クラウドコンピューティングでの高度セキュリティ実現方式の提案
ー 情報処理委託内容の秘匿方式 ー

2010／11／19

電子情報通信学会
SWIM研究会
於
東海大学高輪キャンパス

㈱アイエスイーエム 宮西洋太郎
仙台ソフトウェアセンター嘱託
宮城大学客員教授

概要

- パブリッククラウドなどの自社外のクラウドを利用する場合，データとその処理内容は，暗号化したとしても処理を実行するときには，復号化される．
- すなわちクラウド提供者には，データと処理内容は原理的には筒抜けになってしまう．
- 通常は，クラウド提供者とユーザとの間でSLA契約を交わし，秘密保持を約束する．外部からの不正アクセスについては，クラウド提供者のコンピューティング設備をハード的，ソフト的にガードすることによってセキュリティは守られる．
- しかし，悪意による内部からの不正アクセスに対しては，有効ではない．
- 本提案は，「情報処理を委託しているのだが，委託内容について委託先には知られたくない」といういわば，虫のよい要求にたいしての解決策を提案するものである．
- 現在は構想の段階であり，今後実証することが課題である．

内容

- 1. はじめに
- 2. 基本的な考え方
- 3. 実現の方法
  - 3. 1. プログラムの分割と動作原理
  - 3. 2. プログラムの分割と事前のデプロイ
  - 3. 3. データストアとの関係
  - 3. 4. 結果の合成
- 4. 今後の計画

1．はじめに(1)

- 近年，クラウドコンピューティングはコンピュータ利用分野全体への浸透は目覚しく，従来の**情報検索**といったビジネスに直接的な関わりの少ない利用分野から，**ビジネスの基幹業務への利用**が進展している．これは，コンピューティング資源に要する**コスト**(初期コスト，維持コストともに)を低減でき，**抽象性**(サーバ非依存性)と**弾力性**(規模拡張性)といったクラウドコンピューティングのもつ本質的な利点が，世の中に受け入れられつつある現われであろう．

- 今まさに，情報システムは**「作る」から「使う」へ，「所有する」から「借用する」**といった時代に入ろうとしている[1][2]．

1．はじめに(2)

- このようなビジネス分野へのクラウドコンピューティングの浸透において，ある種の看過できないリスクが存在している．ビジネス活動におけるデータは，ビジネス活動の「魂，エッセンス」であり，ビジネス活動をつつみ隠すことなく表現している(ビジネス活動から情報世界へのマッピング)．さらには，これらのデータをどのように処理しているかという点についても，ビジネス活動の動的な関連を表現しているので，同様に，ビジネス活動のエッセンスである．すなわち，コンピュータの中に蓄積された「データ」(ビジネスで扱われているデータ)と「プログラム」(それらのデータをどのように処理しているか)の内容は，ビジネス活動の「魂，エッセンス」である．すなわち，データと処理方法はビジネスそのものであると言っても過言ではない．

1．はじめに（3）

- さらには，データとプログラムは，パブリッククラウドの場合，**世界中のどこに存在するかわからないサーバ**の中に保管される．これは，産業の国際競争を考えるとき，競争相手国によって不正アクセスされるリスクが大きい．
- 上記の理由から，「**ビジネス上の競争相手に自社の手の内を知られたくない**」という点で，コンピュータの中の**データとプログラム**には，**高度な秘匿性**が求められる．
- クラウドコンピューティング形態の場合，データ，プログラムともに，自社以外の外部に依存することになる（ここでは，プライベートクラウドではなく**自社外のクラウドを想定**）．その場合，ビジネス上の競争相手から手の内（データとプログラム）を知られたくないのは最小限の要求であろう．

1．はじめに(4)

- この要求に対して，クラウド提供者は，提供している情報システムのセキュリティを高め，外部からの不正アクセスを防止し，利用者に安全を保障することによって，この要求に対応している．これによって，外部からの不正アクセスに対する対策はほぼ達成できている．

- それでも，安心できない場合，「データの保管とデータ処理をクラウド提供者に委託しているが，その内容については委託している相手にも知られたくない」といったいわば虫のよい希望をもつことも厳しい競争の下では不自然ではない．
- クラウド提供者とクラウド利用者との間で，SLA(Service Level Agreement)といった契約によって，上記の不安を一応決着している．

1．はじめに（5）

- しかし，**契約による保障**は，契約は遵守されるという性善説に立脚するものであり，**内部当事者の不正行為を磐石に防止するものではない**．すなわち，性悪説に対して効果をもつものではない．
- 現在の技術では，クラウド提供者の情報システムのセキュリティを高め，**外部からの侵入を高度に防止することができる**．保管しているデータを暗号化して，たとえ，**外部にデータが漏洩しても，被害をくいとめることもできる**．しかしたとえデータが暗号化されていても，データを処理する段階になると，データは復号化され，ビジネス上での意味のあるデータ処理がなされる．すなわち，**CPUが扱うデータと命令は平文処理であり，CPUにはこれらの内容がわかってしまう**．

