

分散クラウド環境における SLA を考慮した WEB システムの多目的資源割当最適化

川勝崇史^{†1} 棟朝雅晴^{†2}

災害対策や可用性の高いシステムを構築するために、分散クラウド環境における WEB システムの多目的資源最適割当モデルを提案する。モデル化にあたっては、ロードバランサーによるリクエストの負荷分散やスケールアウトが自動でできるようなシステムを前提とし、ユーザーが求める SLA を満たしたサーバーのスケジューリングを行なう。最適化にあたっては、コスト・レスポンス・リクエスト処理量の三つの目的関数をパラメータとして多目的遺伝的アルゴリズムを用いた最適解の探索を行い、その妥当性・有効性について検証する。

Multi-objective resource allocation optimization of WEB systems considering SLAs in distributed cloud environment

Takafumi Kawakatsu^{†1} Masaharu Munetomo^{†2}

We propose a mathematical model for optimal resource allocation of the WEB systems in distributed cloud environment. In the proposed model, we allocate WEB systems with load balancers to cloud systems that satisfy SLAs according to requests from users, which also have a capability of scale-out in increasing the overheads. We solve multi-objective optimization problems considering cost, response time, and throughput using a multi-objective genetic algorithm, and show the effectiveness of the proposed framework through experiments.

1. はじめに

現在 WEB 多層モデルを使用した SNS、オンラインゲーム、e コマースをサービスとして提供するビジネスが増加している。それに伴ってシステムのサーバーがダウンすることによる企業損失などを強く考えなくてはならなくなった。

2011 年の東日本大震災以降、一カ所にシステム(データセンター・サーバー)を集中配置することによる危険性を論じることが多くなった。クラウドを使って、地理的に離れた場所にあるシステムを相互に繋ぐことによってその危険性を分散させ災害に対する予防する動きが活発となっている。加えて、システムを提供するうえで、サービスを提供する側は故障や急なリクエストの増加にも耐えうるシステムを作る必要がある。それぞれ Disaster Recovery(DR)や SLA, Load Balancer(LB)によるリクエストの負荷分散、スケールアウトすることでどのように対応するのかも考えなければならないため、より最適化されたシステムが必要となる。

システムを異なる場所に配置は、計算機資源の共有や管理が複雑になるので、クラウド化し各データセンターのクラウドを相互に接続したインタークラウドを用いる。この時、異なるデータセンター間で実使用率が低い物理サー

バーの CPU や HDD を有効活用するために、リソースを相互利用し有効活用する。この時、各データセンターではコストやスペックが同じではないためどのシステムにどのクラウドの計算資源を割り当てるかが重要になってくる。

本研究では、クラウド間を相互に接続させたインタークラウド環境を用いて、どのインタークラウドを使用するかでシステムのコスト・遅延時間・WEB リクエスト処理量・SLA を同時に考えた多目的資源割当最適化を行い、どのように資源割当をすれば可用性・冗長性のある、よりよい WEB システムを構築できるかを検討する。

2. 関連研究

関連研究としては、Francois Legillon らによりクラウドを使ったシステムの設計・構築の最適化をする手法が提案されている。これは各クラウドの CPU 処理力や I/O, SLA を数値として計算しその状況の中で最適なものを選ぶ研究である。[3]

また、Rakjkumar Buyya らにより環境に優しいクラウドシステムという観点からの研究が提案されている。これはクラウドでサーバーとして使う Virtual Machine(VM)の電力を省電力することやクラウドを構築しているデータセンターのエネルギー効率化の処理を考えた研究であり、具体的にはクラウドのエネルギーの効率化の管理のアーキテクチャや資源割当最適化の方策とスケジューリングアルゴリズムについて QoS と各デバイスの電力使用率を考えた研究である。これによって来たる温暖化を考えたシステムを

^{†1} 北海道大学大学院情報科学研究科
Graduate School of Information Science and Technology, Hokkaido University.

