課題番号　jh130035-IS01

学際大規模共同利用環境を想定した クラウド基盤ミドルウェアの運用性向上に関する研究

杉木章義（筑波大学）

概要　本研究では，学際大規模情報基盤共同利用環境を想定したクラウド基盤ミドルウェアの運用性向上を目指す．本研究は既存のミドルウェアの一部の改良に留まらず，独自のミドルウェアを開発し，ミドルウェアのアーキテクチャ設計からの改善を目指す．また本研究は，単なるシステム実装の工夫による運用性向上を目的とするだけでなく，数理的検証を専門とする研究者との学際的な共同研究により，信頼性の高いミドルウェアの構築を目指す．さらに，学際的な多様な要望への対応や，管理容易性の向上を含めた総合的な運用性の実現を目指す．

1.　研究の目的と意義

(1) 研究の目的

クラウドコンピューティングは，近年，産業分野および学術分野ともに急速に普及している．学術分野では，従来の HPC（High Performance Computing）のグリッドコンピューティング技術をより発展させ，その応用も大規模数値計算によるシミュレーションから，ゲノム情報やストリームデータなどの大規模データ処理，また仮想化技術に基づくサーバ集約などへと大きく広がっている．よって，各拠点が有する計算資源を大規模情報基盤として統合し，同時に学際的な利用をより推進していくためには，より多彩な要望に応えるようにミドルウェアの研究や開発を進めていく必要がある．

研究代表者らは，先行研究の 2009 年からクラウド基盤ミドルウェア「Kumoi」を開発している．本システムは OS (Operating System) 設計に基づき，ミドルウェアの中核を出来る限りコンパクトにし，その他の多くの機能を拡張部分としてスクリプティングで実現することで，柔軟性とカスタマイズ性を高めている．この設計手法はマイクロカーネルの目標と共通しており，学際的な環境で求められる多様な要望に対応できると考えている．また本ミドルウェアはクラウド基盤ミドルウェアを研究開発するためのメタミドルウェアとしても利用できる．

また本研究課題では，情報基盤環境上での大規模な動作検証を行うことも目的としている．近年，数理的検証技術は，産業界において実稼働システムの設計にも適用されている．上記のミドルウェアの拡張部分でスクリプトとして動作するアルゴリズムを対象に，モデル検査法を中心とする数理的検証技術を適用し，これらの技術がこうした大規模なシステムに対しても有用であることを示す．

(2) 研究の意義

本研究は，独自の問題意識に基づき，クラウド基盤ミドルウェアのアーキテクチャの再設計を行っている点に特徴がある．現在，クラウド基盤ミドルウェアとして，さまざまなものが提供されおり，学術環境でも，これらのミドルウェアは広く利用されている．商用では VMware vSphere，Citrix XenServer などがあり，オープンソースでは，Nimbus，Eucalyptus，OpenNebula，Xen Cloud Platform，OpenStack などが提供されている．これらのミドルウェアはコンポーネントを組み合わせ，よく設計されていると言えるが，今後のメニィコアを始めとする計算環境の変化や，利用形態の変化などを考えると，クラウド基盤ミドルウェアに

対して，より一層の柔軟性やカスタマイズ性が求められていると考えられる．

また本研究は，OS（オペレーティングシステム）設計の視点からの再設計の試みであると考えられる．本研究グループでは，従来の OS の役割とクラウド基盤ミドルウェアの役割は，大局的な視点から見れば共通しており，その役割は「資源の抽象化」と「実行」の 2 つであると考えている．抽象化を行う対象が，CPU やメモリ，ディスクなどの単一ノード内の資源か，仮想マシンや物理計算機，ストレージなどのより粒度の大きい資源かという違いはあるが，その目的は共通している．また，実行の対象が OS のプロセスまたはスレッドか，仮想マシンかという違いはあるが，両者においてスケジューリングは重要な役割を担っている．本研究では，OS の設計手法だけに限らず，分散オブジェクト技術などやプログラミング言語の研究成果を取り込み，これらの問題に対処しており，過去の研究の文脈と連続性があるのも特徴である．

さらには，既存のクラウド研究の多くが上記の既存ミドルウェアの改良をもとにしているのに対し，本研究では独自のミドルウェア開発を進め，かつクラウド基盤ミドルウェアに必要とされる機能を一通り実現している点も特徴の一つである．