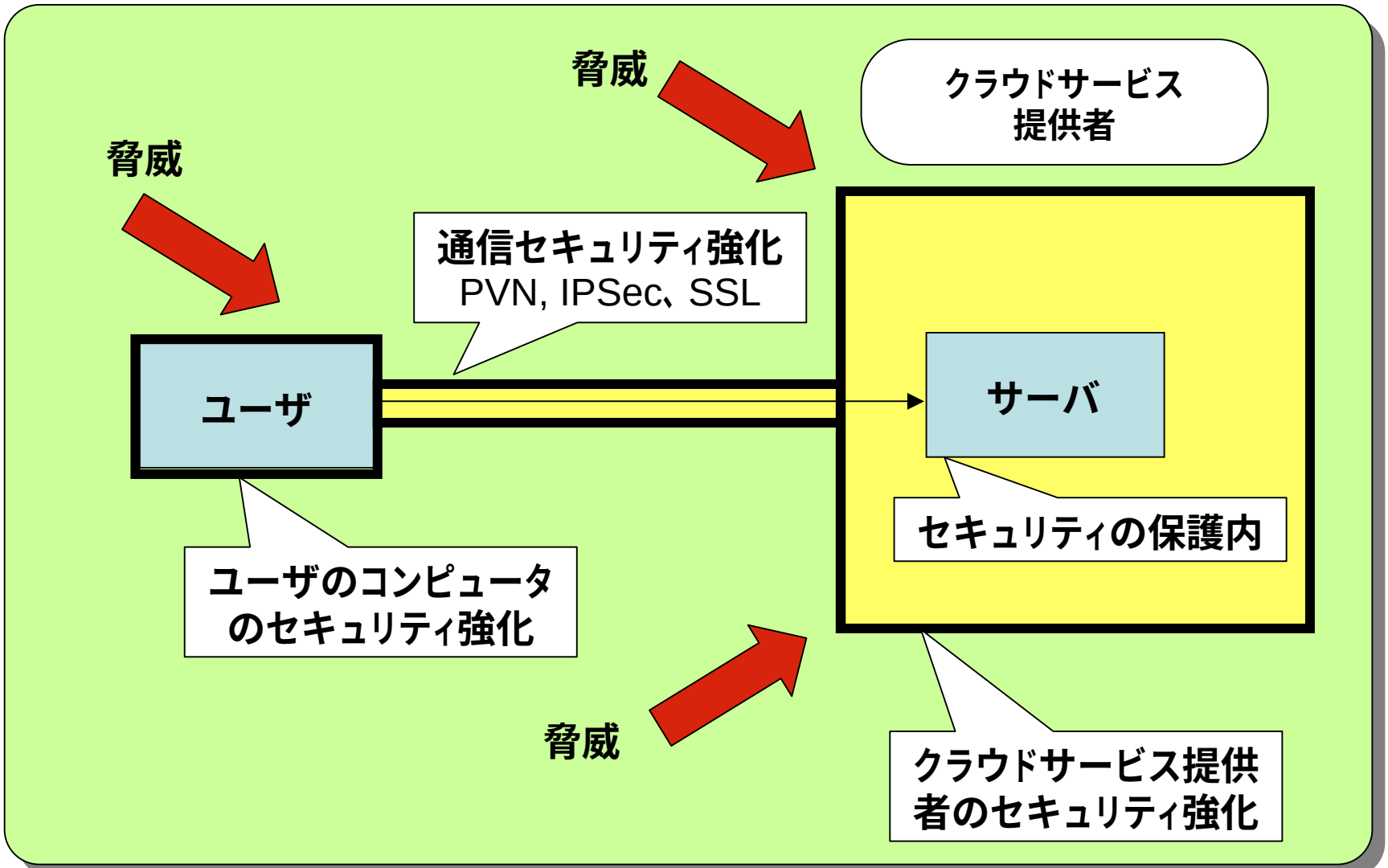
1．はじめに（6）

- これは，とりもなおさず，CPUを管理する人間にもわかってしまうことになる．すなわち，**クラウド提供者の内部側には，データ及びデータ処理が筒抜けになってしまう**．
- この事態を**契約や運用規準によるのではなく**，本提案では「技術」によって，ニーズ「データの保管とデータ処理をクラウド提供者に委託しているが，その内容については委託している相手にも知られたくない」を実現することを目的とすることである．すなわち，「**クラウド提供者に情報処理内容を知られたくない（言葉は悪いが，盗み見されたくない）**」ということが本提案の目的である．
- 前述のように，ニーズは単体CPUとしては原理的に実現不可能であると考えられるので，**システム的な解決方法**が必要となる．

1．はじめに(7)

- 現時点では，まだ提案の段階であり，実証については，今後の課題である．
- 以下本発表は次の構成である．2.は基本的な考え，3.は実現の方法，4.は今後の計画について述べる．

1．はじめに（8）


脅威
クラウドサービス
提供者
脅威
通信セキュリティ強化
PVN, IPSec、SSL
ユーザ
サーバ
セキュリティの保護内
ユーザのコンピュータ
のセキュリティ強化
脅威
クラウドサービス提供
者のセキュリティ強化

2．基本的な考え方(1)

- 1.に述べたように，原理的にニーズを満たすことは不可能であると考えられ，それならば，悪意での情報漏洩の確率(リスク)を小さくするという考えに立つ．すなわち，データと処理内容を分割し，ランダムに割り当てたサーバに部分的なデータ処理を委ねる．そのようにして，部分的な処理内容が悪意で読み取られても，全体が理解できないようにする．
- 処理内容の分割は，粒度の粗い順に，
- ①プログラムレベル分割
- 情報システムを構成するプログラム自体(JavaによるWebアプリケーションならば，JSPやサーブレット)を情報システムの分割(プログラムレベル分割と称する)とみなし，別々のサーバに分配配置(デプロイ)する方法が考えられる．

2．基本的な考え方(2)

- **②オブジェクトレベル分割**
- **次に各プログラムを更に分割することも考えられる．プログラムの分割には，ある程度意味のある粒度（構造化設計されたモジュールやオブジェクト指向設計されたオブジェクト）での分割（オブジェクトレベル分割と称する）する方法が考えられる．オブジェクト指向設計での分散オブジェクトを扱っている技術にCORBA(Common Object Request Broker Architecture)があり[3]，実証時の参考にする．**
- **③マシン命令レベル分割**
- **さらには，最も細かい粒度でのマシン命令レベルでの分割（マシン命令レベル分割と称する）が考えられる．**

2．基本的な考え方(3)

- ①，②の場合，ソースコードでサーバに分配する場合もありうるが，その場合，データの名前，プログラムの名前，変数の名前，オブジェクト(クラス)の名前，属性の名前を機械的な名前に変更し，名前から意味が読み取れないようにする．例えば，「customer」といった変数名あるいは属性名を「a5897241」といった意味を類推しがたい機械的な文字列の名称に変換する(名称の無機質化と称する)．名称を暗号化することと原理的に同等である．
- データストア(データベース)は暗号化して，1社のクラウド提供者のサーバに保管を委ねる．そのクラウド提供者は，データ処理を行わず，保管のみとする．

2．基本的な考え方(4)

