クラウド上のグリッド

木部 真一郎[†], 山際 基[‡], 上原 稔[‡]

クラウド時代には新たな技能が求められ、教育期間はそれを教育する必要ある。教育クラウドはクラウド時代の教育環境であり、学生に自由に使える複数のインスタンスを提供する。我々は教育クラウドを UEC(Ubuntu Enterprise Cloud)に基づくプライベートクラウドとして構築する。我々の教育クラウドの最大の特徴は過飽和にある。過飽和とは、物理資源以上の論理資源を割り当てることである。過飽和クラウドは通常の 10 倍のインスタンスを実行することを許す。性能を犠牲にコストを大幅に低下する。過飽和クラウドのスループットを最大化するには、クラウドにタスクが供給され続けなければならない。一方、遊休資源を活用する PC グリッドは省エネでないため利用が減少している。クラウドとグリッドは相補的な関係にある。グリッドはタスクを供給し、クラウドが消費する。我々は、クラウド上にグリッドを構築することで、このような問題を解決する。本論文では、NC(Node Controller)にグリッドを導入する NG(Node Grid)方式とインスタンスにグリッドを導入する IG(Instance Grid)方式を提案し、その比較評価を行う。その結果、インスタンスがシステム負荷を共有する場合、IG 方式において SLI 戦略(Single Large Instance Strategy)を採用することが最適であると結論した。

Grid on Cloud

Shinichiro Kibe[†], Motoi Yamagiwa[‡], Minoru Uehara[‡]

In cloud era, new cloud skill is required for IT specialists. Educational organizations such as University need to provide educational cloud for students. We are constructing private cloud based on UEC(Ubuntu Enterprise Cloud) as educational cloud. The most important feature of our cloud is super-saturation, which is defined as allocating much more logical resource than physical resource. Super-saturated cloud realizes to run more 10 times instances than conventional cloud. The performance decreases a little but the cost decreases drastically. In order to maximize the performance of super-saturated cloud, we need to continue providing tasks for the cloud. On the other hand, the demand of grid is reduced because of energy saving. Grid and cloud are complementary. Cloud consumes tasks produced by Grid. In this paper, we propose Grid on Cloud(GoC) method in order to solve such an issue. We compare two methods, NC(Node Controller) and NG(Node Grid). As this result, we conclude that SLIS(Single Large Instance Strategy) based IG method is the best if system load is shared by instances.

1. はじめに

近年,クラウドコンピューティングが急速に発展しつつある。また、IPv6 およびモバイル通信の発展は、Internet of Thing の形成を促している。Internet of Thing では、PC だけでなくスマートフォンや様々なセンサーデバイスがインターネットに接続され、その情報がクラウドに集約される。そして、その爆発的に増大する情報はビッグデータと呼ばれ、それをビジネス等に活用するためにデータサイエンスが発展している。このように現代はクラウド時代といえる。

クラウド時代には、従来と異なるスキルが要求される。組み込み技術者はクラウドのサービスを利用するために Web サービスを学ぶ必要がある。ネットワーク管理者は 10-100 倍以上のノードを管理する方法を学ぶ必要がある。IT アーキテクトはクラウドベースのシステムを設計する必要がある。システムエンジニアは短期間でシステムを構築するために複数のサービスをマッシュアップする方法を学ぶ必要がある。IT ストラテジストは継続的イノベーションのために経営に役立つデータをクラウドに集約する方法を学ぶ必要がある。データサイエンティストはクラウドの集約されたデータから経営に役立つ知識を発見する必要がある。

大学はこのような人材を教育するためにカリキュラムや教育環境を変える必要がある。しかし、既存の PC 教室では共有 PC での利用であり、管理者権限が与えられていない。そのため、自由に利用することができない。

我々は、クラウド時代の教育を支援する教育環境として教育用クラウドを提案する。教育用クラウドでは、学生が一人一台専用マシンを持つことができる。専用マシンを使うことで、共用マシンでは不可能だった演習が可能となる。例えば、管理者権限を必要とするシステム管理演習などである。また、一人で複数台の仮想マシン(以降、インスタンスと称す)を持つことができる。これにより、仮想的なネットワークを構築して、サーバ運用や大規模環境の構築など様々な実験を行うことができる。

