WG5: 「ユーザ企業からの視点」のポジションペーパ

SS2010　June 9-11, 2010　横浜開港記念会館

熊谷章（TAO BEARS）

１．　自己紹介

ミニコンの開発、ワークステーションの開発、Unix systems、知的支援システムの研究開発、Formal methods、分散並列コンピューティング、システムズエンジニアリング、記号論、プロジェクトマネージメント。アメリカ、オーストリア、ロンドン、インド、中国との交流。富士通グループ３３年、東京エレクトロン４年、TAO BEARS３年で現在に至る。

２．　自己主張

コンピュータの変革は激しくここ３０数年楽しい仕事が続いている。しかし、変化の原点は常に社会の要請に基づいている。そして、その根底にあるものはコンピュータのユーザの視点である。あらゆる製品と同様にコンピュータも使用されてはじめてその価値を生み出す。コンピュータのイノベーションとユーザの要求は最も遠い位置にあるようで実は表裏一体をなしている。この視点から、今回はコンピュータの近年のイノベーションを眺めてみて、それがユーザにどのような変化を齎すかを考察してみたい。WG ではここであげたテーマを参加者と一緒に議論を深めてみたい。

最近気になっているコンピュータシステムの変化には次のものがある。

- Cloud computing
- Multi-core Parallel computing
- End –user software engineering
- Domain specific languages
- Architecture as language

各々のテーマに関して議論の論点を拾い上げてみる。

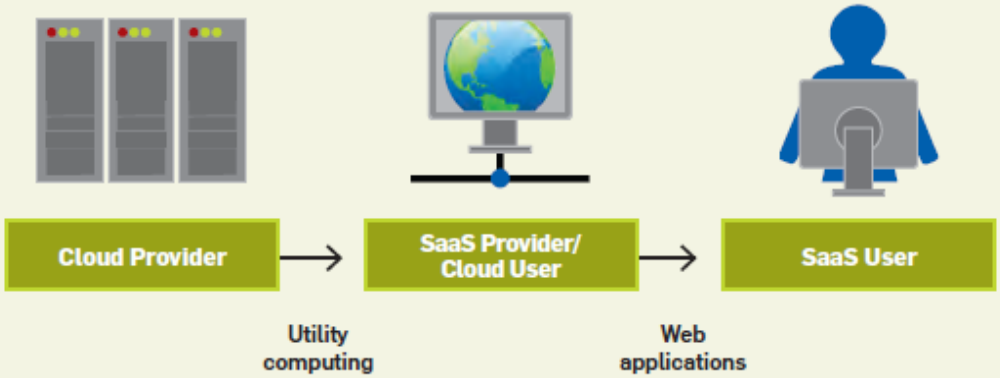
２．１　Cloud computing

Cloud computing の定義は、インターネットを介したサービスとしてのアプリケーションの提供と、データセンターのハードウェアとソフトウェアのサービスの二種類のものを指すと考えられる。この手のサービスは、ずっと前から SaaS（Software as a Service）として行われているものである。ベンダーによっては、このサービスを IaaS（Infrstructure as a Service）と呼んだり、PaaS（Platform as a Service）と命名している。一見、高レベルのサービスと低レベルのサービスの計算能力と補助記憶装置の遠隔地間の共有サービスであるから、Cloud computing とは明らかに異なっている。

一般的に公開されているデータセンターで、使った分だけのハードウェア使用料を支払

うものを「Public cloud」と呼び、これは「Utility computing」とも呼ばれている。一方、自分が属する機関内のそれを「Private cloud」と呼ぶ。利用者は、SaaS の提供者か利用者のいずれかになる。下図にそのモデル図を示す。

Figure 1. Users and providers of cloud computing. We focus on cloud computing's effects on cloud providers and SaaS providers/cloud users. The top level can be recursive, in that SaaS providers can also be a SaaS users via mashups.