- **データ処理(プログラム)は，上記のように①～③の粒度レベルで分割され，複数のクラウド提供者のサーバに割り当て分配される．分配時には暗号化を行い，分配先で復号化される．分配されたサーバは処理に必要なデータをデータ保管サーバから入手しデータを復号化し，データの処理を行い，結果を暗号化し，データ保管サーバにもどす．この部分については，平文の処理となり，悪意ある読み取りは可能であるが，情報システムとしてのデータ処理全体における一部分のみであり，データ処理全体は読み取ることができない．**

2．基本的な考え方(5)

- 部分処理された結果を**合成し，全体処理の結果**を得る．合成方法は，部分処理結果を逐一ユーザに返していく方法（**ハブ方式**と称す）と逐一ユーザに返さず，次の部分処理サーバに受け渡していく方法（**リング方式**と称す）が考えられる．
- 上記の機能をできる限り**ミドルウェアで吸収し**，アプリケーションプログラマにとって，意識しないようにする．全体概念図を図1に示す．

2．基本的な考え方（6）

- **全体概念図**

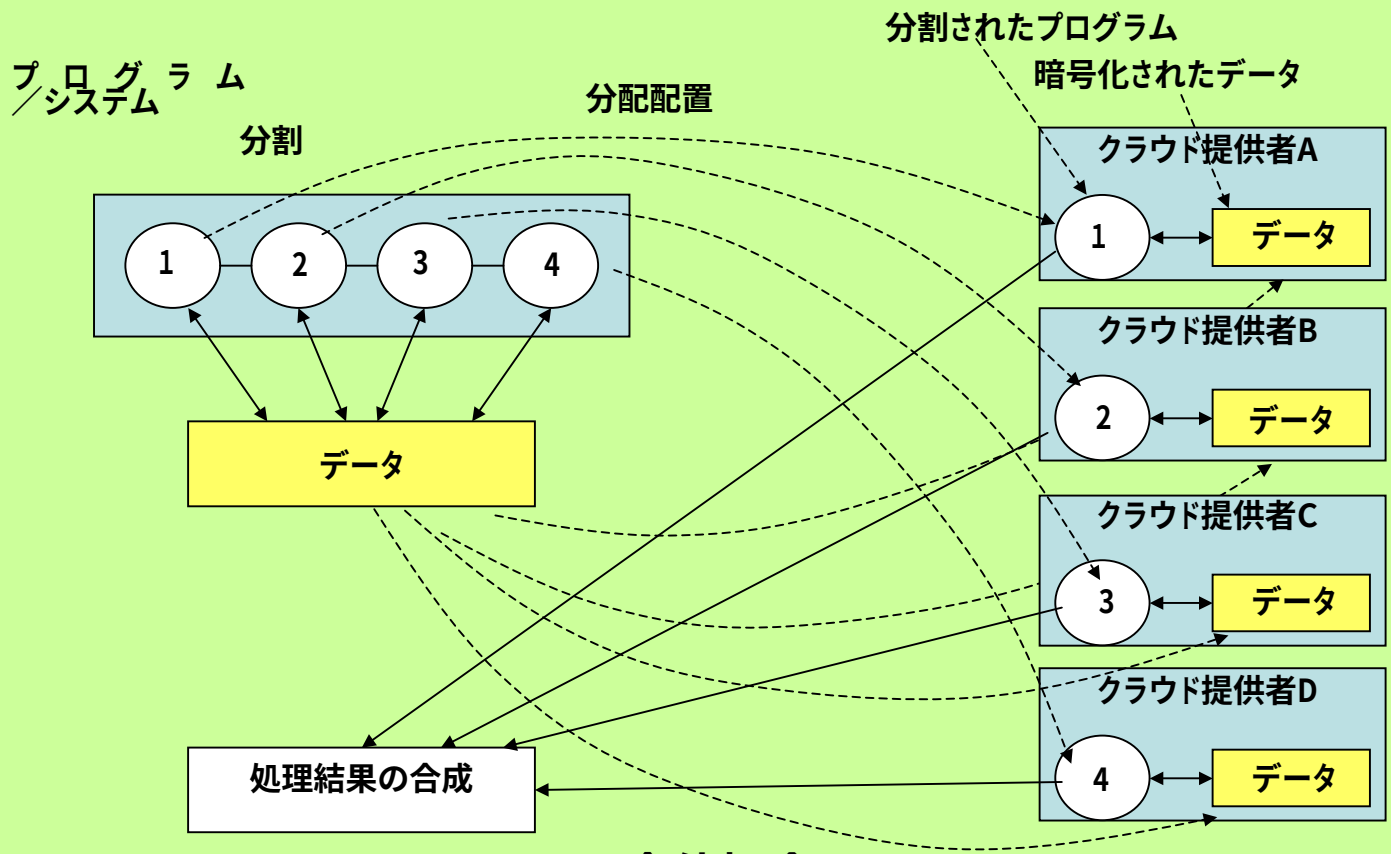


図1　全体概念図

2．基本的な考え方(7)

- 従来からソフトウェア工学において**可視性**(visibility)**を高めること**は，生産性や品質向上のために必要であった．一方，本稿での提起している問題は，できる限り**可視性を抑制すること**である．この**両立をミドルウェアに担わせる**．

3．実現の方法(1)

- **3.1. プログラムの分割と動作原理**
- **3.1.では分割が最も厳しい条件になると考えられるマシン命令レベル分割を考察する．プログラムをほとんど任意に分割し，正常な動作結果が得られることは原理的に可能である．**
- **その一例として，単一CPUの場合のオペレーティングシステム(OS)において，割り込み発生によりタスクスイッチングを行い，優先処理が行われ，その後に元のプログラム(タスク)に処理が戻り，正常に処理が完了する．このことからも明らかである．**
- **ただし，この場合，同一のCPUでのタスクスイッチングであり，状態変数のセイブ，リストアは同一のCPU及びOS間で行われる．図2に単一CPUの場合のタスクスイッチを示す．**

3．実現の方法(2)