従来の方法で教育用クラウドを構築するには高いコストが必要となる。そこで、我々はクラウドを軽量化し、クラウドのコストを削減する。我々は、軽量クラウドを実現するために過飽和という概念を導入した。我々は、過飽和を物理資源以上の論理資源を割り当てることと定める。例えば、CPU 資源はコア数として計量される。あるノードのインスタンスのコア数の合計がそのノードの物理的なコア数より大きいとき、そのノードは過飽和状態にあるという。代表的な教育用途はプログラミング演習である。そのような演習では小さなプログラムを短時間動かすことが多い。

[†]東洋大学工学研究科情報システム専攻
Dept. of Open Information Systems, Graduate School of Toyo University
[‡]東洋大学総合情報学部総合情報学科
Dept. of Faculty of Information Sciences and Arts, Toyo University

したがって、教育用途では過飽和状態のノードであっても十分機能する。逆に、ビジネスと同じ様に利用すると大部分の資源が遊休状態となってしまう。過飽和クラウドは従来のクラウドに比べて 10 倍のインスタンスを動かすことができる。言い換えると、コストは 1/10 になる。

教育用クラウドの課題の一つは利用率の低さである。PC 教室は基本的に演習の時しか利用されない。演習時に合わせて設備を導入することはピークに合わせることになり、必然的に利用率は低下する。演習時間を分散すれば若干平均化するが不十分である。また、自習のために開放すれば若干向上するが、同時にピークを押し上げることにもなる。過飽和クラウドによってコストを削減することができるが、性能の劣化を伴うため、HPC のような演習では問題がある。このように教育用クラウドの利用率は必ずしも高くはない。また、これを向上させる運用には限界がある。なお、パブリッククラウドは抜本的な解となるが、その導入には別に問題が生じる。ここでは、プライベートクラウドをピーク性能に合わせて導入した場合の効率的な利用法について検討する。そこで、導入されたクラウド設備を有効に活用するためにグリッドと組み合わせる運用を提案する。

元々クラウドは伸縮性を持つ。それゆえピーク性能に合わせた場合でも従来方式より過剰な設備は必要としない。過飽和方式によりさらに必要な設備は削減できる。しかし、それでも無駄が生じる。この無駄は導入コストだけでなく、電力などの運用コストにも存在する。

利用率に応じてノードを停止すればかなりの省エネとなる。しかし、利用率が低い時にもすべてのノードが停止するわけではないので決して 0 にはならない。ゆえに稼働中のノードは、その利用率を最大化することが望ましい。そうすれば、ノードが複数あるとき、インスタンスを集約して、稼働するノードを減らすこともできる。しかし、実行すべきインスタンスがなければ利用率を向上させることはできない。そこで、外部にタスクを求める。すなわち、インスタンスの代わりにグリッドを動かし、利用率を向上させる。このような方式を、クラウド上のグリッド(Grid on Cloud, GoC)と名付ける。本論文では、GoC の有効な方式について提案し、評価を行う。

本文の構成は以下の通りである。関連研究として 2 章でグリッドと教育用クラウドについて述べる。3 章では GoC 方式を提案する。4 章で評価を行い、最後に結論を述べる。

2. 関連研究

2.1 PC グリッド

グリッドには様々な方式およびシステムがある。我々が対象とするグリッドは遊休資源を活用する PC グリッドである。

代表的な PC グリッドには BOINC(Berkeley Open Infrastructure for Network Computing)[1]がある。BOINC は UCB が開発した PC グリッドのプラットフォームである。BOINC はクライアント、サーバで構成される。クライアントは Windows, Mac, Linux など多様なプラットフォームに対応している。BOINC クライアントは、スクリーンセイバーを含み、PC の負荷が低くなると自動的にタスクを実行する。タスクは BOINC サーバから取得する。プロジェクト主催者は BOINC サーバを設置し、タスクを配布する。プロジェクトを主宰していない場合には、volunteer computing として BOINC を利用する。BOINC には大規模なプロジェクトが数多くある。中でも WCG(World Community Grid)[2]は複数のプロジェクトをまとめたグリッドポータルであり、多数のタスクをクライアントに供給する。WCG のようなポータルサイトがあるため BOINC は継続的にタスクを処理することができる。

PC グリッドの問題点は省エネでないことである。日本では震災後、原子力発電への批判が高まり、省エネが重要な課題となっている。PC グリッドは余剰資源を利用するが、余剰な PC は電力を無意味に消費する。グリッドに利用することで有意義な電力消費となるが、むしろ節電した方がよい。震災だけでなく地球温暖化の観点からも節電が重視されている。このような時勢ではグリッドの利用は困難である。

