な点である．この研究成果によって，CC の生産性が大きく向上し，社会普及を促すことが期待される．

３．研究の方法

本システムは，コンピュータノード内を構成する基本フレームワークと，ノード間のメッセージパッシングによって実現される自律型分散制御アルゴリズムよって構成される（図 1）．各ノードには仮想マシンモニタが動作し，その上でノードモニタと，仮想マシンが動作する．ノードモニタはコンピュータノード内でノードのハードウェアの状況，および，仮想マシンの起動，停止，動作状況の監視を行う．各仮想マシン内ではゲスト OS が動作し，その上でネットワークサービスを提供するサービスコンポーネント群が動作する．これらのコンポーネントを，拡張可能な自律型分散制御アルゴリズムを用いて制御する．

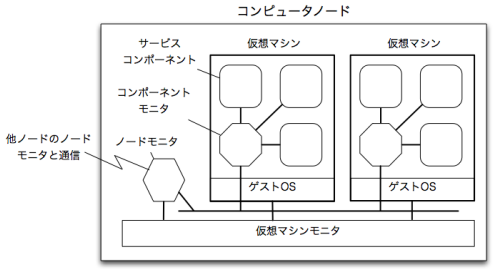

図 1：全体のシステム構成

４．研究成果

(1) 全体のシステム設計

まず，本研究では，前章の図 1 を実現するための全体のシステム設計を行った．

本研究では，ノードモニタ，コンポーネント，サービスコンポーネント，および仮想マシン，仮想マシンモニタなどのすべての資源を分散オブジェクトとして表現することにした．このアプローチにより，すべての資源を統一的に扱うことができるようになり，また後述するセキュリティ機構やキャッシュ機能なども，個別の資源によらず，動作させることができるようになる．

また，それぞれの分散オブジェクト間の通信は Java RMI による同期通信，またはアクターによる非同期通信により行われる．コンポーネントモニタについては，ノードモニタの縮小機能版を仮想マシン内に配置することで，全体の統一性を持たせるとともに，Java RMI によって外部から仮想マシン内部への通信も可能にしている．

それぞれのコンピュータノードの管理については，ノードモニタ同士で Gossip プロトコルにより，メンバシップ管理されており，動的なノードの参加や離脱などが自動的に対処される．

本研究では，以上のようなコンポーネントを提供した後，これらのコンポーネントを組み合わせた上位の高水準なアルゴリズムについては，スクリプトによって記述することができるようになっている．このアプローチによって，全体のシステムの柔軟性を高めている．

(2) アプリケーションのカプセル化

本システムでは，Apache や Hadoop などの既存のアプリケーションからサービスコンポーネントを自動生成する機能を有しており，レガシーなアプリケーションを容易に本システムに対応させることができる．また，一旦サービスコンポーネントが作成されれば，コンポーネントモニタやノードモニタが提供する仮想マシンや物理マシンの管理機能を利用し，信頼性や可用性，スケーラビリティを高めていくことができる．その際，本システムでは，これらの機能をスクリプト記述によって短期間で実現することが可能であり，さらに特定のサービスに特化した高機能なコンポーネントを追加すれば，サービスの品質をより高めることができる．

仮想マシンや物理マシンなどの資源と異なり，実際のクラウド環境では多種多様なアプリケーションが用いられている．これらのアプリケーションに対応するサービスコンポーネントを手動で作成していたのでは，実際の運用上の大きな障害となることが予想される．

そこで，本研究ではアプリケーションに対応するサービスコンポーネントの作成を支援するインタフェース記述からのコンポーネントのスケルトンの自動生成機構を作成した．その概要を図 2 に示す．

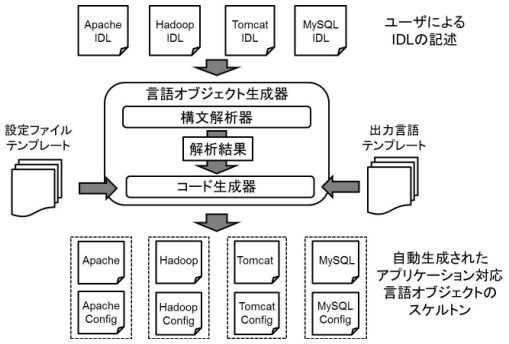

図 2：サービスコンポーネントの自動生成

本機構では，例えば Apache 用などのそれぞれのアプリケーションに対応するインタフェース記述を生成器に与えると，それに対応したコンポーネントの半自動生成を行う．あとは，自動生成が難しい部分を開発者が記述することで，完全なサービスコンポーネントとすることができる．

サービスコンポーネントを使用したスクリプト記述の例を図 3 に示す．ここでは，物理マシンに配置された仮想マシンの中で Apache が稼働しており，その設定を書き換える場合の例である．本フレームワークを使用すると，このような簡単な記述で操作を行うことができる．

```
kumoi> pms.map(_.vms.map { vm =>
  val apache = vm.local.apps(0)
                 .asInstanceOf[Apache]
  apache.maxClients = 1024
  apache.keepAliveTimeout = 5
  apache.reload()
`
```