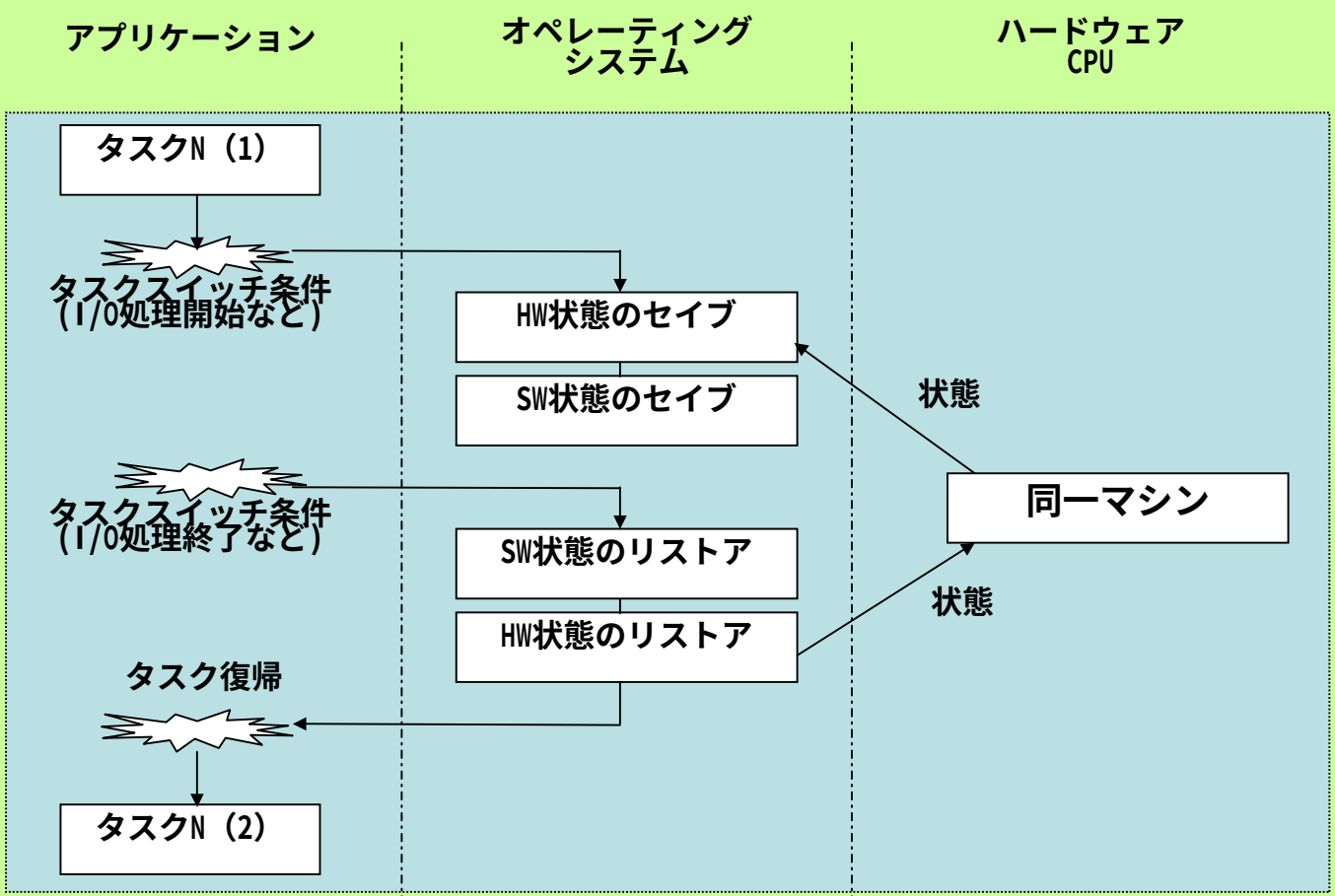


図2 同一マシンでのタスクスイッチ

3．実現の方法(3)

- これに対して，本提案では，仮想マシン間でのタスクスイッチングを行わせることになる．図3に**仮想マシン間でのタスクスイッチ**を示す．
- 仮想マシン間でのタスクスイッチでは，アプリケーションとOSとの間に，**ミドルウェア**のレイヤを設け，そのレイヤで仮想マシン間での受け渡し処理を担わせる．
- プログラム分割単位での動作が終了すれば，ミドルウェアは，現在の仮想マシン(A)のHW状態，仮想マシン上のSW状態をセイブして，セイブ情報を次の仮想マシン(B)に転送する．それを受けた仮想マシン(B)は，仮想マシンHW状態，仮想マシン上のSW状態を自分にリストアして，仮想マシン(B)に予め割り当てられている部分処理を実行する．

3．実現の方法(4)

