

副題： JAWSのCFP内の  
「エージェントの応用」に  
Gridは含まれるけどクラウドが  
ないのはなんでだろう？

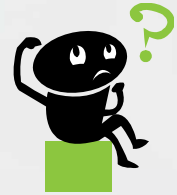
# クラウドコンピューティングにおける エージェントアプローチ

国立情報学研究所  
石川 冬樹

## はじめに

■ 「クラウド (cloud) 」はホットトピック？

➡ JAWSコミュニティにとっては？？？



## はじめに

■ 「クラウド (cloud)」はホットトピック？



➡ JAWSコミュニティにとっては???

- CFPの応用領域諸々に含まれず (グリッドはある!)
- 今年タイトル内に「クラウド」がある論文は1本  
(2010~2008年なし, crowdは1件)
- 2008年浦本さん (IBM) ご講演  
(AAMASも同様)

## はじめに

■ 「クラウド (cloud) 」はホットトピック？ 

➡ JAWSコミュニティにとっては？？？

■ CFPの応用領域諸々に含まれず (グリッドはある！)

■ 今年タイトル内に「クラウド」がある論文は1本  
(2010～2008年なし, crowdは1件)

■ 2008年浦本さん (IBM) ご講演  
(AAMASも同様)

➡ あんまり関係ない？

■ 関係性がない？

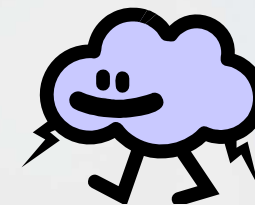
■ 道具として使うが研究タイトルになんて入らない？

■ そんな本質のない流行バズワードけしからん？

# 目次

- クラウドとは
- 道具としてのクラウド
- クラウド（とグリッド）とエージェント
- おわりに

# クラウド：定義



- 「計算資源へのアクセス」のためのモデル
  - *On-demand self-service*：提供者との人手を介したやりとりなく，自動で取得可能である
  - *Broad network access*：様々なプラットフォームから標準的な方法で利用可能である
  - *Resource pooling*：詳細が隠蔽された形でプールされ，複数の利用者に提供される
  - *Rapid elasticity*：迅速に，伸縮可能に提供されており，無限に見えるものから必要な分だけ取得する
  - *Measured Service*：抽象的な指標での測定に基づき，利用が可視化された形で制御，最適化される

[The NIST Definition of Cloud Computing, 2011年9月確定版]

# クラウド：サービスモデル

## ■ SaaS (Software-as-a-Service)

- 構成済みのパッケージソフトウェア
- Gmail, Office 365, salesforce等



利用

## ■ PaaS (Platform-as-a-Service)

- 特定用途・特定プログラミング言語向けの、自動管理ミドルウェアを含むプラットフォーム
- Force.com, Google App Engine, Windows Azure等



計算資源提供



force.com

## ■ IaaS (Infrastructure-as-a-Service)

- 仮想マシンやストレージを自由に（自身で中身を設定して）使うための基盤
- Amazon EC2, Nifty Cloud等



# クラウド：代表的なストーリー



- *量の増減に迅速に（自動）対応でき、事前の見積・準備（投資）を行う必要がない*
  - Animoto（2008）：Facebookユーザへの公開時（3日間で激増）にサーバを50台から4000台にして対応
  - White House（2009）：2日間のアンケート運営に対応（3600万の回答，1秒あたり最大600クエリ）
- *必要なときに使い始め，必要なだけ使い・払い，必要なくなったらやめることができる*
  - New York Times（2007）：新聞を100年分PDF化するため，100台のサーバを24時間利用（10万円強）
  - 定額給付金（2009）：SaaSを用い「使い捨て」のシステムを迅速・安価に構築