そこで、我々は PV(Photovoltaic)発電に基づくグリッド（PV グリッド）[3][4][5][6]を提唱している。PV グリッドでも BOINC を用いた。これはタスクの供給力に優れるためである。

クラウドとグリッドは相補的な関係にある。グリッドの問題点は余剰資源をボランティアに頼っていることであった。クラウドのサーバは常に稼働しているため常に余剰資源を発生させている。そもそもクラウド自体が余剰なサーバ資源を仮想化して小売することである。したがって、クラウドにはグリッドに必要な多量の余剰資源がある。一方、クラウドは常にタスクを必要としている。タスクが不足するとクラウドの利用率が低下する。クラウドは常に稼働するため、利用率が低下すると、エネルギーを消費しても有益な情報が生産されない。このように、グリッドはクラウドにタスクを供給し、クラウドはグリッドにリソースを提供する。

Web サービスは安定したクラウドへ移行する。データグリッドもクラウドへ移行する。しかし、PV グリッドを除く PC グリッドは省エネでないため、このような PC グリッドは衰退する。そのため、草の根ユーザによるグリッドリソースの提供は減少する。しかし、プライベートクラウドがグリッドリソースを提供できれば円満に解決する。

また、我々はグリッド上でグリッドを行うメタグリッド[7][8][9][10][11]を開発した。このメタグリッドでは、グリッド上のノードでグリッドノードのインスタンスをタスクとして動

かす。これによりマルチコアに未対応のアプリケーションも並行に動作させることができる。この技法はGoCにも応用されている。本論文はクラウド上にグリッドを構築するが、メタグリッドはグリッド上にグリッドを構築している。両者は類似の技法である。

2.2 過飽和クラウド

過飽和クラウドは、与えられた資源を最大限利用するクラウドである。我々は、過飽和を物理資源以上の論理資源を割り当てることと定める。例えば、CPU資源はコア数として計量される。あるノードのインスタンスのコア数の合計がそのノードの物理的なコア数より大きいとき、そのノードは過飽和状態にあるという。過飽和クラウドのインスタンス数はメモリ容量で決定される。メモリを過飽和状態で利用すると、swapにより性能が大きく低下する。そこで、メモリの過飽和は推奨できない。一方、CPUの過飽和は著しい性能低下を起こさないため容認できる。ディスク資源は余剰傾向にあり、ほとんど飽和しない。よって、メモリ容量がインスタンス数を決定する。

過飽和クラウドでは、例えばノードは100GBのメモリを持ち、インスタンスは0.5GBのメモリを消費するとする。結果、理論上はノードあたり約200インスタンスを稼働できる。

2.3 教育用クラウド

ここでは既存の教育用クラウドと我々が提案する教育用クラウドについて述べる。教育用クラウドの目的は高度ICT技術者の育成に必要な技能を習得する教育環境の構築である。

教育用クラウドには2つのサービスがある。1つ目は授業支援サポートである。このサービスは履修登録やメール、ファイルサーバなど生徒が授業を受講する前準備として必要となるサービスをサポートする。例えばmanaba[12]やMicrosoft Live@edu[13]などである。

2つ目が授業自体のサポートである。このサービスは授業をやる上で必要となるインスタンスを提供する。教育用はユーザが基本的なシステム管理技術から大規模サーバ運用技術など学習を目的とする環境である。学習目的の環境では高い性能を求めないため、導入コストが低い。例えばEdubase[14]や東京工科大学のLecture CLoud[15]。また、教育用とは別に研究用クラウドがある、これは試験環境の性能評価のための環境である。このシステムではインスタンスでありながら物理マシンと同等の性能を有することが必要となる。しかし教育用と比べ高い性能を有する物理マシンが必要となりコストが増大する。

我々が提案する教育用クラウドは、生徒が管理者権限を持ち、複数のインスタンスを利用できる環境である。このような環境では管理者権限を有する操作、アプリケーションのインストールや設定ファイルの編集などが自由に行うことができる。さらに複数台のインスタンスが利用できるためLinuxの基本的なシステム管理から分散処理(Hadoop, Cassandra, etc)のような大規模サーバ環境を構築することができる。