2. 当拠点公募型共同研究として実施した意義

(1) 共同研究を実施した拠点名および役割分担

東京大学

(2) 共同研究分野

超大規模情報システム関連研究分野

(3) 当公募型共同研究ならではという事項など

本研究は先行研究の成果を活用しているが，学際大規模情報基盤共同利用環境を想定し，実際の運用を目指して改良を進めた点が従来と異なる．また，従来は Non-technical Computing のクラウドを対象にしていたが，HPC を含めた Technical Computing へも対応できるクラウド基盤ソフトウェアを意識して，再設計を進めた点が異なる．

3. 研究成果の詳細と当初計画の達成状況

(1) 研究成果の詳細について

本研究は下記のようなスケジュールで研究を実施してきた．

・2009-2011 年度：先行研究（3 年間）
・2012 年度：JHPCN「クラウド基盤ミドルウェアのスケーラビリティ向上に関する研究」(1 年間)
・2013 年度：JHPCN「学際大規模共同利用を想定したクラウド基盤ミドルウェアの運用成功上に関する研究」(1 年間)

以降の節では，本プロジェクト全体で目指してきたシステムの概要について述べるとともに，本年度の成果についてもまとめる．

Kumoi の概要

本プロジェクトで研究を実施してきたクラウド基盤ソフトウェア Kumoi は，クラウドコンピューティングに求められる俊敏性と堅牢性の両立を目指してきたシステムである．クラウドでは，オンデマンドでの計算資源の確保や従量制による課金を基本としており，個々のテナントの要望に応じて，短期間で環境をカスタマイズできることが求められている．また一方で，仮想ネットワークや仮想ストレージなどに関するさまざまな要素技術が登場し，クラウド基盤ソフトウェアが扱わなければいけない計算資源の種類も増加しており，ミドルウェアの構造が複雑化している．

Kumoi は，以上の問題を解決するために，過去長年にわたり実施されてきた分散 OS や分散オブジェクトなどに関する研究成果を応用し，クラウドコンピューティングという新たな文脈のために再構成を実施したシステムである．

Kumoi の概要を**図 1** に示す．Kumoi では，クラウドに要求される俊敏性への対応から，高度なカスタマイズ性と拡張性を目標としており，中核となる機能を最小化し，その他の多くの機能を周辺機能としてスクリプトで実現するというアプローチを採用している．Kumoi では，ミドルウェアの機能をまず，カーネルとシェルに分割し，カーネル

はデータセンター内の各計算機で動作させ，シェルは管理者の計算機やクラウドのフロントエンド計算機で動作するように設計してある．

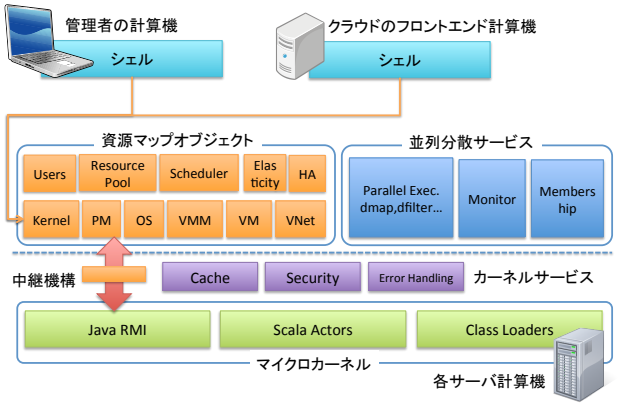

図 1：Kumoi の概要

カーネルでは，Java Remote Method Invocation (RMI)と Scala の Actor を組み合わせた通信機構のみをマイクロカーネルとして最下層に配置している．その上で，物理計算機や VM などの各資源を分散オブジェクトとして抽象化し，統一的なインターフェースを提供している．Kumoi では，これらのオブジェクトを資源マップオブジェクトと呼んでいる．一方で，スケーラビリティを向上させるために， `dmap()`や `dfilter()`などの並列スケルトンや，メンバシップ機能などの並列分散サービスを提供している．

クラウドの管理者や開発者は，これらの 2 つの機能を組み合わせ，実現したい作業をスクリプトとしてシェル環境で記述していく．本記述は Scala をベースとしている．