# メモ： Scale Upと Scale Out

*大量の処理（リクエストなど）をさばくには？*

- Scale Up： サーバを高スペックにする
  - 古典的なオンプレミスサーバ
  - Webスケールのデータになると、もう扱えない
- Scale Out： 管理ソフトウェアツールとともに、（普通の）サーバを大量に用いる
  - 落ちているサーバが常にあるという仮定に基づき、効率的な自動運用  
（AmazonやGoogleレベルの規模の場合）
  - データや機能について、並行実行・障害復帰しやすい特定の形式を用いる  
（例：key-value DBやMap Reduce）

## クラウド：他の特長（スケールメリット）

- リソースプールを一括で購入・（自動）運用
- ➡ 個別に行うより購入・運用ともに低コスト
  - 自分たちで（プライベート, コミュニティ）
  - 外部の専用業者が（パブリック）

参考：Amazon EC2（計算資源）でのデフォルト価格

- 仮想マシンタイプSmallでLinux利用：\$0.085/hour  
（1.0-1.2GHz Opteron/Xeon程度, メモリ1.7GB）

➡  $\times 24h \times 30days = \$61.2/month$

（実際は「勤務時間内」など必要なときのみ）

- EC2からのダウンロード10TBまで：\$0.12/GB
- ストレージ：\$0.1/GB, I/O 1Mリクエスト：\$0.1/GB

[\[http://aws.amazon.com/jp/ec2/\]](http://aws.amazon.com/jp/ec2/)

# クラウド：他の特長（スケールメリット）

「多く買うほど安い」

	1,000 servers	50,000 servers
Network (per 1M/sec)	\$95	\$13
Storage (per 1G)	\$2.2	\$0.4
Management (per 1 supervisor)	140 servers	1000 servers

*[Above the clouds: A berkeley view of cloud computing, 2009]*

- Amazonのクラウドへのトラフィックは、  
「本」業のものより多くなっている
  - 「本」業の基盤のコスト減少にもますます効く

# クラウド：他の特長（Programmable）

## ■ Programmable：

- プログラムから「Webサービス」としてネットワーク越しに呼び出し可能  
(人間がWebインターフェースなどから利用するだけでなく)