我々の教育クラウドはすでに以下のようなサービスを提供している。基本的に教育クラウドはIaaS(Infrastructure as a Service)である。利用者の学生は初心者である。よってCLI(Command Line Interface)よりGUI(Graphical User Interface)の方が親しみやすい。そこで、DaaS(Desktop as a Service)を提供する[16]。DaaSではSSHトンネルにより安全な通信路を確保する。また、トンネル接続の手順を簡略化するツールも提供している。また、IaaS上にPaaS(Platform as a Service)の層を構築している。PaaSでは代表的なWebアプリケーションをワンクリックでインストールできる。さらに、PaaS上でソフトウェア開発に特化したサービスを集約したDEVaaS(Development as a Service)を構築した[17]。DEVaaSは必要な時すぐにでも共同開発が始められるように典型的な開発パッケージをサポートしている。

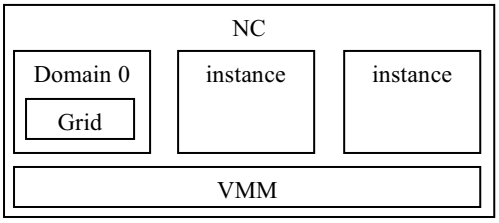
教育用クラウドの問題は、性能面である。既存の教育用クラウドでは高性能な物理マシンを何十台もあるため、十分な性能を持つインスタンスを稼働させることができる。しかし、プライベートで構築する場合にはこの方法ではコストが増大する。我々は2.2節で述べた過飽和クラウドを導入しての教育用クラウドの構築を提案した。これで1台の物理マシンにインスタンスを集約することが可能となる。この環境での問題は1インスタンスあたりの性能の低さである。2.2節で述べたようにCPUの過飽和は容認できるため、プログラミング演習では十分に扱うことができる。しかし、DBのようなストレージアクセスが頻繁に発生する技術ではオーバーヘッドが高いため、十分な演習を行うには不向きである。

3. クラウド上のグリッド

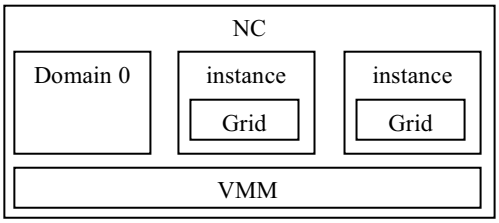
第1章で述べたように教育用クラウドの課題は利用率であり、稼働中は利用率を最大にすることが望ましい。この章では利用率を最大にするため方法を GoC の方式を２つ提案する。

- Node Grid (NG): NC をグリッド資源に提供する。
- Instance Grid (IG): インスタンスをグリッド資源に提供する。

各方式の構成を図 1 に示す。



(a) Node Grid



(b) Instance Grid

図 1. GoC 方式の構成

3.1 NG 方式

NG 方式(図 1 の a)では、NC 自体でグリッドを行うために、NC にグリッドミドルウェアをインストールする。NC は物理マシンであるが、仮想化方式にハイパーバイザを採用する場合は、NC の OS は一つの VM 実体として動作する。ただし、この VM はクラウドによって管理されるインスタンスではなく、むしろインスタンスを管理する。

NG 方式の利点の一つは、余分なインスタンスを起動せずにグリッドを運用することができることである。NC がマルチコア CPU を持つ場合でも、ドメイン 0 の OS はコア数が制限されないため、NC の本来持つ性能に近い計算資源を利用できる。

NG 方式の欠点の一つは、NC にグリッドをインストールする手間がかかることである。NC が増えるほど管理コストは大きくなる。グリッドの設定を変更するには、すべての NC を遠隔操作する必要がある。NC がクラウドによって管理されていないことが原因である。

NG 方式が有効であるには、NC の負荷が共有される必要がある。システム負荷には CPU の利用率やジョブ数など様々な基準がある。負荷の基準によっては NC 全体で共有されないこともある。例えば、ドメイン 0 のジョブ数は他のインスタンスに依存しない。また、ジョブ数は NC で動作するインスタンスとも異なる。本論文では、NC 内で VM によって共有される負荷基準を大域負荷、共有されない負荷基準を局所負荷と名付ける。局所負荷を用いた場合、NG 方式では適切な負荷を見積もることができない。CLC に NC で動作するインスタンスを問い合わせることもできるが、各インスタンスへのアクセスはセキュリティ機構によって禁止されているため、それぞれの局所負荷を知ることはできない。よって、局所負荷を集計して大域負荷とすることはできない。