図 2 は具体的なスクリプト記述例である．まず，`deploy()`は引数で指定された物理計算機に VM を 1 台デプロイするための関数である．この配置の際には，VM の名前や起動イメージ，ネットワークアダプタなどのカスタマイズがそれぞれの VM ごとに行われる．このように Kumoi では，多くの設定やそのカスタマイズ作業をスクリプト上で完結させることができる．一方のスクリプト記述のトップレベルの `pms` は現在，システムに参加している物理計算機のリストである．この `pms` から CPU 使用率とメモリの空き容量に余裕がある計算機を `filter()`で抽出し，その中から 5 台の計算機を選択し，`deploy()`関数を呼び出して実際に VM の配置を実施している．

```
def deploy(p: (HotPhysicalMachine, Int)) {
  val (pm, no) = p
  val v = pm.vmm.createVM
  v.name = "centos6-" + no
  v.add(FileDiskAuto("/nfs/centos6-"+no+".img"))
  v.add(VirtualBridge(ovsbr, tap0, genmac(no))
  pm.vmm.add(v)
}

pms.filter(_.cpuRatio < 0.3)
   .filter(_.memory > 1024 * 1024 * 1024)
   .take(5)
   .zipWithIndex
   .foreach(deploy)
```

図 2：スクリプトの記述例

Kumoi を使用してクラウド環境を構築した場合の例を図 3 に示す．前節で述べたように，Kumoi のカーネルはサーバ資源プールの各計算機で動作する．一方で，シェル環境はクラウドのフロントエンドとなる計算機で動作させ，また必要に応じて管理者の計算機で起動することもできる．スクリプティングによって，さまざまな利用者インターフェースを柔軟に構築し，クラウドの利用者に提供することができる．なお，シェル環境はあくまで利用者インターフェースの構築に利用することを想定しており，クラウドの利用者が直接，シェル環境を操作することは想定していない．

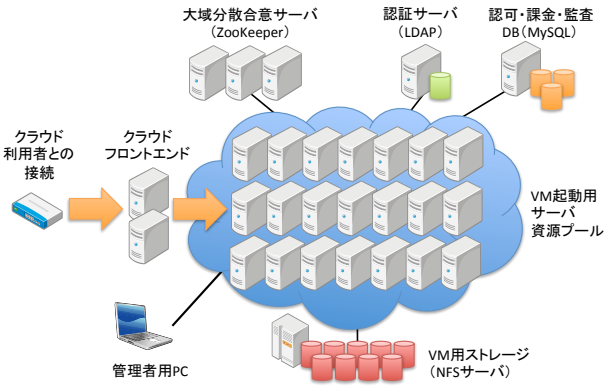

図 3：Kumoi を利用したクラウド環境の構築

Kumoi では，この他にもさまざまな計算機を利

用して，クラウド環境を構築する．まず，資源プールなどの大域的な機能の高可用性を実現するために，ZooKeeper を利用する．次に，LDAP サーバを通じて利用者認証を行い，その認証情報をもとに，さまざまな資源へのアクセス認可を行うデータベースを MySQL で構築している．また，クラウド環境の提供には欠かせない課金情報の記録や監査ログの記録なども MySQL で行っている．最後に，VM のライブマイグレーションを実現するために，VM 用のストレージも利用しており，現在は NFS サーバで実現している．

本年度の実施内容

前年度の JHPCN では，キャッシュ機構の導入による性能向上や各種障害対策機構の導入など，実際に運用可能なスケーラビリティの向上を目指してきた．本年度の JHPCN では，前年度に引き続きミドルウェアの運用性を向上させる改良を進め，仮想ネットワークへの対応やセキュリティ機構の実装を目指した．

（A）仮想ネットワーク機能の改良

仮想ネットワーク機能に関して，個々の計算機の仮想スイッチ設定である Open vSwitch への対応を早くから実施してきたが，さらに本年度は OpenFlow への対応も進め，クラウド環境全体のネットワークを統括的に管理できるようにした．これにより，学際大規模情報基盤共同利用環境で柔軟なネットワークが構成できるようになった．

本仮想ネットワーク機能は，従来からの設計の自然な延長で実現されている．仮想ネットワークに必要な仮想スイッチ，仮想ブリッジ，仮想ポートなどがすべてオブジェクト化されており，これらがすべて統一的なインターフェースを備えている．