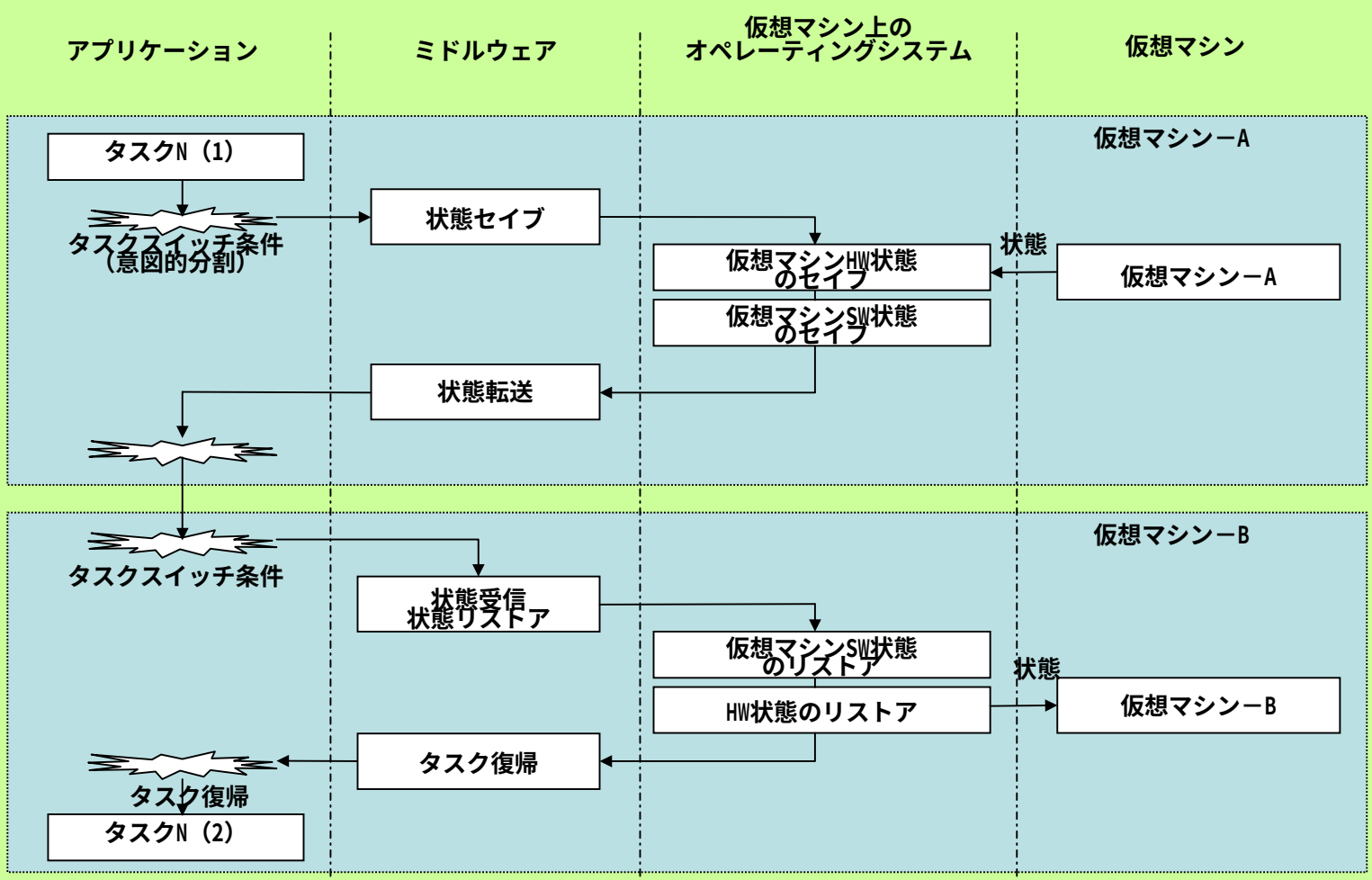


図3 仮想マシン間でのタスクスイッチ

3．実現の方法(5)

- 3.2. プログラムの分割と事前のデプロイ
- まず，プログラムに「分岐(選択)」や「繰り返し」がない場合を考える．あるプログラムの実行モジュール(バイナリー)をマシン命令レベルで分割する(Javaコードの場合は，中間コード)．分割前のプログラムに対して，分割されたプログラムを**部分プログラム**と称することにする．部分プログラムは，**暗号化**して，あらかじめデータ処理を担当するクラウド提供者に**配置(デプロイ)**しておく．配置されたクラウド提供者の仮想マシンでは，この暗号化された部分プログラムを復号化して，3.で述べたタスクスイッチングのタイミングで実行される．
- 「分岐」や「繰り返し」が存在するプログラムは，分割に制約を設け，**途中での分割を行わないものとする**．図4にこれらを含む場合の分割時の制約について示す．

3．実現の方法(6)

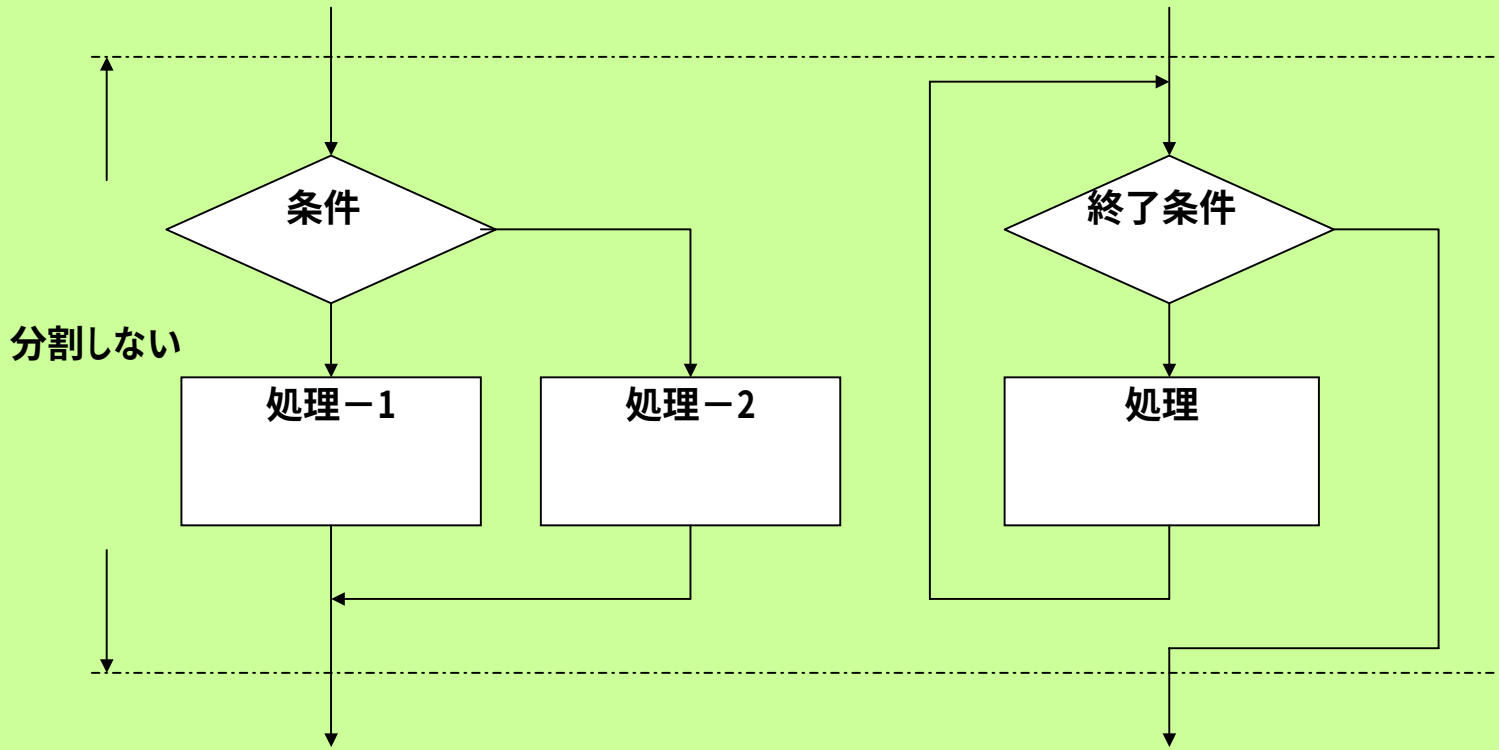


図4 分岐，繰り返しを含む場合の分割の制約

3．実現の方法(7)