^{†2} 北海道大学情報基盤センター
Information Initiative Center, Hokkaido University

構築することができる.[4]

上記のように、現在クラウドを使った研究は数年前と比べ格段に増えている。クラウドは、今までの技術を複合し応用したものであるので変わらない部分も多い。現在では、その技術をどのように高パフォーマンス・高信頼でシステムを稼働させていくかに焦点が当てられている。

近年、企業や研究機関でクラウドを実際にシステムに新規導入している所も多々ある。ここではクラウドの強みであるスケールアウトを最大限に活かし、必要な時に必要な分だけシステムを柔軟に組み替えることができるシステムを構築している。また、導入数が増えるにつれて新しいビジネスも生まれてきている。「cloudability」[10]・「newvem」[11]という企業によりシステムの最適化をビジネスに取り入れたものである。これらは企業で扱うクラウドの管理を行い、開発者や運用している人にクラウドの現状を知る方法を提供し、リソースの無駄遣いや異状の発生を防ぎ、ユーザーにスケールアウトすべきかどうかを提案するサービスである。

本研究では、地震や津波などの災害からの DR を考えた WEB システムを提案する。上記の研究と異なり、地理的に分散したクラウドを相互に連携させたインタークラウドを用いてコスト・遅延時間・WEB リクエスト処理量・SLA を考えた多目的最適化を考える。

3. クラウドコンピューティング

クラウドとは、ネットワークを介して、データセンターなどの計算資源をサービスとして利用する形態のことである。現在では、システムをクラウド化する企業も存在し、中には普及はしているが、SLA やセキュリティなど、依然運用上の問題もあるが改善が進んでいる。

3.1 Amazon Web Service

クラウドサービスの中でも一番よく知られているのが Amazon Web Service(AWS)である。AWS のサービスとしては、Elastic Compute Cloud(EC2)・Elastic Block Store(EBS)・Simple Storage Service(S3)・Elastic Load Balancing(ELB)・ROUTE53・Cloud Watch・Auto Scaling・Relational Database Service (RDS) の主な 8 つのサービスが存在する。これらはすべて利用した分だけ料金を支払う従量課金制で、サーバーのスペックや利用地域、データ転送量などに応じて課金をするサービスである。

AWS で出来ることは snapshot によるデータのバックアップ、サーバーの複製、動的なサーバーのスペックアップ/ダウン、サーバー台数の増減(Scale out)、サーバーの冗長化、各 VM のヘルスチェック、DNS の設定、負荷分散の方法など、これらを組み合わせたものを構築することができ

る。

今回、数理モデルを考えるにあたって上記の AWS の 8 つのサービスを用いて Web システムを考えた。下図に示したものがそれである。システム構築するだけであるならば AWS で十分である。しかし、AWS では地域や場所によって Virtual Machine(VM)のパフォーマンスにかなりの違いが生じるので実際に使用すると考えるとその差も考えて構築しなければならない。

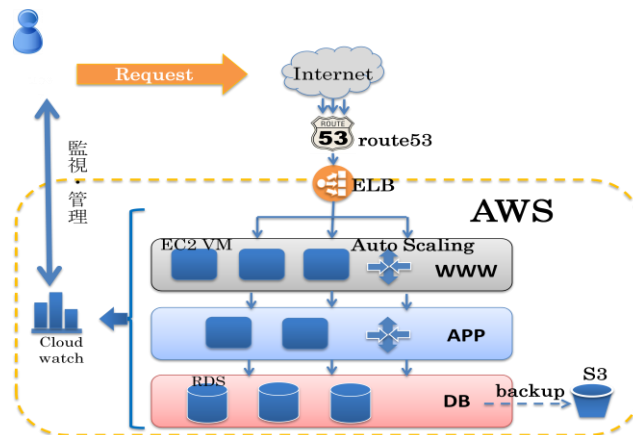


図1 Amazon Web Service 構築例

3.2 インタークラウド

インタークラウドとは、AWS のようなパブリッククラウドとは異なり、地理的に離れたプライベートクラウドを Virtual Private Network(VPN)で相互に接続し連携させて一つのクラウドのように利用する方法である。これにより今まで使用率の少なかったデータセンターを効率的に使用することができる。