大域負荷の場合、BOINC のように資源の利用率を自動的に判断して稼働するグリッドミドルウェアをそのまま利用することができる。運用コストは 0 に近い。逆に、局所負荷の場合、常にグリッドは稼働する。その結果、インスタンスの性能を低下させるが、その影響は小さいと予想される。多くのグリッドタスクは CPU ネックである。我々の過飽和クラウドでは、元々過剰にインスタンスを動かしているので、コア数のインスタンスが増加しても影響はほとんどない。これは評価によって検証する必要がある。言い換えると、局所負荷ではグリッドタスクの下限はコア数であるが、大域負荷では 0 になる。その結果、わずかに大域負荷の方が効率的にグリッドを運用できる。

一般に、負荷の基準はグリッドミドルウェアに依存する。したがって、適切なミドルウェアを選択する必要がある。BOINC は大域負荷に適する。

3.2 IG 方式

IG 方式(図 1 の b)では、個々のインスタンスにグリッドミドルウェアをインストールし、負荷に応じてインスタンスを稼働する。

IG 方式の利点の一つは、すべての NC にグリッドミドルウェアをインストールする必要がないことである。グリッドミドルウェアをインストールしたイメージを一つ用意しておけば、クラウドによって稼働中の NC に自動的に配分される。

IG 方式の欠点の一つは、グリッドインスタンスの起動を適切に制御する必要があることである。すなわち、運用コストが小さくない。このような運用を管理者が行うのは現実的でない。そのため、自動的にグリッドを運用するシステムが必要となる。もう一つの欠点は、グリッドインスタンスが複数の NC に対して適切に分散する必要があることである。グリッドインスタンスの分布に偏りがあると、クラウド全体で十分な性能が得られない。

IG 方式の有効性も負荷の共有に依存する。負荷が共有されると運用が容易になる。

3.3 SLIS と MSIS

使用するインスタンスには二つの戦略が考えられる。

一つの大きなインスタンス戦略(single large instance strategy, SLIS)：コア数分の CPU 資源を持つ大きなインスタンスを 1 つ動かす。その特徴は NG 方式とほぼ等しい。一つのインスタンスイメージを用意すればよいので、むしろ NG 方式より管理が容易である。しかし、インスタンスを NC へ均等に分散する必要があるため運用が難しい。インスタンスを動かす NC を明示的に指定できればよいが現段階では困難である。局所負荷の場合は負荷に下限が生じる。

複数の小さなインスタンス戦略(multiple small instance strategy, MSIS)： 1 コア分の CPU 資源を持つ小さなインスタンスをコア数だけ動かす。この方式では、NC を明示的に指定できなくても、確率的にインスタンスを分散させることができる。確率的な変動の範囲内で均一化する。ただし、複数のインスタンスを稼働させる分、資源の消費は大きい。特に、過飽和クラウドではメモリの消費が大きい。例えば、WCG(World Community Grid)の中でもっともメモリを消費するプロジェクトの必要メモリサイズは 1GB であり、そのため SLIS では約 2GB 割り当てる。一方、MSIS ではコア数を 8 とすると 16GB(=8*2GB)を消費する。過飽和クラウドではインスタンスあたりの平均メモリ量が 0.5GB であるから、稼働可能なインスタンス数が約 30 減少する。これは大きな問題である。

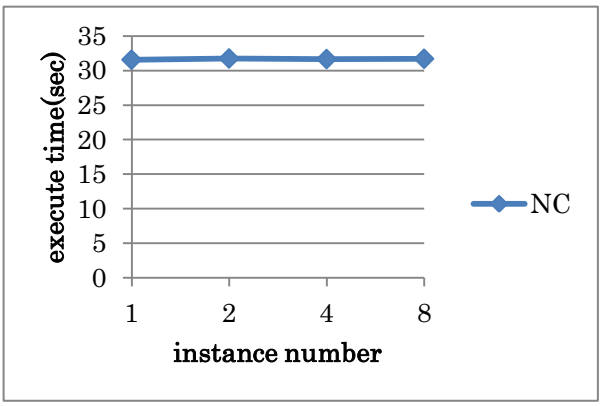


図 2. インスタンス数の増加の影響

ここで、二つの方式を比較する。大域負荷の場合は SLIS が問題なく良い。局所負荷の場合は、SLIS を用いるためには明示的なノード指定が必要か、あるいは必要以上の負荷をかけることを承知の上で複数の大きなインスタンスを動かす必要がある。一方、MSIS は CPU 資源を有効に利用するが、メモリ資源を過剰に消費する。MSIS でメモリ資源を有効に活用するには、グリッド以外のインスタンスが増えるにつれ、グリッドインスタンスを減らす工夫が必要である。