図 4 に仮想ブリッジ上で仮想ポートを作成する場合の例を示す．各物理計算機には，Open vSwitch に対応する **vswitch** オブジェクトが結びつけられており，これを操作することで，仮想ブリッジを操作することができる．同様に各ブリッジでは，仮想ポートを操作することができるようになっており，図 4 ではこれを利用して仮想ポートを 2 つ作成している．図 4 の最後の命令は OpenFlow コントローラに接続する作業を行っている．

```
kumoi> val br = pms(0).vswitch.bridges(0)
br: kumoi.shell.vswitch.HotVirtualBridge = ovsbr1
kumoi> br.add(Port("tap0"))
res12: kumoi.shell.vswitch.HotVirtualPort = tap0
kumoi> br.add(TaggedPort("tap1", 100))
res13: kumoi.shell.vswitch.HotVirtualPort = tap1
kumoi> br.connectTo(ofc)
```

図 4：仮想ブリッジ上での仮想ポートの作成

また，OpenFlow のルールは**図 5** のように記述する．図 5 では，データパス上に OpenFlow 対応のスイッチが参加した場合と，パケットがコントローラに到着した場合のルールを記述している．この例のように，OpenFlow の操作も VM の操作と同じスクリプティング環境に統合されており，またパターンマッチを使用して，簡潔にルールを記述することができる． パケットの到着のケースでは，OpenFlow 対応スイッチにパケットをフラッディングで転送する命令を行っている．

```
// Installing a flow
val pattern = List(InPort(mtch.inPort),
        DlVlan(vlan),
        DlSrc(smac), DlDst(dmac),
        NwSrc(mtch.nwSrc), NwDst(mtch.nwDst))

sw.install(FlowMod(pattern, Cookie, Command.Add,
     TimeoutsDefault, PriorityDefault,
     bufId, Port.None, List(Flag.SendFlowRem),
     List(Output(engressPort, 0xFFFF))), context)
```

図 5：OpenFlow ルールの記述

OpenFlow 対応スイッチにフローをインストールする作業も記述することができ，その作業は**図 6** のように記述する．フローにマッチする条件をまず記述し，対応するアクションもスクリプトで記述する．

```
kumoi> ofc.add({ case DatapathJoin(sw) =>
  println("datapathId=" + sw.datapathId)
  true
}: OFlowRule)
kumoi> ofc.add({ case PacketIn(sw, bufId, inPort,
                totalLen, reason, data, context) =>
  sw.send(PacketOut(bufId, inPort, data,
    List(Output(Port.Flood, 0))), context)
  true
}: OFlowRule)
```

図 6：OpenFlow のフローインストール

実際の OpenFlow コントローラは，Floodlight をもとに実装されており，コントローラ自身も資源マップオブジェクトとすることで実現している（図 7）．他のサーバ計算機上にあるカーネルと RMI や Actor で通信可能とするために，OpenFlow コントローラ内部にもミニカーネルと呼ぶ機能が内蔵されている．

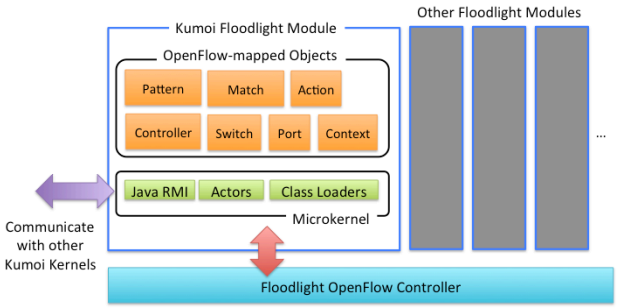

図 7：OpenFlow コントローラの実装

(B) 仮想ネットワーク機能の検証

以上までの仮想ネットワーク機能の検証をまずは手元の環境で実施した．本仮想ネットワークの機能は学際大規模情報基盤共同利用環境で最終的に利用することを想定しているが，現行の JHPCN 環境の機器的な制約から，容易に実験を進めることができないことが予想され，まずは手元の環境で実施することとした．

検証環境は，Xeon E3 1220v2 CPU，32GB メモリ，300GB SATA ディスクを内蔵したサーバ 4 台で構成されている．これらのサーバは OpenFlow 対応のスイッチである Pica8 Pronto 3290 で 1000BASE-T 接続されている．一方のソフトウェアは，CentOS 6.4（Linux 2.6.32）64 bit，KVM 0.12.1，Libvirt 0.10.2，Scala 2.9.3，Java 1.7.0，Open vSwitch 1.10.0 で構成されている（図 8）．