➡ 自身の要求に合った制御を，自動化された形で実現することができる

- アプリケーションとして
- フレームワーク・ライブラリとして

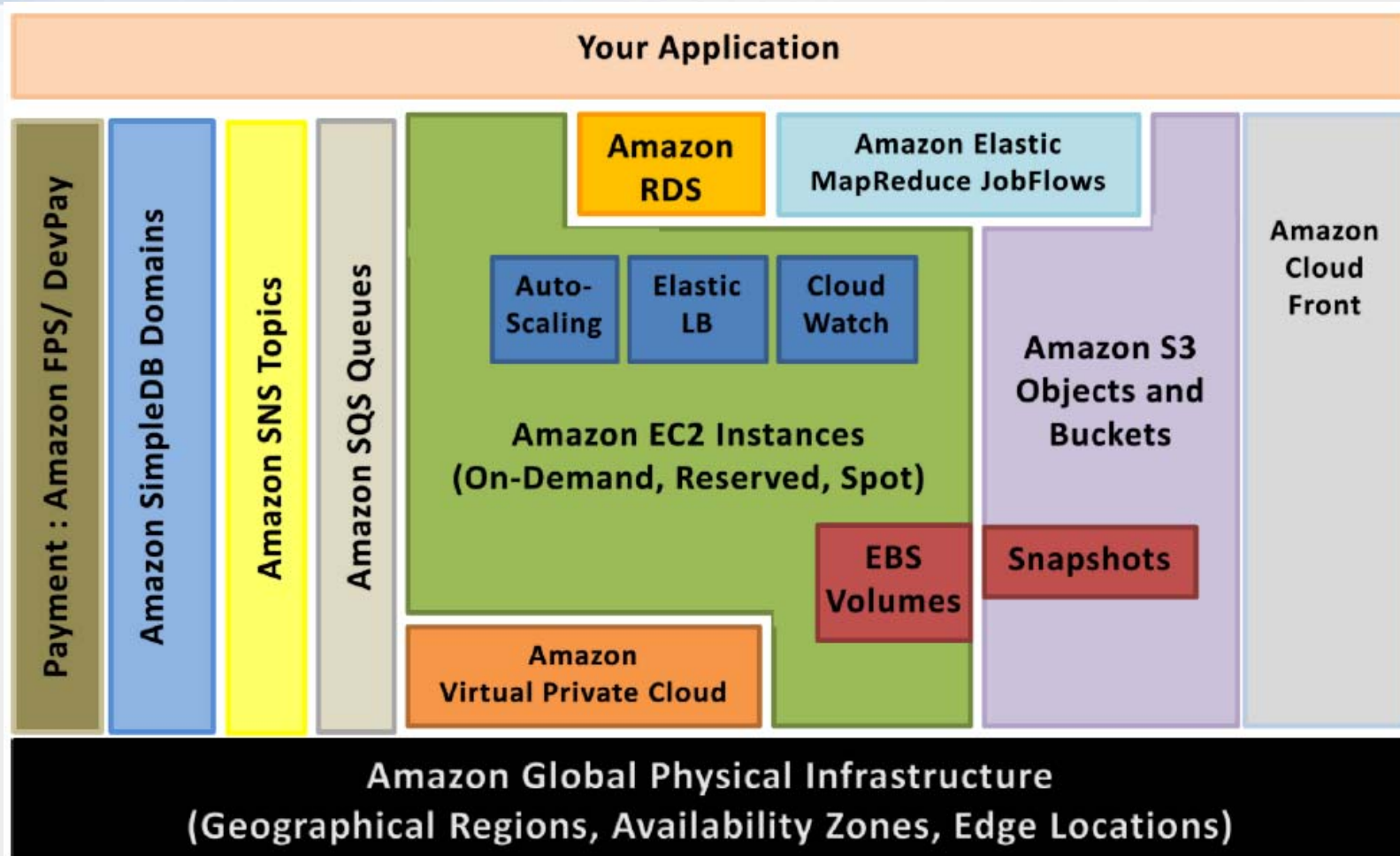
# 目次

- クラウドとは
- 道具としてのクラウド
- クラウド（とグリッド）とエージェント
- おわりに

# エージェントをクラウドで走らせる？

- 大規模シミュレーションなどニーズはある？
  - 大量の構造・非構造データの保持
    - Relational DB, Key-Value DBなど
  - 大量の処理を実行する個々の計算資源
    - CPU/Memory  
(IaaS上での仮想マシンの実行,  
または, PaaS上でのプログラムの実行)
- 大量の処理を分散並行実行するアーキテクチャ実現のための仕組み
  - Queue, Publish-Subscribe
  - Auto Scaling, Load Balancing
  - Map Reduce (Hadoop)

# 例：Amazon Web Services全体像



[J. Varia, *Architecting for The Cloud: Best Practices*, 2010]

# 例：Amazon EC2 (Web管理画面)

**AWS Management Console** > **Amazon Elastic Compute Cloud (EC2)** Fuyuki Ishikawa | Help

**Navigation**

Region: US East (Virginia)

- EC2 Dashboard
- INSTANCES
  - Instances
  - Spot Requests
  - Reserved Instances
- IMAGES
  - AMIs
  - Bundle Tasks
- ELASTIC BLOCK STORE
  - Volumes
  - Snapshots
- NETWORK & SECURITY
  - Security Groups
  - Elastic IPs
  - Placement Groups
  - Load Balancers
  - Key Pairs