MSIS あるいは明示的に NC を指定できない SLIS では、複数のインスタンスを稼働する必要があり、その数の増減を制御することが望ましい。これには分散制御と集中生制御の２つの方式が考えられる。

分散制御では、起動したグリッドインスタンスは、同じノードで実行されているインスタンスの数と種類を調べ、負荷が過剰であると判断した場合には、自主的に停止する。しかし、この方式では、複数のインスタンスが同時にグリッドを起動することで、計算の前に負荷を超える可能性がある。これが周期的に繰り返されると饑餓状態になる。これを避けるには、イーサネットのように起動タイミングを乱数で分散させることが有効である。

集中制御では、グリッドインスタンスを起動するための制御ノードを設ける。制御ノードは NC の状態を調べて、必要に応じてインスタンスを増加・減少させる。このような制御ノードをグリッドコントローラ(Grid Controller, GC)と名付ける。GC はインスタンスの一つである。

次に、IG 方式におけるグリッドインスタンスを設計する。グリッドミドルウェアとして BOINC を採用する。また、BOINC で実行するタスクは WCG のプロジェクトから選択する。グリッドインスタンスは BOINC のタスクを処理できるスペックが必要である。WCG によると、最大のスペックを要求するプロジェクトは The Clean Energy Project であり、そのスペックは 1GB メモリ、2GB ディスクである。最小のスペックを要求するプロジェクトは Help Cure Muscular Dystrophy - Phase2 であり、そのスペックは 64MB メモリ、50MB ディスクとなっている。

本論文では、次の節で SLIS と MSIS に基づく IG 方式、SLIS-IG、MSIS-IG を比較する

4. 評価

ここでは GoC 方式の比較評価を行う。今回の評価で用いた教育クラウドの仕様を以下に示す。ここで、NC の OS は Ubuntu desktop 11.10(gnome)である。

表 1. クラウドの仕様

	#cores	Memory[GB]	Disk[GB]
NC	8	96	1000
instance	1	0.5	6

4.1 CPU ベンチマーク

まず、過飽和クラウド自身の評価を述べる。過飽和クラウドでは、元々過剰にインスタンスを動かしているので、コア数のインスタンスが増加しても影響はほとんどないことを確認する。ここでは独自の CPU ベンチマークを稼働したインスタンス上で実行し、その平均応答時間を測定した。結果を図 2 に示す。応答時間は、コア数 8 までは一定である。よって、応答時間に対する影響は小さい。

次に、BOINC の負荷基準が大域か、それとも局所か調査した。1 つの NC でインスタンスを 2 つ動かし、DaaS を用いてそれぞれのパフォーマンスモニタを観察する。両方で同じ負荷が観測されれば、負荷は大域的である。結果は異なる負荷が観測された。この結果から負荷は局所的であるといえる。

次に、NC を指定してインスタンスを起動する方法について述べる。教育クラウドでは NC と CC が 1:1 に対応する。CC はゾーンと対応する。よって、ゾーンを指定することで明示的に NC を指定することができる。

4.2 BOINC による性能評価

最後に BOINC による GoC の性能を示す。SLIS-IG 方式と MSIS-IG 方式の環境は表 2 に示す。また、両方の方式にて 3 日間グリッドを運用した結果を表 3 に示す。

表 2. SLIS-IG 方式及び MSIS-IG 方式の仕様

Method	#instances	#cores	Memory[GB]	Disk[GB]
SLIS-IG	1	8	4	6
MSIS-IG	8	1	0.5	6

表 3. GoC のスループット

Method	points	#tasks	Runtime (d:h:m:s)	Throughput [/s]
SLIS-IG	67296	74	3:14:29:48	0.000238
MSIS-IG	69249	88	13:15:24:00	0.000075

ここで、MSIS-IG のランタイム時間(runtime)は稼働した 8 つのインスタンスの総計である。また、MSIS-IG 方式での#task は稼働した 8 つのインスンタスの総タスク数となっている。WCG における評価基準には獲得ポイント(points)とスループットが考えられる。前者は WCG 独自の基準で

