計算数理工学論文集 Vol. 10 (2010 年 12 月), 論文 No. 05-101210 JASCOME

GAE環境での遺伝的アルゴリズムの実装

Implementation of Genetic Algorithm on GAE Environment

鵜飼 俊介[1], 王 迪[1], 左 毅[1], 北 栄輔[2]

Shunsuke UKAI, Di WANG, Yi ZUO and Eisuke KITA

[1] 名古屋大学大学院情報科学研究科 (〒 464-8601 名古屋市千種区不老町 1)
[2] 名古屋大学大学院情報科学研究科 (〒 464-8601 名古屋市千種区不老町 1, E-mail: kita@is.nagoya-u.ac.jp)

This paper describes the simple scientific computing on the cloud computing using Google App Engine (GAE). In the numerical examples, the minimization of the De Jong Test function is performed by Genetic Algorithms. The results of the first example revealed that, in the GAE environment, one job should be done within 30 seconds in CPU time. For improving this difficulty, the control system in the local personal computer was described. The control system can re-request a job to GAE if the job cannot be done successfully. The final result showed that the control algorithm could perform the re-request of jobs successfully.

Key Words: Cloud Computing, Google App Engine (GAE), Genetic Algorithms (GA).

*

1. 緒論

近年, クラウドコンピューティング (Cloud Computing) と呼ばれる技術が注目されている[1, 2, 3]. クラウドコンピューティングとは, ネットワーク上に存在するサーバが提供するサービス (以下クラウドサービス) を, そのようなサーバを意識することなく利用することができる形態である. クラウドコンピューティングによって提供されるサービスの多くは従量課金制のため, ユーザは利用した資源の量に応じた料金をサービスプロバイダに支払えばよく, サービスが必要なくなった際には即座に利用を中止できる. このようにクラウドコンピューティングには IT コストの削減効果があり, 近年の不況と相まって注目されている.

クラウドコンピューティングにはいくつかの利点がある. 第 1 はコスト削減である. 利用者は業務用アプリケーションの購入や自前でデータセンターを構築する必要が無いので, IT コストを削減できる. 第 2 は拡張性である. 資源が仮想化されているため, 負荷やユーザの増減に対して計算環境を柔軟に拡張できる. 第 3 は, クライアントホストのハードウェア要求が低下し, メンテナンスなしに最新サービスを利用できることである. 第 4 はユビキタス化である. 資源がネットワーク上におかれるため, クライアント側 PC の OS や機器に依存することなく, あらゆる場所から資源を利用できる.

その一方で, クラウドサービスを提供するサーバーがクラッカーに狙われやす, サーバがダウンしたり, クラウドサービスプロバイダがサービス提供を停止してしまった場合, 企業活動全てが停止してしまうこともあり得る.

このような状況を鑑みて, 本研究では自前で計算サーバーを構築して管理運営する代わりに, クラウドコンピューティングサービスを学術研究用に用いる方法について検討する. このために, Google 社が提供する Google App Engine (GAE) を利用する. いくつかのクラウドコンピューティングサービスの中で GAE を利用する理由は, 制約の範囲内において無料で利用することができ, 高機能なソフトウェア開発キット (SDK) が提供されているからである. そして, GAE 上に遺伝的アルゴリズム (GA)[4, 5] を実装する. いくつかのシミュレーションの結果, GAE には計算機資源以外にも制約条件のあることが明らかとなった. それは, 1 回のリクエストを 30 秒以内に終了しなければならない制約である. この制約は, 問題の規模やアルゴリズムによらない. 本研究では, この制約を回避するための制御システムをローカルホスト側で構築する. 簡単な解析例を通してシステムの有効性を確認する.

2. クラウドコンピューティング[1, 2, 3]

2.1. クラウドコンピューティングの背景

クラウドコンピューティングの原型は, 1960 年代のタイム・シェアリング・システム (TSS) に見られる. それが最近になって再び注目され始めたには以下の理由がある.

第 1 は, ネットワーク帯域幅のコスト低下である. 以前はデータをできるだけ利用者側に置いて応答性を向上させ,

2010 年 9 月 7 日受付, 2010 年 11 月 10 日受理
*Dedicated to the memory of Prof. Masataka TANAKA

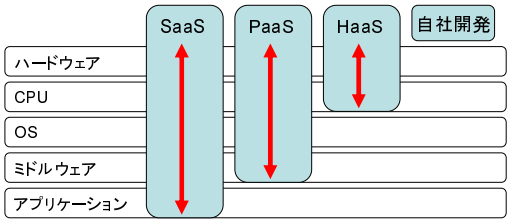

Fig. 1　Cloud computing

Table 1　Software as a Services (SaaS)

Provider	Service name
Google	Gmail, Google Document
Microsoft	Windows Live, Microsoft Office Live
Adobe Systems	Photoshop Express
Oracle	Siebel CRM On Demand
Twitter	Twitter