**My Instances**

Launch Instance Instance Actions Show/Hide Refresh Help

Viewing: All Instances All Instance Types Search 1 to 10 of 10 Instances

	Name	Instance	AMI ID	Root Device	Type	Status	Security Groups	Key Pair Name	Monitoring	Virtu
<input type="checkbox"/>	empty	i-a839f7c8	ami-9725e6fe	ebs	t1.micro	stopped	Group1	fyukey	basic	parav
<input type="checkbox"/>	empty	i-14a66f74	ami-472ae92e	ebs	t1.micro	stopped	Group1		basic	parav
<input type="checkbox"/>	empty	i-16a66f76	ami-472ae92e	ebs	t1.micro	stopped	Group1		basic	parav
<input type="checkbox"/>	empty	i-a8d31ac8	ami-472ae92e	ebs	t1.micro	stopped	Group1		basic	parav
<input type="checkbox"/>	empty	i-aad31aca	ami-472ae92e	ebs	t1.micro	stopped	Group1		basic	parav
<input type="checkbox"/>	empty	i-aed31ace	ami-472ae92e	ebs	t1.micro	stop			basic	parav
<input type="checkbox"/>	empty	i-b0d31ad0	ami-472ae92e	ebs	t1.micro	stop			basic	parav
<input type="checkbox"/>	empty	i-b2d31ad2	ami-472ae92e	ebs	t1.micro	stop			basic	parav
<input type="checkbox"/>	empty	i-b4d31ad4	ami-472ae92e	ebs	t1.micro	stop			basic	parav
<input checked="" type="checkbox"/>	empty	i-b6d31ad6	ami-472ae92e	ebs	t1.micro	stop			basic	parav

1 EC2 Instance selected

**EC2 Instance: i-b6d31ad6**

Description Monitoring Tags

**AMI:** Loading ami-472ae92e... **Zone:**

**Security Groups:** Group1 **Type:**

**Status:** stopped **Owner:**

**Instance Management**

- Connect
- Get System Log
- Create Image (EBS AMI)
- Add/Edit Tags
- Change Security Groups
- Change Source / Dest Check
- Launch More Like This
- Disassociate IP Address
- Change Termination Protection
- View/Change User Data
- Change Instance Type
- Change Shutdown Behavior

**Instance Lifecycle**

- Terminate
- Reboot
- Stop
- Start

**CloudWatch Monitoring**

- Enable Detailed Monitoring
- Disable Detailed Monitoring

© 2008 - 2011, Amazon Web Services LLC or its affiliates. All rights reserved. | [Feedback](#) | [Support](#) | [Privacy](#) | [AWS.com](#) company



## 例：Amazon EC2（API利用）

```
AmazonEC2 ec2 = new AmazonEC2Client(...);
```

認証情報を与え  
スタブ作成

```
RunInstancesRequest req  
    = new RunInstancesRequest(amiID, minNum, maxNum);  
req.setInstanceType(instanceType);  
req.withSecurityGroupIds(securityGroupID);  
RunInstancesResult res = ec2.runInstances(req);
```

各種の設定とともにインスタンス起動

- イメージID
- 立ち上げ個数（最小・最大）：同時立ち上げ可能な数の制限と照らし合わせて実際の個数が決定
- インスタンスタイプ（CPU/Memoryのサイズを表す）
- セキュリティグループ（ファイアウォール設定を指定）

[\[http://aws.amazon.com/jp/documentation/ec2/\]](http://aws.amazon.com/jp/documentation/ec2/)

## 例：Amazon EC2（API利用）

```
AmazonEC2 ec2 = new AmazonEC2Client(...);
```

認証情報を与え  
スタブ作成

```
DescribeInstancesResult res  
= ec2.describeInstances(  
    new DescribeInstancesRequest());
```

現在起動中のインスタンス情報の取得  
(例：立ち上げたインスタンスの起動処理  
が終わり利用可能になっているか  
チェックするため)

```
ec2.stopInstances(new StopInstancesRequest(instanceIDs));
```

インスタンス停止

[\[http://aws.amazon.com/jp/documentation/ec2/\]](http://aws.amazon.com/jp/documentation/ec2/)