ハードウェア提供の視点からみた Cloud computing の 3 つの特徴は次の通り。①必要に応じた制限のない計算能力が提供できること、②Cloud ユーザは事前の契約をすることなしに必要に応じて少ない計算能力から大きな計算能力に動的に変更できること、③必要に応じてコンピューティング資源を短期間ベースで支払うことができ、使用をやめることもできること。この Cloud computing サービスを可能にするためには、そのデータセンターは巨大規模の普通のコンピュータを使用し、価格で５％～７％低価格で提供できることがビジネスのキーポイントになる。どんなアプリケーションでも、計算モデル、データの記憶モデル、それにコミュニケーションモデルが必要である。これらを自動的にスケーリングできるためには、自動的なリソースの配置方式とマネージメントが必要になる。

従って、utility computing のサービスレベルの違いによって、Cloud system のソフトウェアの抽象レベルとリソースの管理方式のレベルが違って来る。例えば、Amazon EC2 は、物理的なハードウェアの Cloud computing であるから、自動的なスケーリングは困難である。なぜなら、Amazon は、アプリケーションの内容を知らないので、自動的には変更できず、すべてユーザが決めなければならない。一方、Google では、Web application に特化した AppEngine を用いた Application domain-specific platform なので、Google がスケーリングが可能で、かつ記憶装置はアプリケーション向けの MegaStore data storage を使用できる。Amazonn と Google の中間に位置しているのが、Microsoft の Azure であり、ここには.NET library と言語に非依存の Common Language Runtime がある。この Framework は、AppEngine に比較すれば遥かに flexibility が高い。

次に Cloid computing の障害とその対策を 10 個の視点から考察してみよう。

Table 2. Top 10 obstacles to and opportunities for growth of cloud computing.

	Obstacle	Opportunity
1	Availability/Business Continuity	Use Multiple Cloud Providers
2	Data Lock-In	Standardize APIs; Compatible SW to enable Surge or Hybird Cloud Computing
3	Data Confidentiality and Auditability	Deploy Encryption, VLANs, Firewalls
4	Data Transfer Bottlenecks	FedExing Disks; Higher BW Switches
5	Performance Unpredictability	Improved VM Support; Flash Memory; Gang Schedule VMs
6	Scalable Storage	Invent Scalable Store
7	Bugs in Large Distributed Systems	Invent Debugger that relies on Distributed VMs
8	Scaling Quickly	Invent Auto-Scaler that relies on ML; Snapshots for Conservation
9	Reputation Fate Sharing	Offer reputation-guarding services like those for email
10	Software Licensing	Pay-for-use licenses

１． Business Continuity and Service Availability

停電などのトラブルで Cloud computing サービスが停止することがある。対策としては複数の Cloud computing サービスを使用すること。

2. Data Lock-In

Cloud computing の Storage API はベンダー依存で標準仕様がないため、Cloud サイト間で自由にデータとプログラムの移動ができない。

3. Data Confidentiality/Auditability

機関によっては機関外に置いたデータの機密の安全性やデータのAuditabilityが要求される。Cloud usersは、データの安全性は機関内外で要求されることになる。この問題はデータセンターで既に経験済である。しかし、Cloudでは、Cloud users、Cloudベンダー、ソフトウェアのベンダーらの多くの人々が関与すし、データセンターよりも複雑化している。

4. Data Transfer Bottlenecks

アプリケーションは極度のデータ中心である。だから、データ転送の時間と費用が実用の課題になる。１テラ当たり１００ドルから１５０ドルし、これが急速に値上がりするのが問題である。一つの解として、Amazonがやっているようにディスクそのものを送ること、またはコンピュータ全体を送る考えがある。一つの例で試算してみよう。

１０TBのデータをU.C.BerkeleyからAmazonシアトルに転送する例

１０*１０１２Bytes/（20×106 bits/second）=(8×1013)/（２×１０７）seconds
=4,000,000seconds → > 45 days

1TBのディスクを10個送る場合、1日で届く。ネットワークの伝送速度を1,500MB/sec以上のサービスにすれば解決できるが、、。

5. Performance Unpredictability

VM（Vertual machines）では、CPU とメモリを共有することはうまく行われているが HDD の共有とネットワークは難しく首尾よくいっていない。従って、異なった EC2 では異なった性能になるのが普通である。例えば、75 EC2 で、STREAM memoey benchmark を計測したら、平均の Bandwidth が 1,355MB/sec なのに、計測値は 52MB/sec で僅かに平均値の 4%だったらしい。同じように、EC２ディスクからローカルディスクへの転送速度を測ったら、平均値は 55MB/sec だが、実測値は９MB/sec で 16%であったという結果がでている。

6. Scalable Storage

Cloud computing の特徴は、①短期間の使用が可能、②事前の契約と支払いが不必要、③必要に応じた無制限のリソースが保障されている、の三点である。記憶装置のスケーラビィティを保証するためには、困難な課題があり新しい工夫が必要である。