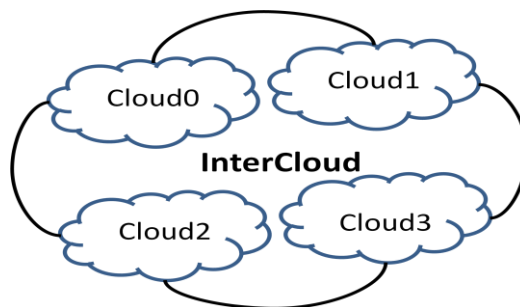


図2 インタークラウド

3.3 マルチクラウド管理

現在 AWS や Eucalyptus, CloudStack, OpenStack などクラウドにもかなりの種類がある。これらを管理するためにマルチクラウド管理のためのミドルウェアがいくつか存在する。

例としては Scalr (OSS 版/有償版) や RightScale (有償) 等、クラウドを自動で制御するためのミドルウェアがある。

これを使うことによって複数のデータセンターのクラウドの稼働状況を管理し、急なリクエストがあった場合は他の

クラウド(Amazon, Google, etc.)から一時的に仮想マシンを調達して対応することも可能となる。これによりクラウドの組み合わせによってユーザーが求めるクラウドシステムを容易に構築できるようになり、加えて管理も容易になった。

3.4 Service Level Agreement (SLA)

サービスの目安としてよく使われる指標であり、通信サービス事業者が、利用者にサービスの品質を保証する制度のことである。回線の最低通信速度やネットワーク内の平均遅延時間、利用不能時間の上限など、サービス品質の保証項目や、それらを実現できなかった場合の利用料金の減額に関する規定などをサービス契約に含めることを指す。

転じて、最近ではクラウドサービスの可用性/安定性指標としても使われるようになっている。

3.5 WEB システム

クライアントサーバーモデルとは、コンピュータ・システムにおける分散処理形態の一つであり、サーバーはクライアントからの処理要求を受けて実際の処理を行い、その結果をクライアントに返すというモデルである。一般的には、共有プリンタやデータベースなどの情報資源を集中的に管理するサーバーと、サーバーを利用するクライアントで構成されている。Web の中にもクライアントサーバーモデルに基づくものもあるが、現在は3層クライアント・サーバー・システムを採用するものが主流である。三層アーキテクチャは、「ユーザインタフェース(WWW)」・「アプリケーションサーバ(APP)」・「データベース(DB)」という3つで構成され、それらが各々に独立したモジュールとして開発または保守される。この利点としてこれらのモジュールは技術の進歩や要求に合わせて各層を個別に書き換えることができる。つまり、PCのOSを替えたとしても影響が出るのはユーザインタフェース層のみである。

また、このモデルはプレゼンテーション層がデータ層に直接通信することはないのでミドルウェアに必ず一度通信を行う。また Web 開発の分野において Amazon などのオンラインショッピングのウェブサイトの構成などの e コマースサイトによく使われる構成である。

負荷分散装置/Load Balancer(LB)とは、SLB (Server Load Balancing) と呼ばれ、www などのアクセスを動的に複数のサーバーへ配信する処理および技術である。一般に Web システムではサーバーの処理の負荷分散を行うために L4 スイッチに置かれるものである。

必要性としては、アクセス集中の低下を防ぐことや高い耐障害性が主に考えられる。加えて、サイトの長時間にわたるシステムダウンはビジネスの損害に直結することも考えられる。

負荷分散の実現方法としては、DNS によるラウンドロ

ビンやサーバーのクラスタ化なども考えられるがこの二つはローテーションの結果、負荷の偏りが発生しやすいこととサーバーの障害検知は原則できない。メリットとしては、多彩なアルゴリズムによる、負荷の割り振りによる柔軟性やメンテナンスの容易性、サーバーに対してヘルスチェックを行い、割り当てローテーションをダイナミックに変更することができる高可用性、加えて身の冗長構成により耐障害性を持つ。

最近の分散技術の利用動向として、「負荷分散」よりも「可用性」や「耐障害性」を重視してサービスを落とさないという命題をクリアするための装置導入が多い。特に大規模な処理を行う場合 GSLB(Global Server Load Balancing)を使う例もある。複数サイト間でサーバロードバランスを行う。