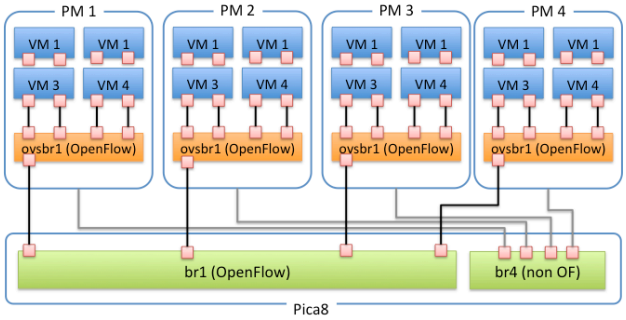

図 8：予備的な検証環境

この環境で各物理計算機あたり VM を 4 台，合計 16 台配置し，Open vSwitch を活用し，図 4 のように配線した．この VM の配置と Open vSwitch の設定に要したスクリプトの行数はわずか 14 行であった．一方の VM 間で通信可能とするために必要な OpenFlow ルールはその複雑さに応じて行数が変化するが，学習スイッチと同等の機能を実装した場合には 111 行で実現できた．検証環境で以上までの作業を行い，正しく通信が行われることを確認している．

(C) セキュリティ機能の改良

クラウド基盤ミドルウェア Kumoi のセキュリティ機能については，前年度でも簡単に説明しているが，本年度，本格的な動作検証を実施し，研究成果をまとめる作業を実施したため，ここでその成果について述べる．

Kumoi のセキュリティ機能として提供しているのは，下記の 5 つである．

- **認証機能**（authentication）：アカウント名やパスワードなどの情報をもとに，クラウド利用者が本人であるかどうか確認する．
- **認可機能**（authorization）：上記の認証情報をもとに，仮想マシン（VM）やストレージ，仮想スイッチ，OpenFlow コントローラなどの資源へのアクセスを制御する．
- **課金機能**（accounting）：VM やネットワーク転送などの資源の利用状況を追跡し，安全な場所

に記録する．クラウド利用者の異常な資源利用の傾向を追跡可能にするとともに，資源の実使用量に基づいた料金の請求のためにも用いる．

- **監査機能（audit）**： 資源に対する操作履歴をすべて記録しておくことで，クラウド利用者の不正な操作を事後的に追跡可能にする．
- **通信経路の暗号化機能（encryption）**： Kumoi の管理用の通信をすべて暗号化することで，通信内容の読み取りや改ざんを防止する．

以上の機能は，Kumoi の当初設計からの自然な拡張として実現している．Kumoi では，物理計算機，VM，仮想ネットワークなどのさまざまな資源を資源マップオブジェクトとして抽象化するとともに，統一的な操作方法を与えている．これらのオブジェクトは，実際には，遠隔オブジェクト（remote objects）で実装している．よって，この遠隔オブジェクトへの通信をすべて中継すれば，セキュリティに必要な完全調停（complete mediation）を実現できる．

この通信の中継は，すべての上記のセキュリティ機能を実現するために利用しており，また，統一的な操作方法は，特定の資源に依存しないセキュリティ機能の実現に役立てている．

認証機能

セキュリティ機能を有効化した Kumoi では，シェル環境をカーネルに接続した後，すべての資源アクセスの前に認証を要求する．認証機能は，JAAS（Java Authentication and Authorization Service）を使用して実装しており，現在は LDAP（Lightweight Directory Access Protocol）を使用して認証を行う．認証が成功すると，シェル環境上の auth と呼ばれる特別な変数に認証情報を格納する．auth は AAA 型のオブジェクトであり，認可機能をはじめとするさまざまな機能で使用する．

認可機能

認可機能では，資源マップオブジェクトのすべてのメソッド呼び出しを，カーネル通信中継機構によりフックし，呼び出しを許可するかどうか判定する．呼び出しを許可しない場合には，例外を送出する．

Kumoi では，ある物理計算機で動作している VM の一覧を取得する場合，**図 9** の内部呼び出しを必要とする．これは実際の資源の物理的な階層構造を反映しており，カーネルからの物理計算機の取得，ハイパバイザーの取得，VM 一覧の取得へと 3 回のメソッド呼び出しとそれぞれの権限を必要とする．