7. Bugs in Large-Scale Distributed Systems

大規模の分散並列コンピューティングシステムでの難題は、エラーの修正である。一つの解は、Cloud computing 上の VM の高信頼性である。SaaS 提供者は VM を使ってソフトウェアのバグを抑止し、バグを見付けだすことができる。

8. Scaling Quickly

ネットワークと HDD の場合「使った分だけ払う」の基本単位はバイトである。しかし、CPU の課金は違っていて、バーチャル化の度合いに応じて金額が決まる。Google ではロードサイクルの回数で、AWS ではインスタンスの数が存在している時間数である。金額を少なくするためには、スケーリングをすばやく変更する必要がある。

9. Reputation Fate Sharing

同じ Cloud を使っている他のユーザの悪意のある表評判に対して、良策を講じる必要がある。

10. Software Licensing

現状のソフトウェアライセンスは、CPU 単位や保守契約であり、Cloud computing には合致していない。また、売買契約の金額の 22%前後の年間保守料金をユーザに課するベンダーもあり、コンピュータ利用の大きな制約になっている。Amazon や Microsoft は、ソフトウェアライセンスを「使った分だけ払う（pay-as-you-go）」方式の契約を開始している。EC2 の Windows 使用料は、１時間当たり 0.15 ドルである。IBM もこの方式の契約を提供し始めている。

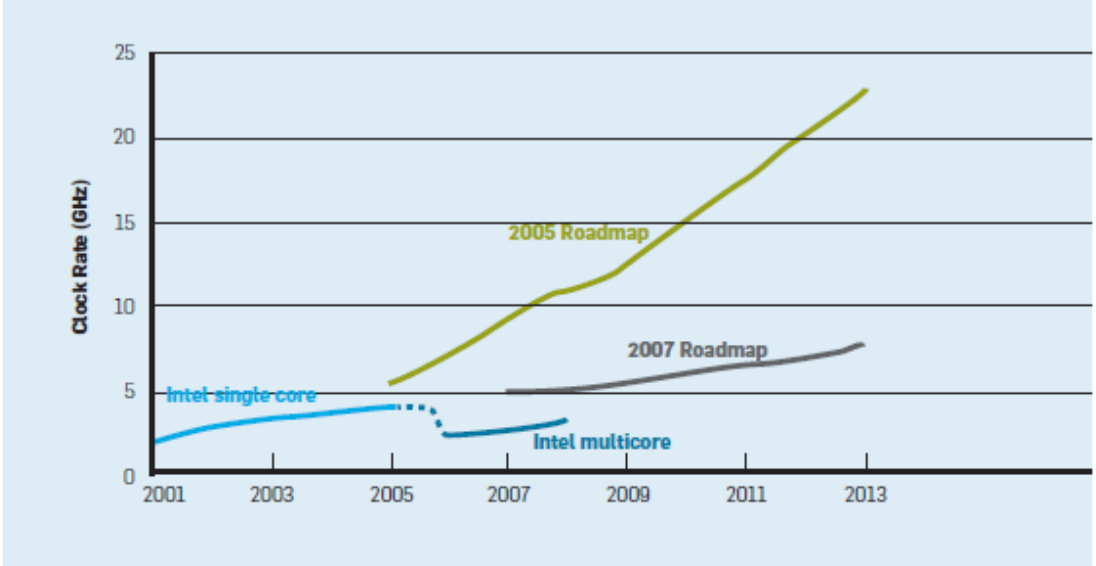
ここであげた課題はいずれも致命的なものでなくビジネス社会で徐々に解決されてゆくと考えられる。従って、肝心なのはこのようなサービスが、コンピュータのユーザ企業の経営とコンピュータシステムにどのような影響を及ぼすかである。中小企業は、自前のコンピュータ設備を構築するよりも、あらゆる面で安全で費用対効果が良くなる可能性が高い。特に、注目されるのは、スムーズなスケーラビリティ、事前契約の不必要性、ライセンス契約形態の変更、システムのデペンダビリティである。新しいアイデアの計画、プロトタイピング（実験)、システムの開発、運用などのシステムのライフサイクルに応じてCloud computing のパワーをうまく使うことが製品の QCD(品質、コスト、提供時期)の改善に繋がる可能性が高い。身近な例で Cloud computing の特徴を実験してみる価値がありそうだ。