## 例： Amazon EC2（計算資源サービス）

### ■ 様々な資源量・価格の選択肢

- 最大でCPU 3GHz強相当×8コア，メモリ68.4GB
- HPC向けや，GPUも

### ■ 異なる利用・価格形態の選択肢

- On-Demand： その場その場で
- Reserved： 割引付きの長期契約
- Spot： さらに安い，資源余剰時の入札式利用

## 例： Amazon EC2（計算資源サービス）

### ■ 様々な資源量・価格の選択肢

- 最大でCPU 3GHz強相当×8コア，メモリ68.4GB
- HPC向けや，GPUも

### ■ 異なる利用・価格形態の選択肢

- On-Demand： その場その場で
- Reserved： 割引付きの長期契約
- Spot： さらに安い，資源余剰時の入札式利用

- 需要と供給に応じ（？）定期的にAmazon側で価格設定
- 利用者は時間あたりの最大支払可能価格を設定

⚡ [ 現在の価格がそれより下なら割り当て，起動される  
[ 実行中（処理終了前）に価格が上回ったら停止される

[\[http://aws.amazon.com/jp/ec2/spot-instances/\]](http://aws.amazon.com/jp/ec2/spot-instances/)

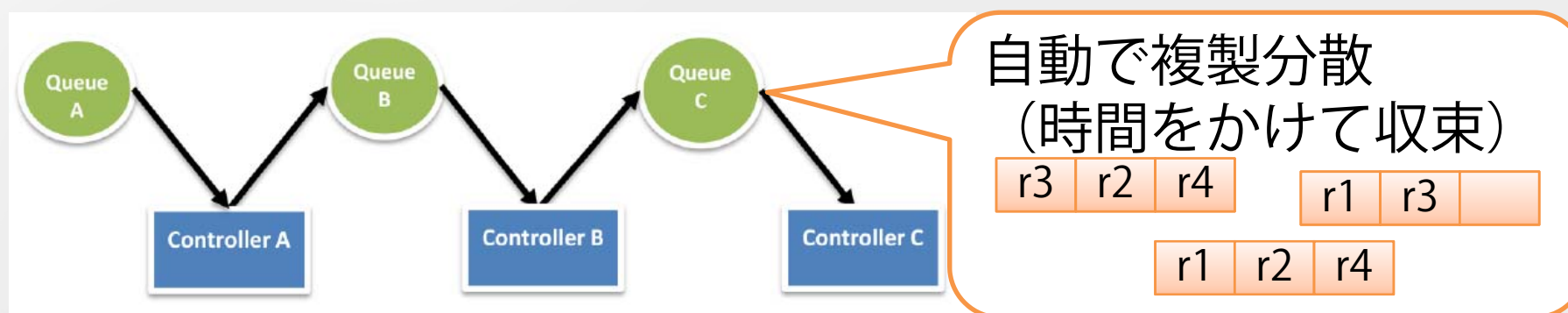
# メモ：クラウド上でのアーキテクチャ

Amazonのドキュメントを見てみると・・・

## ■ コンポーネントの分割

■ 各コンポーネントの失敗や遅れが互いに影響しないように疎結合を

➡ 特にバッチ処理の場合など、水平（同機能の）スケールのためには非同期アーキテクチャに



➡ 複製による並行処理・耐故障化を容易に

[J. Varia, *Architecting for The Cloud: Best Practices*, 2010]