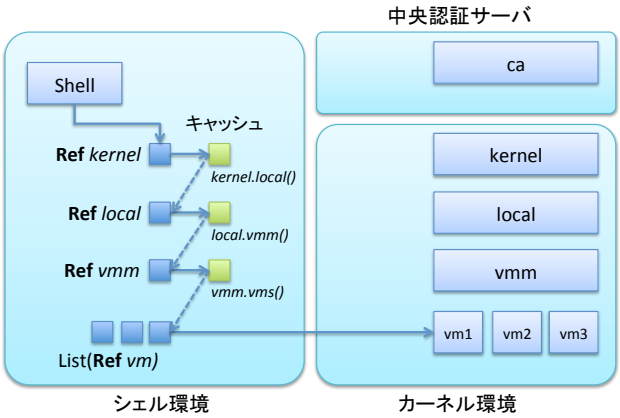

図 9：セキュリティ機構とキャッシュの連携

しかしながら，図 9 のように本機構はキャッシュ機構とも連携しており，中央認証サーバでの認可処理，カーネルでの資源アクセス処理がともにキャッシュが有効な間について省略される．

課金機能

課金機能では，課金処理専用のアクターをそれぞれのサーバ計算機に配置し，定期的に資源使用量を中央認証サーバに報告させることで実現している．Kumoi では，アクターを OS のプロセスのように使用している．また，VM の起動時や終了時など注意が必要な場面では，資源マップオブジェクトが直接課金処理を依頼できるようになっている．

監査機能

監査機能では，カーネル通信中継機能を使用し，資源マップオブジェクトのメソッド呼び出しをすべて記録する．時刻やホスト名を含む詳細な呼び出し情報を記録するとともに，呼び出しの成功の

可否までを含めて記録することで，不正な操作の事後追跡を可能にする．

通信経路の暗号化機能

セキュリティ機能の拡張では，データセンター内部で使用する Kumoi の管理用の通信を暗号化する． Kumoi で暗号化が必要なのは，下記の 3 つの通信である．これらの暗号化は， 個別に SSL（Secure Socket Layer）化することで実現した．

- **RMI 通信**: 資源マップオブジェクトへの同期的なアクセスに使用する．
- **アクター通信**： 非同期的に行うメッセージ通知に使用する．また，Kumoi 内部の分散アルゴリズムの実装に使用している．
- **遠隔クラスローダ通信**： シェル環境で定義したクロージャをカーネル環境で実行する場合など，遠隔のクラス定義を必要とする場合に用いる．

(2) 当初計画の達成状況について

本研究は,先行研究の成果および前年度の JHPCN 採択課題の成果を受けて，着実に実施した．具体的な成果として，仮想ネットワークへの対応やセキュリティ機能の拡充を進めた．

4. 今後の展望

本研究は先行研究の 2009 年より研究を開始し，3 年間研究を進めた後,JHPCN では 2012 年度,2013 年度と 2 年間にわたり研究を進めてきた．その過程で，クラウドコンピューティングをとりまく状況も大きく変化してきている．2009 年当時は，クラウドの研究を実施するためにミドルウェア自身から整備する必要があったが，2009 年当時には存在しなかった OpenStack などの多数の企業開発者がフルタイムで参加し，オープンソースで開発されるようなミドルウェアも多数登場してきている．

以上のような状況を踏まえて，現在，学際大規模情報基盤で用いられている CloudStack，OpenNebula などのクラウド基盤ソフトウェアや，今後検討される可能性がある OpenStack などのクラウド基盤ソフトウェアにこれまでの成果をフィードバックすることが考えられる．これらのクラウド基盤ソフトウェアは，Kumoi と比較して柔軟性はわずかに劣るものの，プログラムから操作するための一定の API を備えており，研究を進めることが可能である．また，Kumoi 自身もこれまでの成果をオープンソースとして公開しており，これを利用して第三者が研究を実施することが可能となっている．

5. 研究成果リスト

(1) 学術論文（投稿中のものは「投稿中」と明記）

[1] 杉木 章義, 加藤 和彦,“クラウド基盤ミドルウェア Kumoi のセキュリティ設計と実装”,日本ソフトウェア科学会:コンピュータソフトウェア(ソフトウェア論文特集号), 30 (4), pp. 3-17, 2013 年 11 月．

(2) 国際会議プロシーディングス

[2] A. Sugiki, ″An Integrated Management Framework for Virtual Machines, Switches, and their SDNs″, In Proc. of the 19th IEEE Int'l Conf. on Networks (ICON 2013), Special Session on Software Defined Network, pp. 1-6, December 2013.

(3) 国際会議発表

(4) 国内会議発表

(5) その他（特許，プレス発表，著書等）

[3] GitHub
https://github.com/axi-sugiki/kumoi