Cloud computing の今後を考える視座には次の点がある。一つ目は、アプリケーションソフトウェアは、スケールアップと同様にスケールダウンも考える必要があるということ。そして、使った分だけ（pay-for-use）形態のライセンスが必要になること。二つ目は、Infrastruvture software はもはや裸のハードウェア下で動作することを前提とするのではなく、VM（Virtual Mechines）下で動作することが基本だとするべきこと。三つ目は、ハードウェアシステムを作成するときは最低でも１２個以上のラックを製品単位として設計すべきであるということ。なぜならそれがハードウェア売買の最小単位となるから。

２．２ Multi-core Parallel computing

18 ヶ月毎に二倍の性能というムーアの法則はトランジスタの速度と必要十分な開発費によって成立していたが、2007 年頃から成立しなくなった（次図を参照)。

Figure 1. Microprocessor clock rates of Intel products vs. projects from the International Roadmap for Semiconductors in 2005 and 2007.[16]



そこで性能改善の途は必然的に Multi-core Parallel computing になった。その証左は、スーパーコンピュータの変遷で証明されている。いまや超大型のベクトルコンピューティングであったスーパコンピュータは消滅し、多数のスカラーコンピューティングをネットワークで結合した PC クラスターがスーパーコンピュータの典型となった。そこでの課題は並列コンピューティングである。しかし、並列計算への挑戦は昔から為された。Convex、Encore、Floating Point Systems、Inmos、Kendall Square Research、MasPar、nCUBE、Sequent、Silicon Graphics、Thinking Machines などがその例であるがことごとく失敗した。コンピュータサイエンスの中で、parallel computing は最も困難な挑戦であることが分っている。

並列コンピューティングの関係者を大きく分けるとハードウェア、ソフトウェア、アプリケーションの三者になる。その関係を次図に示す。

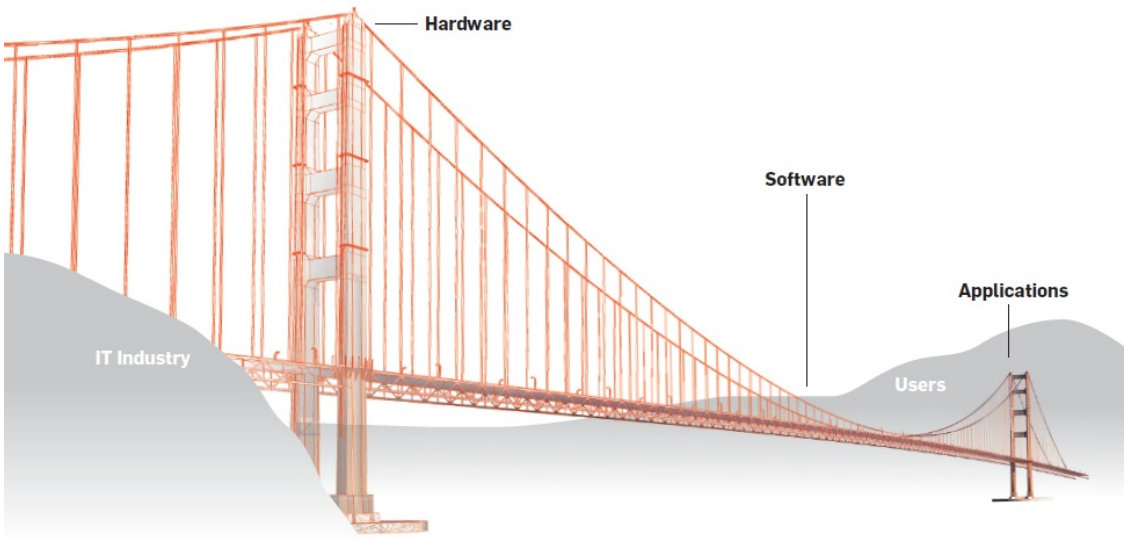


Figure 2. Bridge analogy connecting users to a parallel IT industry, inspired by the view of the Golden Gate Bridge from Berkeley, CA.