災害時等でサイト自体が運用停止しても、ビジネスプロセスを維持しようという目標の実現に、負荷分散技術を応用している。DNS の動きと連動させて、ユーザーのリクエストを最適なサイトへ誘導させる実装例がある。

クラウドを使った Web システムではより膨大なリクエストを処理することを考えているため LB 一つでは足りない。本研究ではインタークラウドに各クラウドを統括する LB を一つ、各クラウドに LB を配置することにより VM のスペックとそのリクエストによってシステムのリクエスト処理に偏りがないようにバランスを保つことを目標にしている。

分散データベースとは、複数のサーバーに分散して置かれたデータベースをそれぞれのサイトまたは、サーバーからあたかも単一のデータベースの如く取り扱えるようにしたものである。分散データベースは、大別して垂直分散型と水平分散型二つの構成形態に分類される。本研究ではよりクラウドで拡張しやすくするため水平分散型を考える。これは各サーバーがそれぞれ異なる種類のデータを保持し、独自の機能を果たすとともに、各サーバーに存在しない情報は他のサーバーに問い合わせる。垂直分散型および水平分散型は、このような特徴を持つものである。これにバックアップ機能や書き込み処理の整合性、スケールアウトなどを考えたデータベースを想定する。

4. 提案する数理モデル

本研究は WWW 層と APP 層と DB 層の三層で形成される Web 三層モデルのシステムにおいて、ユーザーが求める VM(Virtual Machine)の数などの制約条件をインタークラウドマネージャーを通して入力することで自動でシステムをどのクラウドを使用して動的に資源割当を行うかのモデル考える。

実際には、図1の用な WEB 多層モデルに LB を組み込んだスケールアウトが出来るシステムを考える

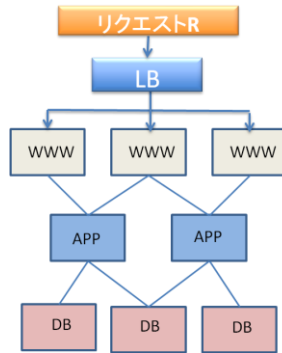


図3 Web 三層モデル

この時、各クラウドに関しては Amazon Web Service を例としてデータセンター各地域に配置し、インスタンス・データ転送料金もこれにならって設定する。以上を踏まえた上でコスト・遅延時間・WEB リクエスト処理量・SLAを満たすように多目的最適化する。

準備として

各 WWW 層・APP 層・DB 層をそれぞれ wN ・ aN ・ dN とし、クラウドの VM 総数を N とする。この時 $N=wN+aN+dN$ となる。

クラウドのステータスとして、それぞれ配置 VM のスペック、限界処理量、ダウンタイムを以下のように表す

$$C_i = \{\text{location, VMspec, RequestMax, downtime}\}$$

$$\text{VMspec}_i = \{\text{cost, CPU, I/O}\}$$

各個体表現については、 n を個体数、 CN をクラウドの個数として以下のように表現する。

$$x[n] = \langle x_1, x_2, \dots, x_N \rangle \quad (1)$$

各 x_i を VM の ID とすると $x_i \in \{0, 1, 2, \dots, CN\}$ ($i = 0, 1, \dots, n$) は各 VM がどのクラウドを使用しているかをベクトルで表す。また、どの VM と繋ぐかも紐付けして表現する。

目的関数 F は以下で定義する。

$$F(x) = \{f_{\text{cost}}(x), f_{\text{response}}(x), f_{\text{request}}(x)\} \quad (2)$$

電気使用量 E_j について以下の (3) 式で定義する。

$$E(x)_j = E_s(x) + E_a(x) \times E_f + \alpha \quad (3)$$

ただし、クラウドの総数を N 、各クラウドの番号を j 、全電気使用量を E_j 、各 VM の待機電力を E_s 、サーバー使用率を E_a 、サーバー 100% 使用時の電力 E_f 、ネットワーク上の電気使用量 α 。