# 目次

- クラウドとは
- 道具としてのクラウド
- クラウド（とグリッド）とエージェント
- おわりに

# クラウドとグリッド

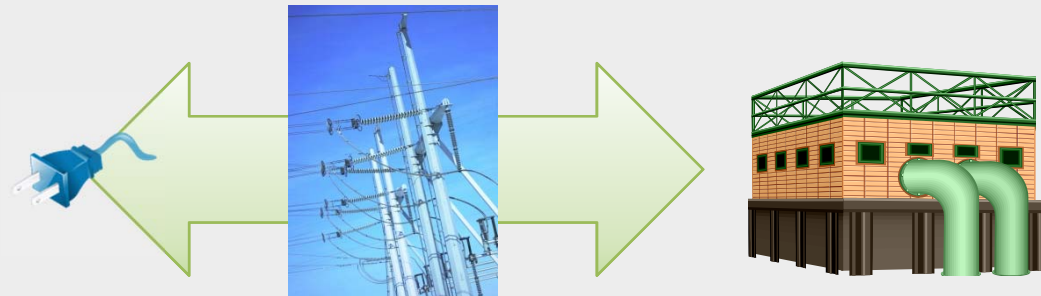
再：クラウドはJAWSやAAMASにおけるCFPの  
応用領域諸々に含まれず  
(グリッドはある！)

## ■ グリッドのそもそものコンセプト

grid： (電気・ガスなどの) 配管網



提供元やその場所,  
実現手段は気にせず  
ただ「つなぎ, 使う」



➡クラウドも満たす

## クラウドとグリッド（続）

### ■ クラウドから見たグリッド

- よく言われること：「クラウドは既存技術組合せ」
- 当然これまでのグリッドでの取り組みも活用

### ■ グリッドから見たクラウド

（アメリカでのFutureGridプロジェクトなど）



- 仮想化とスケーラビリティに関する技術の恩恵
- 今のグリッドプロジェクトは  
「分散・グリッド・クラウドコンピューティングの  
統合的な活用基盤を！」
- グリッドプロジェクトは、サービス運営よりも、  
サイエンス（大規模科学技術計算）指向ではある



# クラウドとグリッド (続)

Ian Foster いわく

Cloud, Grid, and Services can make us smarter

- **Services** make distributed resources and capabilities accessible over the network
- **Grid** assists with integration via standardized service interfaces and collective VO services
- **Cloud** provides for scalable hosting of collective services

# クラウドとグリッド (続)

Ian Foster いわく

Cloud, Grid, and Services can make us smarter

- **Services** make distributed resources and capabilities accessible over the network
- **Grid** assists with integration via standardized service interfaces and collective **VO** services
- **Cloud** provides for scalable hosting of collective services

## クラウドとグリッド（続）

- （元々の）思想・スタンス
  - グリッド：「大学連合」などVirtual Organization (VO) として、資源やサービスを共有・融通しよう
  - ➡ 個々の組織のユーザ優先度や連携インセンティブ、組織間のポリシーすり合わせや合意形成、公平性、Social Welfare, . . .

## クラウドとグリッド（続）

- （元々の）思想・スタンス
  - グリッド：「大学連合」などVirtual Organization (VO) として、資源やサービスを共有・融通しよう
  - ➡ 個々の組織のユーザ優先度や連携インセンティブ、組織間のポリシーすり合わせや合意形成、公平性、Social Welfare, . . . エージェントっぽい？

## クラウドとグリッド（続）

### ■（元々の）思想・スタンス

- グリッド：「大学連合」などVirtual Organization (VO) として、資源やサービスを共有・融通しよう
- ➡ 個々の組織のユーザ優先度や連携インセンティブ、組織間のポリシーすり合わせや合意形成、公平性、Social Welfare, . . . エージェントっぽい？
- クラウド：大量の資源を集中させ、一組織にてスケールメリットを活かし購入・運用の効率向上
- ➡ 資源は「見かけ無限」に十分なほどあり、提供者は組織内で画一的に効率化し、利用者同士は競合せず各自の希望をそのまま提供者に実現してもらう

## クラウドとグリッド（続）

### ■（元々の）思想・スタンス

- グリッド：「大学連合」などVirtual Organization (VO) として，資源やサービスを共有・融通しよう
- ➡ 個々の組織のユーザ優先度や連携インセンティブ，組織間のポリシーすり合わせや合意形成，公平性，Social Welfare，・・・
- クラウド：大量の資源を集中させ，一組織にてスケールメリットを活かし購入・運用の効率向上
- ➡ 資源は「見かけ無限」に十分なほどあり，提供者は組織内で画一的に効率化し，利用者同士は競合せず各自の希望をそのまま提供者に実現してもらう