IT テクノロジーのハードウェアとユーザのアプリケーションとの間を橋で結ぶのがソフトウェアの役割である。

ここでのソフトウェアの課題に、並列コンピューティングプログラムを書けるプログラマの能力、並列コンピューティング下で動くオペレーティングシステム（OS）とプログラミング言語のコンパイラ、並列コンピューティングで動く並列コンピューティングプログラムの性能を確かめる方法がある。最初はプログラマの問題で、従来のプログラマを再教育してもほとんどの人々が並列コンピューティングプログラムを書けないことである。排他制御のロック処理、ロードバランス、スケジューリング、メモリの無矛盾性などが難し

い点である。従って、一般のプログラマでも並列コンピューティングのプログラムをかける仕組みを考案する必要がある。次は、システムソフトウェアの問題で、具体例がオペレーティングシステム（OS）とコンパイラで、これらは変化に対して柔軟に対応できない状況になっている。新しい製品が市場に定着するには、10 年以上掛かるという評価があるほどである。OS とコンパイラの開発者がいかに迅速に現存システムを進化させることができかかがキーポイントである。最後に並列プログラム言語の評価がある。いままで並列プログラミング言語の特徴と良さを主張している研究や論文は多いが、実際のプログラムを書くプログラマのことを心理学やプログラマの人間性と能力を主とした研究がなかった。今後はこの方面を研究して、現在遭遇している課題を解決することが重要である。

アプリケーションソフトウェアでは、向こう 10 年か 20 年間を見据えて、並列コンピューティングパワーを必要とするキラーアプリケーションを見出すことがキーポイントである。従来のレガシーソフトウェアが動作することを前提条件としつつ、より多量のデータ処理や計算処理を必要とするアプリケーションが重要となる。

２．３　End–user software engineering

End-use computing や Ed-user programming は自然で重要なアプローチであるが、コンピュータシステムの使用が社会システムやお金の処理を含む経理システムに使用されているので、安全性と信頼性をいかにして保証するかというデペンダビリティ（Dependability）が要求されるようになった。この視点から考えれば、システム開発の全ライフサイクルを通して、安全性、信頼性を考慮した End-user software engineering が要求されていると考えられる。システム開発の各段階で、常に有用性（Validation）と検証（Verification）を考慮した開発方法とチェックの仕方が必要である。この視点から、End user 開発を見直す時機になったと考えられる。

この問いに対する一つの解が Unicage 開発方法である。そのアイデアは、データモデルを従来の手続き志向からデータ中心、言い換えればデータフロープログラミングに変えたことである。データ中心を徹底すれば、それを処理するプログラムは使い捨てでよいとう結論になり、使い捨てでよいというプログラムは End user が簡単に使用できる Shell や 4 GL 風のスクリプト言語で十分ということになる。今後、この手の事例出現に努力しかつ注目したい。

２．４　Domain specific languages と Architecture as language

既に紙面は尽きたが、前項の End-user software engineering を進めるためには、ドメインイに依存したプログラミング言語の開発が望まれる。急速なハードウェア技術と生産技術の進歩により、ハードウェア価格は劇的に変化した。また、ソフトウェアの発展のより

フリーソフトウェアの利用などにより、自前のシステム開発が低コストで実現できるようになった。すべてのコンピュータシステムのコンポーネントとして一般製品を使用するのが従来の常識であったが、これからは End-user 特定のコンピュータシステムの実現が可能になり、かつ様々な意味で効率をあげることになると予想できる。簡単な例をあげると、いままで Windows、Unix、Database、ミドルウェア、各種パッケージと SI ベンダーに金縛りにあっていたものから、自前のシステム開発ができるビジネス環境になったと考えられる。

そのときの視点は一般化の視点からアプリケーションシステムを展開するのではなく、End-user のシステムを起点としたシステムズエンジニアリングがまさに重要になったとうことである。この考えを進めると、そこには DSL（Domain specific languages）とそれを支えるアーキテクチャ（Architecture as language）がある。

この点を今回は十分に議論できなかったが次の機会に更に深く議論してみたい。

おわりに

この WG を企画した熊野さんと参加して戴いた坂本さん、杉田さん、緒方さんに感謝したい。今回は全体のプログラムとの関係で、議論する十分な時間がなかったが、WG では多くの示唆があり今後のためになる内容が多かった。どこかで続きをやりましょう。

文献

K.Asanovic, etc., ”A view of the Parallel Computing Landscape”, CACM, October 2009
M.Armbrust, etc., “A view of Cloud Computing”, CACM, April 2010
M.Freudenthal, “Domain-Specific Languages in a Customs Information System”, IEEE Software, March/April 2010
M.Volter, “Architecture as Language”, IEEE Software, March/April 2010
M.Burnett, etc., “END-USER SOFTWARE ENGINEERING”, CACM, September 2004
J.Torrellas, etc., “The Bulk Multicore Architecture for Improved Programmability”, CACM, December 2009