次に電気使用料のコスト関数 $f_{\text{cost}}(x)$ について以下の (4) 式で定義する。

$$f_{\text{cost}}(\vec{x}_i) = \sum_j^N (CP_j E(x)_j + r\text{Cost}_j + t\text{Cost}_j) \quad (4)$$

$$t\text{Cost}(x) = p\text{NW} + \text{trasNW}$$

ただし、各クラウドにおける 1kw 当たりの電気使用料金を CP_j 、VM リソース・データ転送・各 VM 間の転送料金・データ転送時の電気料金をそれぞれ $r\text{Cost}_j$ 、 $t\text{Cost}_j$ 、 $p\text{NW}$ 、 trasNW とする。

レスポンス目的関数 $f_{\text{response}}(\vec{x}_i)$ について以下の (5) 式で定義する。

$$f_{\text{response}}(\vec{x}_i) = R_p + R_d(m, n) \quad (5)$$

$$R_p = t_i / p \quad (6)$$

$$R_d(m, n) = (\beta / c') \times 2(s) \quad (7)$$

ただし、処理時間を R_p 、処理量を t_i 、通信速度を p 、応答時間を $R_d(m, n)$ ただし m, n は通信する VM 同士の位置とする。実測距離を β 、光の速度 c' 、このとき (7) 式に定義するように応答の場合には上り下りとあわせて二倍となる。

WEB リクエスト処理量の目的関数 $f_{\text{request}}(\vec{x}_i)$ について以下の (8)(9) 式で定義する。

$$f_{\text{request}}(\vec{x}_i) = (-1) \times \min \{R / \text{CPU}(x)_j\} \quad (8)$$

$$f_{\text{request}}(\vec{x}_i) = (-1) \times \min \{R / (I/O(x)_j)\} \quad (9)$$

ただし、各クラウドの CPU スペックを $\text{CPU}(x)_j$ 、各クラウドの I/O スペックを $I/O(x)_j$ とする。また、WWW 層 APP 層での処理の場合 (8) 式を、DB 層での処理の場合 (9) 式を用いる。

5. 多目的最適化

多目的最適化とは目的関数が複数存在する最適化のことである。実際に目的関数が単一である問題もありますが複数の問題を最適化する必要がある問題も存在する。今回は目的関数が三つあるので多目的最適化を用いる。

実験で使用したアルゴリズムは多目的遺伝的アルゴリズム NSGA-II である。特徴は非優越ソートによるランキング方法、混雑距離の導入、混雑度トーナメント選択の導入の三つである。各 VM における消費電力について E_s 、 E_f は実際の電力量をもとに設定する。また、各地域の電気料金は今回ウェブサービスに焦点をおいているので産業用の電気料金で計算する。各料金コストについて、 $r\text{Cost}$ と $t\text{Cost}$ $p\text{NW}$ 、 trasNW をクラウド内で規模を自在に変更可能なウェブサービスである Amazon Elastic Compute Cloud (Amazon EC2) を元に各料金コストを設定する。また $c\text{Cost}$ については Amazon Cloud Front をもとに各料金コ

ストを計算する. 各リクエスト処理量については各インスタンスのスペックである CPU と I/O とリクエストの大きさを用いて WEB・APP 層では主に処理が主な負荷になるので CPU とリクエストを用いて計算する. DB 層では入出力が主な負荷となるので I/O とリクエストを用いて計算する. 今回はネットワークでのボトルネックは考えないものとし理論値とし計算した各インスタンスの処理のボトルネックのみで計算を行う.

6. 実験

実験として, 上記のモデルを日本の各都市(札幌・仙台・東京・名古屋・大阪・広島・福岡・沖縄)の八都市をデータセンターの配置場所とし, 電気料金は各都市の産業用電気料金(kw\$/h), Amazon EC2 のリソース・転送料金価格をもとにそれぞれ設定する. 各パラメータを以下のように定義する.