- 分岐後長い処理を行う場合には，図5のように分割することにより，プログラム分割を小さいサイズにすることができる．

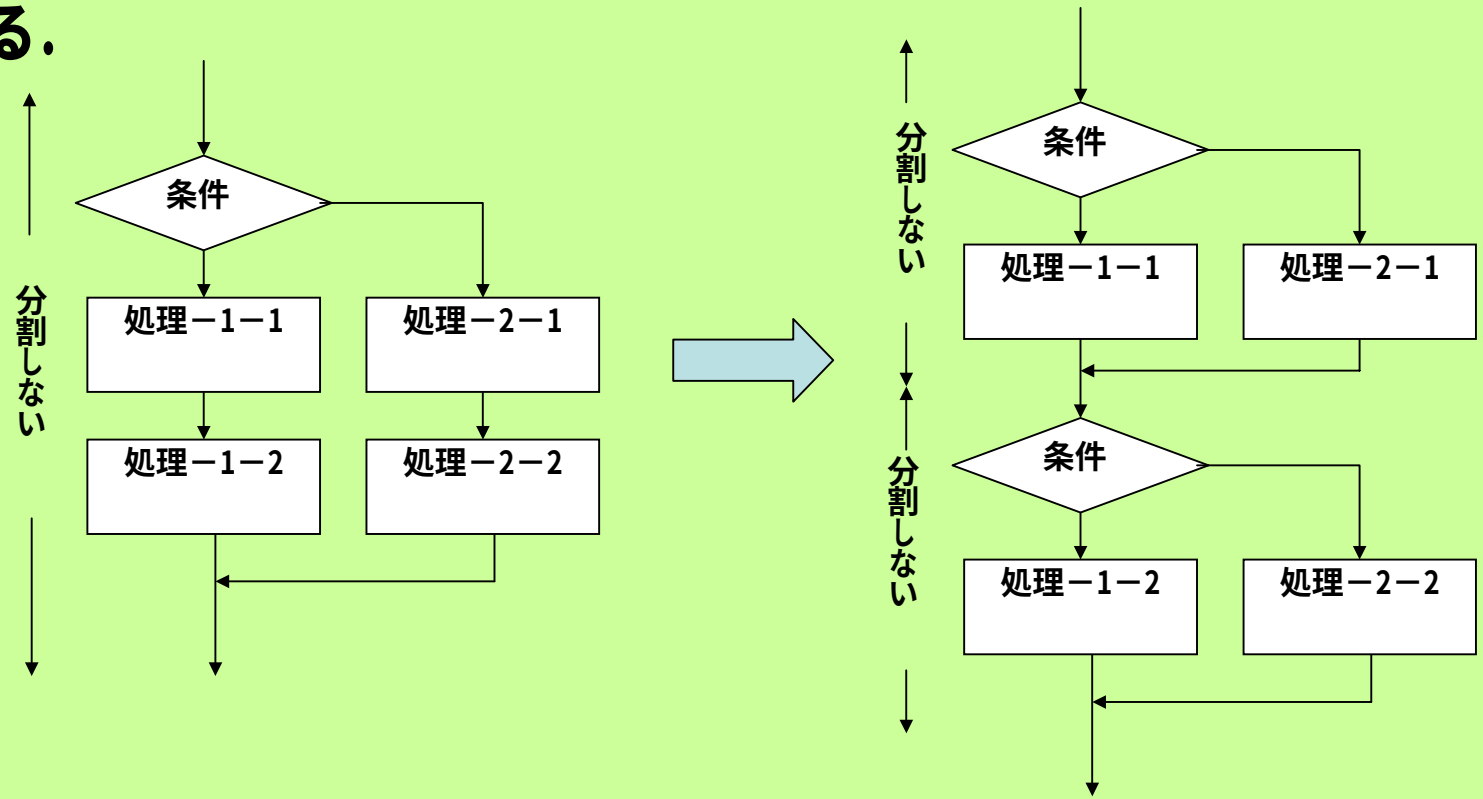


図5 分岐後の長い処理を短くする方法

3．実現の方法(8)

- 長い処理の繰り返しも繰り返しを分割することによって，同様に分割単位を小さくすることができる．
- プログラムから使用するサブプログラムやクラスについては，分割対象から外し，各部分プログラムが使用するサブプログラム，クラスも事前に配置しておくものとする．サブプログラムやクラスも当然ながら，2.で述べた名称の無機質化は行う．
- サブプログラム，クラスを分割対象から外すことにより，意味の残った分割となり，分割プログラムの処理内容について，情報漏洩のリスクは増えるが，本提案の実現メカニズムが熟成してくれば，次に考えることとする．

3．実現の方法 (9)

- 3.3. データストアとの関係
- タスクレベルでの処理の継続性は，3.，4.に述べたとおりであるが，アプリケーションとしての処理の完結は，一般にデータストアとの関係 (読み取り，書き込み) を伴うものである．
- 前述のプログラムレベル分割やオブジェクトレベル分割といった粗い粒度でのプログラム間でのデータの引渡しはデータストア経由によることが多いと考える．図6にプログラムレベル分割の場合のプログラムとデータストアの関係を示す．

3．実現の方法(10)