図 3：スクリプトの記述例

実際にサービスコンポーネントを仮想マシンの中に組み込み，仮想マシンとその中で動作するアプリケーションを一体的に管理できることを確認した（図4）．実験では，Xenの中で動作するHadoopのスレーブノードを3台から9台まで伸縮させるスクリプトを記述し，実際に動作することが確認された．

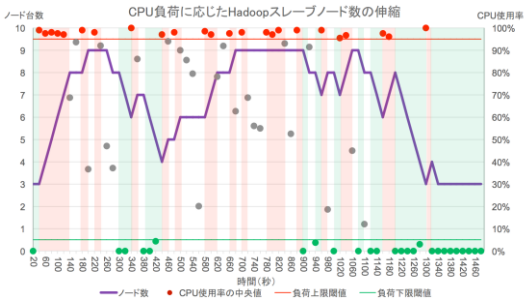

図 4：仮想マシン数の自動伸縮

(3) 障害対策

本フレームワークの完成度が高まった段階で，サービスの運用上必要不可欠な機能である障害対策を実施した．全体の可用性を優先させるFailure-Oblivious Computingによるアプローチと，代替となる資源をうまく活用するLineageの 2つのアプローチによる障害対策を実現した．

① Failure-Oblivious Computing

本研究では，障害の発生した要素を除外して計算をそのまま進める Failure-oblivious Computing をクラウド分野に応用した障害対策を実施した．本障害対策では，まず分散オブジェクトのメソッド呼び出しに失敗した場合に，代替となる値をリフレクションにより生成し，計算を進める方式を実装した．また，障害の影響の伝播を抑えるために，操作の過程で使用する関数の性質を活用し，伝播を抑える方式で実現した．これらの組み合わ

せによって，クラウド環境における特定の操作パターンについて，障害による影響を抑え，全体の処理を継続できることが確認された．

② Lineage を利用した障害対策

クラウドの多くの操作では，豊富な資源をもつ資源プールから，条件に合う資源を取り出し，その一部の資源に対して操作を行うということが繰り返される．この際，操作に使用されなかった冗長な資源に着目し，障害が発生した場合に，代替要素として取り出し，自動的に操作を再実行する機構を作成した．

(4) セキュリティとキャッシュ機構

障害対策と同時に認証・認可・課金などを行うセキュリティ機能も実現し，その導入による性能上の影響を抑えるためにキャッシュ機構の導入も進めた．これらの機能はすべて，図 5 の RMI 通信の中継機構によって実現されている．この方式は，特定の資源によらず，すべての通信を中継するため，効率的に実装を進めることができる．

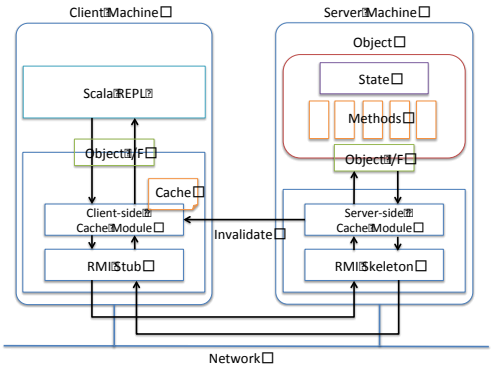

図 5：RMI 通信中継機構

① セキュリティ機構

上記の中継機構を利用し，下記のようなセキュリティ機能を実現した．

- 認証機能:アカウント名やパスワードを元に利用者が本人かどうか確認する．
- 認可機能:上記の認証情報を元に個別の資源へのアクセス権限をチェックする．
- 課金機能：資源の利用量を追跡し，適切な利用者に課金する．
- 監査機能:資源に対する操作履歴を記録する．
- 通信路の暗号化機能：機密性を高め，改ざんを防ぐために，通信を暗号化する．

② キャッシュ機構

本研究のキャッシュ機構では，すべての資源が分散オブジェクトとして抽象化されていることから，オブジェクトのメソッド呼び出しの結果をキャッシュとして保持することで，全体にキャッシュ機構を導入することができる．

キャッシュは図 5 の中継機構をもとに実装されているが，キャッシュのポリシーは図 6 のように指定する．本研究は，キャッシュの有効期間と無効化を組み合わせたキャッシュの一貫性管理方式を実装しており，それぞれの資源に応じて，ポリシーを制御することができる．

```
@remote trait HotPhysicalMachine {
  @persistcache def name(): String
  @cache(3000) def cpuRatio(): Double
  @cache(3000) def freeMemory(): Long
  @persistcache def maxMemory(): Long
  @persistcache def vmm(): VMM
  @cache def vms(): List[HotVM]
  @invalidate @nocache def shutdown()
  @invalidate @nocache def restart()
}
```