あり、貢献度を反映している。後者は純粋な科学的基準である。獲得ポイントは MSIS-IG 方式が優れる。しかし、スループットは SLIS-IG 方式の方が優れる。

以上の結果から考察すると、両方式はクラウドの運用状況によって使い分ける必要がある。WCG 中での貢献を認められること（例えばランキングを上げることなど）を目的とするならば MSIS-IG 方式は有効である。しかし、純粋な処理効率の点では SLIS-IG 方式が有効である。グリッドを普及する目的で MSIS-IG 方式を採用することは必ずしも悪いことではない。

5. まとめ

本論文では、グリッドを効率良く運用するため、クラウド上でグリッドを構築する GoC 方式を提案した。GoC 方式では、クラウドとグリッドの双方の利点を享受できる。今後は、複数の NC/CC を持つ教育クラウドにおいて不必要な NC を停止することで省エネに努める。

参考文献

1) BOINC, http://boinc.berkeley.edu/
2) World Community Grid, http://www.worldcommunitygrid.org/index.jsp
3) Kenichi Fujii, Motoi Yamagiwa, Minoru Uehara: ”Solar Powered Grid Based on Reused PCs”, In Proc. of the 5th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS2011), pp.257-262, (Seoul, Korea, 2011.6.30-7.2)
4) Kenichi Fujii, Motoi Yamagiwa and Minoru Uehara: "A Study on Improving the System of Grid Powered by Solar Cell", In Proc. of 3rd International Workshop on Information Technology for Innovative Services(ITIS2011) in conjunction with the 14th International Conference on Network-Based Information Systems(NBiS2011), pp.535-540, (2011.9.7-9,Tirana,Albania)
5) Kenichi Fujii, Motoi Yamagiwa, Minoru Uehara: “A Study on Operation of Photovoltaic Grid System Using Weather Forecast”, In Proc. of the 3rd IEEE International Conference on Intelligent Networking and Collabolative Systems (INCoS2011), pp.157-163, (2011.11.30-12.2, Fukuoka, Japan)
6) Kenichi Fujii, Motoi Yamagiwa, Minoru Uehara: “An experiment and discussion of the reboot feature in the Photovoltaic Grid System“, In Proc. of 4th International Workshop on Information Technology for Innovative Services(ITIS-2012-03) in conjunction with 2012 26th IEEE International Conference on Advanced Information Networking and Applications (AINA2012), pp.987-992, (Fukuoka, Japan, 2012.3.26-29)
7) Kenichi Tanaka, Minoru Uehara, Makoto Murakami, Motoi Yamagiwa: "Web-Disk-Based Efficient Application Delivery Method for Grids", In Proc. of 2009 International Conference on Network-Based Information Systems(NBiS2009), pp.391-397, (2009.8.19-21)
8) Kenichi Tanaka, Minoru Uehara, Hideki Mori: "Performance Evaluation of a Grid using Windows PCs", Inderscience publisher, International Journal of Web and Grid Services,Vol.4, No.4, pp.395-417, (2008)
9) Kenichi Tanaka, Minoru Uehara, Hideki Mori: "Building a Linux Grid on a Virtual Machine using a Windows Grid", Springer LNCS Volume 5186/2008 Network-Based Information Systems(NBIS2008), pp.132-141 (Turin, Italy, 2008.8.21)
10) Kenichi Tanaka, Minoru Uehara, Makoto Murakami, Motoi Yamagiwa: "Office Grid based on Windows PCs", In Proc. of the Third International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC-2009), pp.427-432, (2009.3.16-19)
11) Kenichi Tanaka, Minoru Uehara, Hideki Mori: "A Case Study on Linux Grid on Windows using Virtual Machine", In Proceedings of 4th International Symposium on Frontiers in Networking with Applications(FINA2008), pp.195-200, (2008.3.25)
12) Manaba, http://manaba.jp/ja/function-course/
13) Microsoft Live@edu, http://www.microsoft.com/liveatedu/
14) Edubase, http://edubase.jp/
15) 授業クラウドプロジェクト http://www.oss.teu.ac.jp/project/lcloud/
16) 木部真一郎, 小山輝明, 上原 稔: "クラウド教育環境における DaaS の評価", RIS2012,信学技報, pp.1-6 (2012.4.10)
17) 松本勝慶, 木部真一郎, 上原 稔, 森 秀樹: "教育用クラウドにおける DEVaaS の提案", RIS2012, 信学技報, pp.1-6 (2012.4.10)