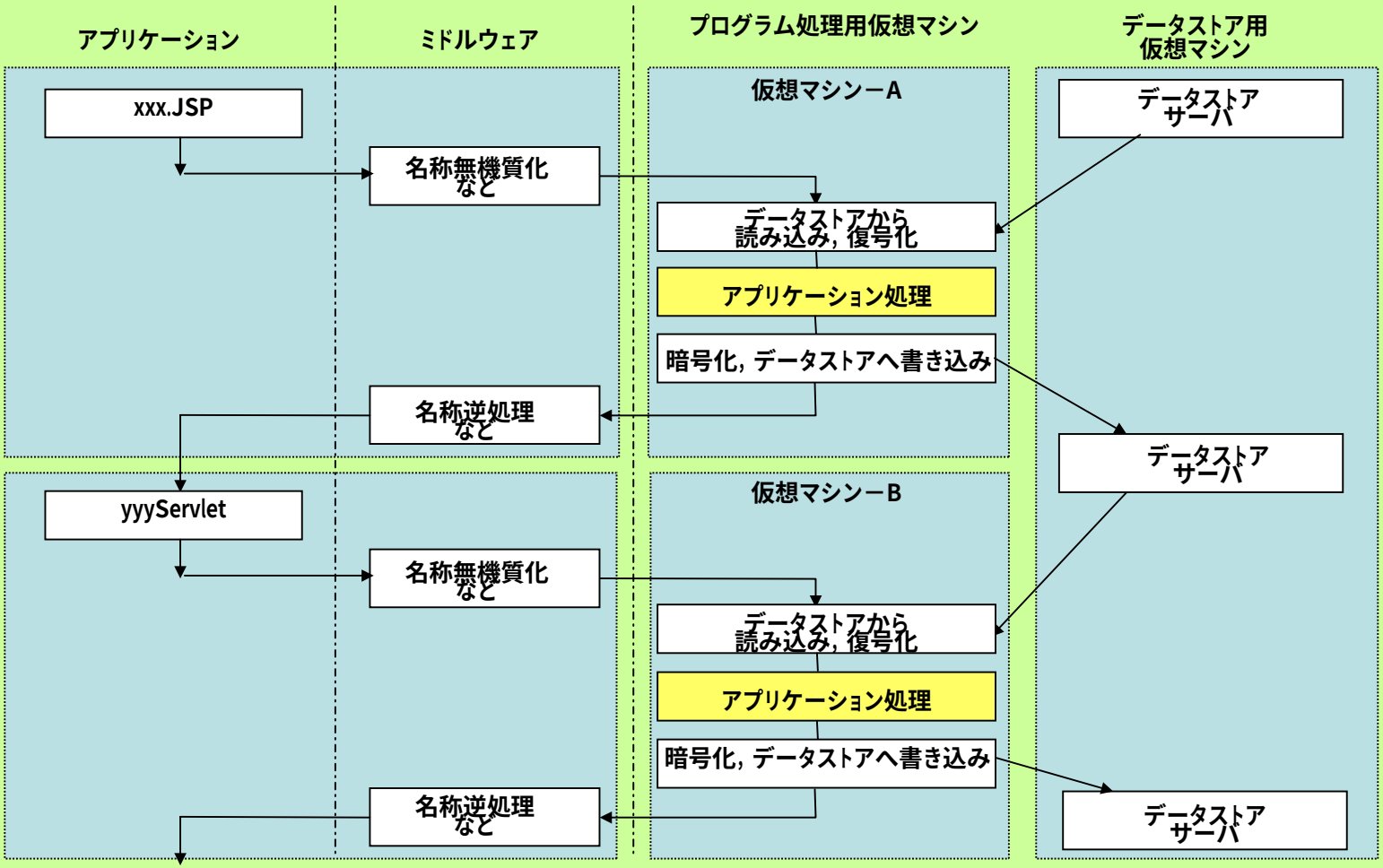


図6 プログラムレベル分割とデータストア

3．実現の方法(11)

- 3.4. 結果の合成
- プログラムレベル分割では，処理結果はデータストアに残される．オブジェクトレベル分割やマシン命令レベル分割では，**計算結果を引き継ぎ合成する**ことが必要であり，この場合次の2つの方式が考えられる．3.1.で述べた状態のセイブとリストアとも共通してくるが，その点については，今後検討し，整理を行う予定である．
- ①ハブ方式
- 部分処理結果を逐一、ユーザ側に返し、ユーザ側から次段の仮想マシンに引き渡す方式（**ハブ方式**と称することにする）である．

3．実現の方法(12)

- ②リング方式
- ①のように，逐一ユーザ側に返さずに，仮想マシンから仮想マシンへと処理の中間結果を引き渡していく方式（**リング方式**と称することにする）が考えられる．
- 両者とも，同様の効果が得られると考えられるが，システムにトラブルが発生したときの対応（後段処理の再分配配置など）としては，ハブ方式が有利と考えられる．処理性能はリング方式が有利と考えられる．
- 図7に両方式を示す．

3．実現の方法(13)