地域	札幌	仙台	東京	名古屋	大阪	広島	福岡	沖縄
電気料金 (kw\$/h)	0.1438	0.1515	0.158	0.1467	0.1721	0.1578	0.1167	0.1345
Downtime	0.05	0.15	0.01	0.15	0.01	0.1	0.15	0.2
DCの転送効率	0.8	0.9	1.0	0.95	0.9	0.85	0.8	0.7

インスタンス	S(1ECU)	M(2ECU)	L(4ECU)	XL(8ECU)
リソース料金 (kw\$/h)	0.088	0.175	0.35	0.7
転送料金(kw\$/h)	0.1	0.1	0.1	0.1
CPU(bps)	1.0	2.0	2.5	3.25
I/O(bps)	50	100	125	162.5
VMの最大転送率 (bps)	50.0	100.0	125.0	162.5
待機電力	100	100	150	200
CPU100%使用時の電力	150	200	200	300

実験上のパラメータを以下のように設定する.

実験パラメータ		実験パラメータ	
インスタンス	S	GA集団サイズ	1000
WWWサーバー数	3	GA世代数	2000
APPサーバー数	2	選択・交叉	0.4
DBサーバー数	3	突然変異	0.1
リクエスト	5(MB)		

災害時におけるサーバーバーストとダウンについて
サーバーバーストとは, サーバーにおけるリクエスト過剰である状態のこと, またダウンはサーバーの停止を意味し, この時の SLA についても考える.

可用性という視点から考えた時システムが停止しないよう

にすることは運用する上で重要である. この時, システムが停止しないという条件を考えると,

- 1) WWW 層/APP 層/DB 層のうち必ず一つ VM が生き残っていること
- 2) DB のバックアップの中にデータが残っていること
※DB 三つの場合, バックアップ率を BR として各 VM が持っているデータが均等であるとする
100%→容量2倍
50%→容量1.5倍
0%→容量1倍

という条件を満たせばシステムは停止しない.

これについて DB のバックアップ率を BU とし, もし災害が起こったとしてある一定の確率でデータセンターが落ちてしまった時の SLA の計算を行う.

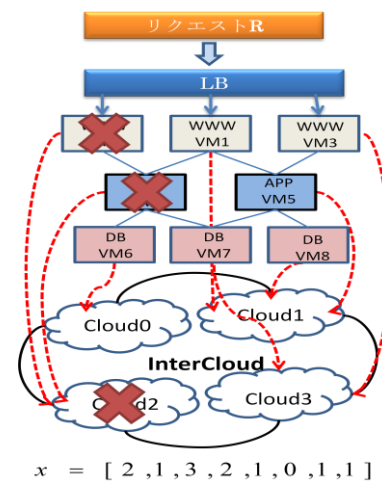
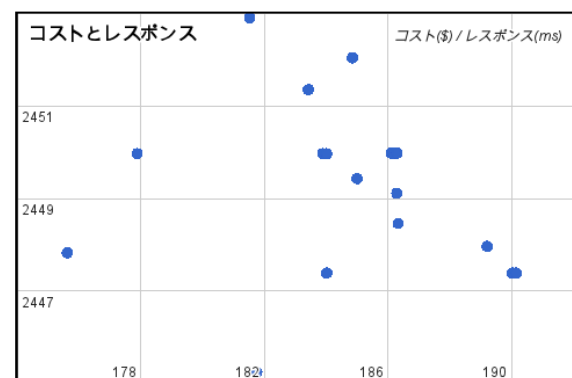


図4 災害時のシステム

これにより, どれほどのコストやバックアップ率であればシステムを停止させることなく運用できるのか検討する

7. 考察



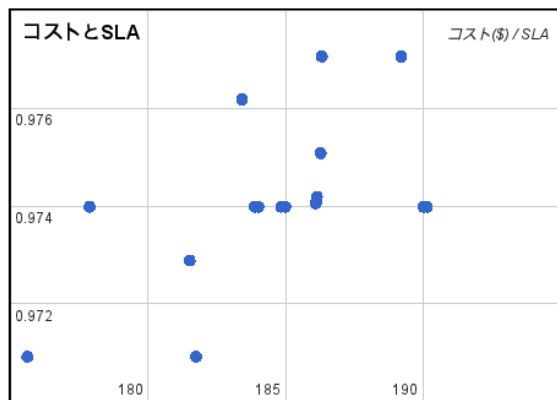


図5 実験結果

実験結果について考察する。

まず図5のコストとレスポンスについて、この二つの目的関数はどちらも最小に最適化されるようになるのでグラフの左下に向かってパレートを作ればよい。設定したパラメータとの関係によりパレート解におけるトレードオフな関係が存在するといえる。しかし、実際にはいくつか存在する。主に $(Cost, Response) = (177, 2448)$ に明らかにずれた点が存在する。この遺伝子はほとんどのVMが広島データセンターで構成されていた。リクエストの処理力とSLAという観点からみるとかなり低い値をとっていた。