エージェントっぽい？

効率的だがエージェント研究者には全く面白くない？

## クラウドとグリッド（続）

### ■（元々の）思想・スタンス

- グリッド：「大学連合」などVirtual Organization (VO) として、資源やサービスを共有・融通しよう

➡ 個々の組織のユーザ優先度や連携インセンティブ、組織間のポリシーすり合わせや合意形成、公平性、Social Welfare, . . .

エージェントっぽい？

- クラウド：大量の資源を集中させ、一組織にてスケールメリットを活かし購入・運用の効率向上

➡ 資源は「見かけ無限」に十分なほどあり、提供者は組織内で画一的に効率化し、利用者同士は競合せず各自の希望をそのまま提供者に実現してもらう

そんなことAmazonやGoogleなどにしかできない！

でも全て委ねたくない（遅延、セキュリティ、政治、などなど）

## クラウドとグリッド (続)

### ■ (元々の) 思想・スタンス

- グリッド: 「大学連合」などVirtual Organization (VO) として, 資源やサービスを共有・融通しよう

→ 個々の組織のユーザ優先度や連携インセンティブ, 組織間のポリシーすり合わせや合意形成, 公平性, Social Welfare, . . .

エージェントっぽい?

- クラウド: 大量の資源を集中させ, 一組織にてスケールメリットを活かし購入・運用の効率向上

→ 資源は「見かけ無限」に十分なほどあり, 提供者は組織内で画一的に効率化し, 利用者同士は競合せず各自の希望をそのまま提供者に実現してもらう

グリッドに近い実現模索

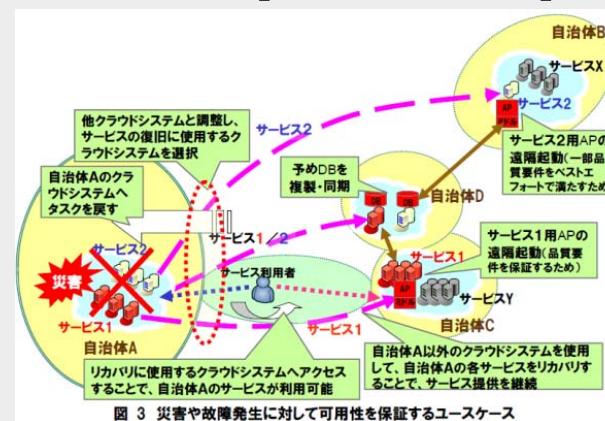
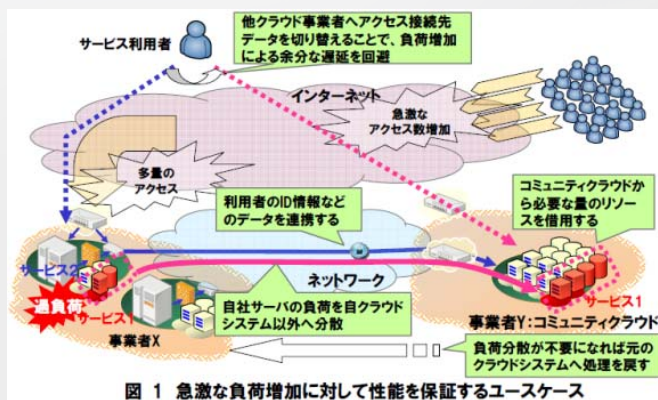
そんなことAmazonやGoogleなどにしかできない!