図 6：キャッシュのポリシー指定

(5) OpenFlow への対応

コンポーネントモニタとノードモニタ間の内部的な通信と，サービスコンポーネント同士のゲスト通信を分離し，柔軟性をより高めるため OpenFlow を基礎とした仮想ネットワークへの対応を行った．OpenFlow への対応では，これまでと同様に OpenFlow コントローラもサービスコンポーネントとして実現されており，従来の設計の自然な拡張として実現されている．また，Open vSwitch への対応も行っている．以上の結果として，仮想計算機，アプリケーション，ネットワークを全て一つの環境で連続して操作することができる．

本研究課題の意義や特徴は，CCに求められる信頼性や可用性を維持しつつ，柔軟性や俊敏性を高めている点にある．利用者の需要の変化や環境の変化に対応する柔軟性や俊敏性は，特にCCの普及とともに急速に必要とされており，CCの分野における重要な課題の一つである． 一方で，オンプレミスで構築されていた重要なサービスのクラウドへの移行が進んでいることから，特定のサービスごとの信頼性や可用性の維持も重要な課題である．本研究はこれら2つの課題解決の両立を目指して研究を実施し，この成果が応用されれば，実社会においても大きな影響や効果があることが期待される．

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計 6 件）

① 杉木 章義, 加藤 和彦, “仮想資源管理基盤におけるキャッシュ機構の導入”, 情報書学会：論文誌 (ACS), 6(1), pp. 31-44, 2013 年, 査読有.
② A. Sugiki and K. Kato, “Elements and Composition of Software-defined Data Centers”, ACM/IFIP/USENIX Middleware 2012, Demos and Posters Track, 3, 2 pages, Dec. 2012, 査読有.
③ 相川 拓也, 杉木 章義, 加藤 和彦, “サーバ・チューニング記述のためのスクリプティング環境”, 情報処理学会：論文誌 (ACS), 5(5), pp. 138-151, 2012 年, 査読有.
④ 杉木 章義, 奥畑 聡仁, 加藤 和彦, “クラウド基盤ソフトウェアにおける Failure-Oblivious Computing の導入”, 情報処理学会：論文誌 (ACS), 5(5), pp. 103-117, 2012 年, 査読有.
⑤ Sugiki and K. Kato, “An Extensible Cloud Platform Inspired by Operating Systems”, IEEE UCC 2011, pp. 306-311, Dec. 2011, 査読有.
⑥ Sugiki, K. Kato, Y. Ishii, H. Taniguchi, and N. Hirooka, “Kumoi: A High-Level Scripting Environment for Collective Virtual Machines”, IEEE 16th Int’l Conf. on Parallel Distributed Systems (ICPADS 2010), pp. 322-329, 2010, 査読有.

〔学会発表〕（計 5 件）

① 大山 裕泰, 杉木 章義, 加藤 和彦, “サービス指向の IaaS クラウドシステム環境の構築”, インターネットコンファレンス 2012, ポスター展示, 2012 年 11 月 16 日（富山国際会議場）.
② 奥畑 聡仁, 杉木 章義, 加藤 和彦, “クラウド基盤ソフトウェアにおける Lineage を利用した障害対策手法の検討”, SWoPP 鳥取 2012, 2012 年 8 月 2 日（とりぎん文化会館）
③ 杉木 章義, 奥畑 聡仁, 加藤 和彦, “クラウド基盤ソフトウェアにおける Failure-Oblivious Computing 導入の検討”, SACSIS 2012 先進的計算基盤システムシンポジウム, 2012 年 5 月 17 日（神戸国際会議場）
④ 相川 拓也, 杉木 章義, 加藤 和彦, クラウド環境におけるサーバパラメータ調整のためのスクリプティング環境, SWoPP 鹿児島 2011, 2011 年 7 月 27 日(かごしま県民交流センター)
⑤ 杉木 章義, 加藤 和彦, “国産クラウド基盤ソフトウェア Kumoi 開発の現状と今後について”, 情報処理学会 ComSys2010, ポスター発表, 2010 年 11 月 29 日（大阪大学中之島センター）

〔その他〕
ホームページ等
https://github.com/axi-sugiki/kumoi

６．研究組織
(1)研究代表者
加藤　和彦（KATO KAZUHIKO）
筑波大学・システム情報系・教授
研究者番号：90224493

(2)研究分担者
杉木　章義（SUGIKI AKIYOSHI）
筑波大学・システム情報系・助教
研究者番号：50536828
長谷部　浩二（HASEBE KOJI）
筑波大学・システム情報系・助教
研究者番号：80470045

(3)連携研究者
品川　高廣（SHINAGAWA TAKAHIRO）
東京大学・情報基盤センター・准教授
研究者番号：40361745