またコストとSLAの関係について考察する。この二つの目的関数についてコストは最小にSLAは最大に最適化されるようになるのでグラフの左上に向かってパレートを作ればよい。故に、この問題ではSLAが高いがコストも高い、SLAは低いがコストは安いというトレードオフな関係における問題についてパレート解を示している。この解はある一定以上のSLAを持ったシステムを構築したいときに指標になると考えられる。例えば、この実験では「SLA」＝「可用性」であるのでSLAが「0.99」の時年間で4日システムの停止が起こってしまう。より停止時間を減らすためにSLAを一定以上満たすシステムの構築を行う場合にこの目的関数は有用であると考えられる。

VMの処理速度は一部実測値で行ったので処理時間は計算出来るが応答時間は理論値で考えた。この点に関しては、レスポンスのコスト関数を理論値(光の速度と実測距離)ではなく、ルーターやネットワーク上のホップ数で近さ遠さを加味しより現実に計算する方法を考える必要があった。次に、NSGA-IIアルゴリズムについて今回の実験結果から環境がより複雑になりより難しい状況で最適化を行うにあたり容易に目的関数を増やせ各目的関数の優良個体を保存しながら最適化出来る使用者が自分の望む最適化を選べるというのは利点である。

8. まとめ

本研究では、コスト・遅延時間・リクエスト処理量・SLA

という多目的な観点からトレードオフな関係にある要素を同時に最適化を行った。また、災害対策という観点からシステムを運用していくコストやSLAを最適化することができた。加えてある一定のコストやSLAなどのより制約が厳しくなった場合、その制約内からより良いシステムを構築する場合にも応用できると考えられる。今後、この研究を使用したスケジューリングをクラウドミドルマネージャーに組み込むことによって行いたいと考えている。

謝辞

本研究の実施にあたり、クリエーションライン(株)との共同研究「マルチクラウド構築・運用技術に関する研究及びコントローラの開発」、科研費(22500196)、および、学際大規模情報基盤共同利用・共同研究拠点の支援を受けた。

参考文献

- 1) 棟朝雅晴, 遺伝的アルゴリズム—その理論と先端的手法—, 森北出版, 2008.
- 2) Kalyanmoy Deb, Multi-Objective Optimization using Evolutionary Algorithms, 1995.
- 3) Francois Legillon, Noudine Melab, Didier Renard, El-Ghazali Talbi, Cost Minimization of Service Deployment in a Multi-Cloud environment, IEEE Congress on Evolutionary Computation, 2013
- 4) Anton Beloglazov, Jemal Abawajy, Rakjkumar Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing, Future Generation Computer Systems, 2012
- 5) Bang Minyoung, Asim Munawar, Omar Abdul-RahmanMasaharu Munetomo, Kiyoshi Akama, Optimal Resource Allocation for Distributed Cloud Systems, 2011.
- 6) Rajkumar Buyya, An-ton Beloglazov, Jemal Abawajy, Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges, 2010.
- 7) IEA, "Energy Balances of OECD Countries 2010", "Energy Balances of non-OECD Countries 2010" OECD/IEA, Energy Prices and Taxes, Volume 1999-1/Volume 2005-1/Volume 2011-1, 2
- 8) <http://aws.amazon.com/jp/>,
<http://aws.amazon.com/jp/cloudfront/>
- 9) Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Voshall and Werner Vogels, Dynamo: Amazon's Highly Available Key-value Store, SOS'07, 2007
- 10) <https://cloudability.com/>
- 11) <http://www.newvem.com/>