でも全て委ねたくない (遅延, セキュリティ, 政治, などなど)



# インタークラウド

- 必要性・ユースケースは検討されている
  - 課題Top 10のうち3つに対応：可用性確保，ログイン回避，通信コストの問題対処 [Berkeley'09]
  - グローバルクラウドマーケット [Buyya, FGCS'09]
  - フォーラムでのユースケース列挙 [GICTF 2010]



- 特にIaaSレベルでは互換性が高い  Apache Libcloud
  - Amazon互換でオープンソース・自前

## ・・・とエージェント

### ■ 技術的にはかなり進んでいるはずでは？

#### ■ 一例：AAMAS 2010論文

「価格と解約ペナルティ」について交渉

➡ *Amazon EC2の固定価格モデルよりよい！*

#### ■ 需要と供給が動的・不確かな際に、よりSocial Welfareが向上

*[Automated Negotiation with Decommitment for  
Dynamic Resource Allocation in Cloud Computing, 2010]*

## ・・・とエージェント

### ■ 技術的にはかなり進んでいるはずでは？

#### ■ 一例：AAMAS 2010論文

「価格と解約ペナルティ」について交渉

➡ *Amazon EC2の固定価格モデルよりよい！*

■ 需要と供給が動的・不確かな際に、よりSocial Welfareが向上

*エージェント研究者には当然？すごいことなのでは？*

*[Automated Negotiation with Decommitment for  
Dynamic Resource Allocation in Cloud Computing, 2010]*

## ・・・とエージェント

### ■ 技術的にはかなり進んでいるはずでは？

#### ■ 一例：AAMAS 2010論文

「価格と解約ペナルティ」について交渉

➡ *Amazon EC2の固定価格モデルよりよい！*

#### ■ 需要と供給が動的・不確かな際に、よりSocial Welfareが向上

(注：もちろんオークションアルゴリズムとも比較)

*エージェント研究者には当然？すごいことなのでは？*

#### ■ 公平性, 嘘の防止, 協カインセンティブの確保, などなどのアルゴリズム・メカニズム

*[Automated Negotiation with Decommitment for  
Dynamic Resource Allocation in Cloud Computing, 2010]*

## エージェントの活躍・実証の場？

- 「うまく」自動で（人と相互作用しつつ）
- 「財」に対するコスト投資を「うまく」
- インセンティブや社会利益の確保を「うまく」

## エージェントの活躍・実証の場？

- 「うまく」自動で（人と相互作用しつつ）



お金を預けて自動で壺を買わせる状況って来る？

- 「財」に対するコスト投資を「うまく」



質と価格のバランス？ベストエフォートで無料なんだけど？

- インセンティブや社会利益の確保を「うまく」



組織内に閉じて画一的・単純にやっちゃうんだけど？

# エージェントの活躍・実証の場？

## ■ 「うまく」自動で（人と相互作用しつつ）



お金を預けて自動で壺を買わせる状況って来る？

→ クラウドでは、課金を伴うこと（購買そのもの、価格決定）まで当然のように自動化されている

## ■ 「財」に対するコスト投資を「うまく」



質と価格のバランス？ベストエフォートで無料なんだけど？

→ クラウドでは、質や量の確保に対して、契約があり、応じた料金を支払う仕組みになっている

## ■ インセンティブや社会利益の確保を「うまく」



組織内に閉じて画一的・単純にやっちゃうんだけど？

→ クラウドでは「（所有せず）他組織のものを使用」が基本で、インタークラウドでは提供者側も組織間

# 目次

- クラウドとは
- 道具としてのクラウド
- クラウド（とグリッド）とエージェント
- おわりに



# おわりに

## ■ 個人的「感想」

- 「クラウド」という言葉がほとんど出てこないのは、少し寂しい（JAWSらしい？）
  - 自分はクラウド運営側にも近いので
- きっと道具としても活用できる
- エージェントが活躍できる場所はクラウドに限らず元々多いはず
  - 特定分野の適用は、新たな学術的課題を産み出し「研究」になるかどうか？
  - 現実味を持たせる・追求する道具の一つ？