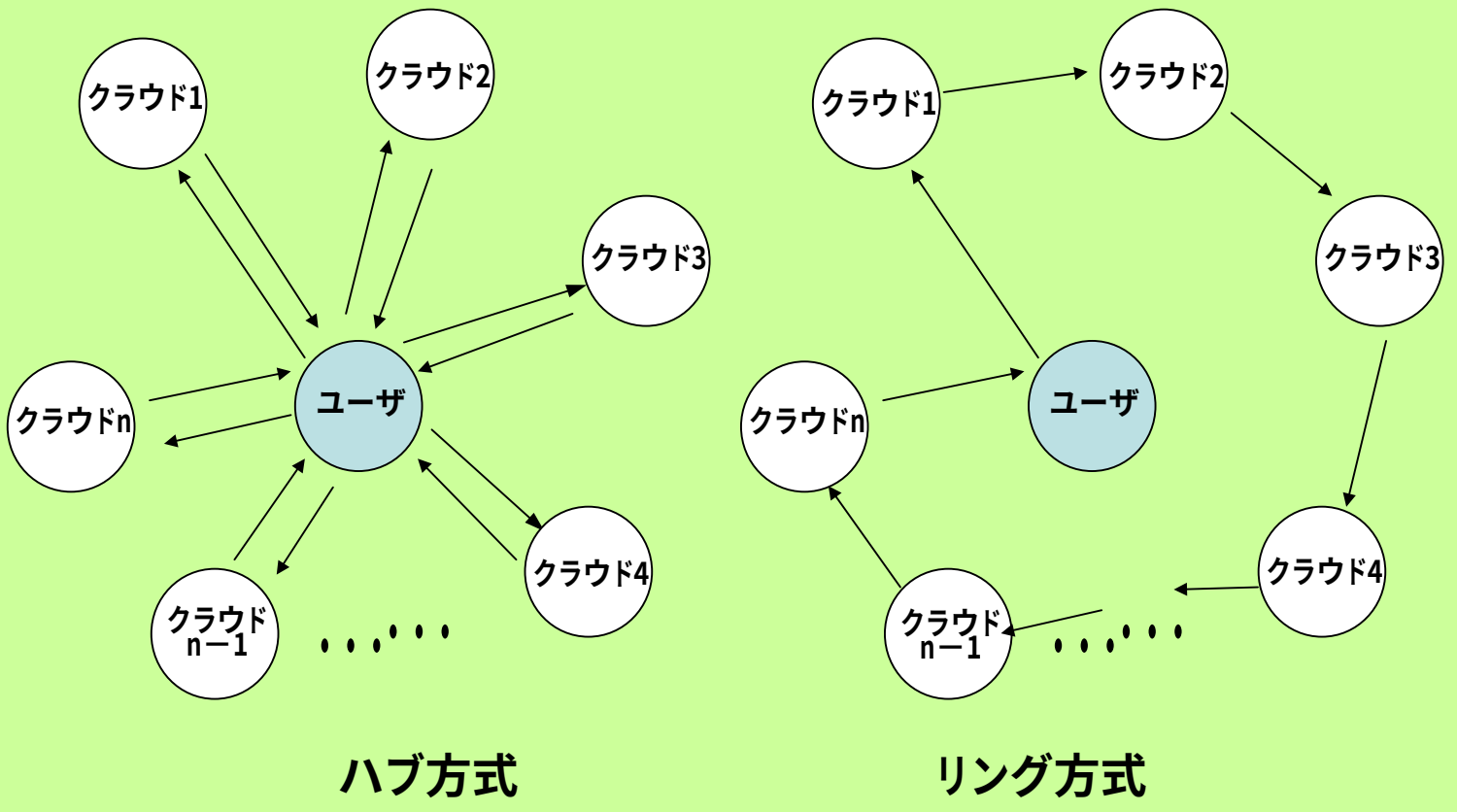


図7　結果の合成方式

3．実現の方法(14)

- 図7において，「一連の処理」の最終結果は，クラウドnにおいて得られる．そこで，クラウドnに対して，処理内容が秘匿されているのかという疑問をもたれるかもしれない．
- しかし，任意のクラウドで処理されるのは「一連の処理」の一部分であり，部分的な処理については，盗み見することは可能であるが，「一連の処理」の全体については，盗み見することはできない．それが，最終的な処理なのかどうかもクラウドにとってはわからない(特にハブ方式において，リング方式では，ユーザに返すことにより，最終的な処理であることはわかるかもしれないが)．
- すなわち，各クラウドは，あるデータについて，自分に割り当てられた処理を行い，その結果を，定められた相手に返すこと，及び暗号化されたデータストア専用のクラウドに中間結果を保存することのみである．

3．実現の方法(15)

- さらには，**オブジェクト名称**や**変数名称**を**「無機質化」**することにより，秘匿性を高めている．これは処理を担当するクラウドにとっては名称が暗号化されていることと同等であり，この場合の名称の復号は，暗号鍵なしで元の名称を推量することに相当し，ほぼ不可能である． 無機質化するほう(図7でのユーザ)は，簡単な**対応テーブル**で行うことができる．そのテーブルはどこにも送出しない．
- 処理を担当しているクラウドとしては，名称そのものの復号化はあきらめざるを得ず，せいぜい部分的な情報処理内容から，どのようなオブジェクトか，どのような属性かを類推するしかできない．そしてそれも全体の一部分である．

4．今後の計画(1)

- 今後の作業は，机上検討では，より深く本提案を検討し，設計することと，そして本提案を実証するために，実験的なクラウドサービス提供者を構築することが必要であり，Hadoop[4]などのオープンソフトによる擬似的なクラウドサービス提供者を構築し，それを用いて，本提案によるソフトウェア群を実証したい．
- 次の順序で実証を進めたいと考えている．
- ①Hadoopによる擬似的なクラウドサービス提供者を3システム以上を設定する．(1つはデータ保管用，他の2つは，データ処理の分散配置用)
- ②小さなアプリケーションシステムを構築し，プログラムレベル分割を実証する．比較的容易と考えられる．名称の無機質化は，手動で行う．データ，プログラムともに暗号化を行わない．

4．今後の計画(2)

- この実験の目的は，異なるサーバで一連の処理が完結するか否かの基本的な機能を確認することである．
- ③続いて，同様に，オブジェクトレベル分割を実証する．
- ④続いて，同様に，マシン命令レベル分割を実証する．この実験のためには，グループウェアの基本的な機能（セイブ，リストア機能）を備えることが必要である．

- 以上の実験で基本的な機能の確認ができる．その次の段階の実験としては，ミドルウェアについての設計，試作，確認を考えている．

4．今後の計画（3）

- 最後に，本稿の査読プロセスにおいて，2名の査読者から有益なコメントをいただいたことに感謝します．コメントについては，できるかぎりの改訂を行ったが，ハブ方式とリング方式の比較など，今後の検討と実証に待つところもあり，すべてのコメントに現時点では対応することはできなかったことをお詫びする．

参考文献

- [1] 松本正雄編著“Webサービス時代の経営情報技術，”電子情報通信学会2009年2月
- [2] 宮西洋太郎，“クラウドコンピューティングとどう向き合うか～クラウドはビジネス系情報システム構築技法の革命となりうるか～，”電子情報通信学会研報，Vol.110，No.70 pp.1-6，June 2010
- [3] CORBA http://www.corba.org/
- [4] Hadoop http://hadoop.apache.org/