ネットワーク・コストを削減することが重要であった．しかし，近年のブロードバンド通信の普及により，データを集中管理された遠方のデータセンター内のサーバに置いても十分な性能が実現できるようになった．

第 2 は，PC の低価格化等によりコスト全体に占める人件費の割合が大きくなり，その削減が重要となっていることである．クラウドサービスを利用すれば管理・運用にかかる人件費を削減できる．

第 3 に，ビジネスにおいて情報システムの柔軟性や応答性の重要性が増大していることがある．そのため，ゼロから自前でシステムを作り上げるよりも，他社が提供する既存のサービスを活用した方が有利な場合が多くなっている．

2.2. クラウドコンピューティングの分類

クラウドサービスは，提供されるサービスの深度によって以下の 3 つに分けられる (図 1)．

(1) Software as a Service (SaaS)　ローカル PC にインストールして使うアプリケーションとは異なり，ソフトウェアの機能がインターネットを介して提供されるサービスである．SaaS ではハードウェアからアプリケーションまで全てがインターネットを介したサーバから提供される．短時間でシステムが導入可能で，運用管理の必要が無い．例として，ウェブメールやオンラインストレージ，辞書，地図サービス等がある (表 1)．

(2) Platform as a Service (PaaS)　SaaS を提供するための開発環境や運用環境を提供するサービスである．インターネット経由のアプリケーション実行用のプラットフォームを提供し，アプリケーションの負荷が増加した際サーバの増強を増強する作業（スケーリング）を自動的に行う．PaaS ではミドルウェアからハードウェアまでがサービスとして提供され，開発者は仮想マシン上にアプリケーションの構築を行う．開発言語や環境に制限があり，他のプラットフォームへの移植は難しいことが多い (表 2)．

Table 2　Platform as a Services (PaaS)

Provider	Service name
Google	Google App Engine
Apple	Mobile me
Microsoft	Azure Service Platform

Table 3　Hardware as a Services (HaaS)

Provider	Service name
Amazon	Amazon EC2/S3
ServePath	Go-Grid
xseed	myDC

(3) Hardware as a Service (HaaS)　サーバのハードウェア資源をインターネット経由で提供するサービスである．HaaS では PaaS とは異なり，ユーザがスケーリングを考慮して仮想マシン単位でハードウェアを追加する．HaaS では開発者が任意の OS やミドルウェアをインストールし，その上でアプリケーションを構築する (表 3)．

3. Google App Engine の利用

3.1. 開発環境の準備 [6, 7]

(1) アカウントの取得　Google App Engine (GAE) を利用するためには Google アカウントと App Engine アカウントが必要である [6]．Google アカウントを取得してログイン後，GAE ホームページのスタートガイドに従い，App Engine アカウントを登録する．App Engine アカウントの登録には携帯電話のメールアドレスが必要となる．

(2)Google App Engine SDK　2010 年 1 月現在，GAE では Python と Java 用のソフトウェア開発キット (SDK) が提供されている．これには，App Engine 環境をシミュレートするための Web サーバアプリケーションや，URL をフェッチしたりメールをホストから直接送信したりするための App Engine API が含まれている．今回は Python を利用して開発を行うため，Windows 用 Google App Engine SDK for Python をインストールする [7]．

(3)Python のインストール　今回 Python を用いて開発を行うために，Python をインストールする [8]．

3.2. アプリケーションの開発

(1) ソースコードと設定ファイルの作成　同一のディレクトリにソースコードと設定ファイルを記述する．ソースコードを Python により記述し，設定ファイルを YAML という形式で記述する．

(2) 開発用サーバの起動テスト　SDK に付属している開発用サーバ上で開発したアプリケーションの動作確認を行う．実行するにはソースコードのあるディレクトリより 1 つ上のディレクトリで以下のコマンドを実行する．

```
dev_appserver.py (ディレクトリ名)/
```

また，アプリケーションを実行するには，Web ブラウザで以下の URL にアクセスすればよい．

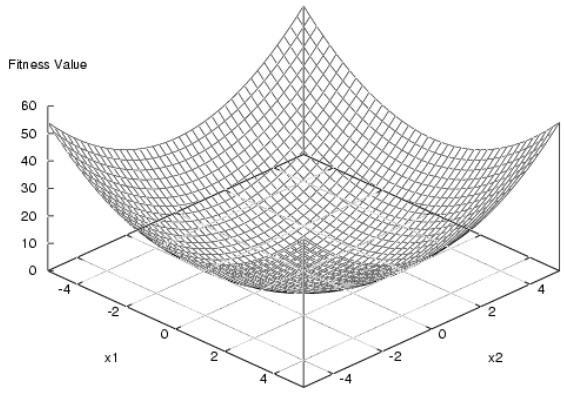

Fig. 2　DeJong Function 1

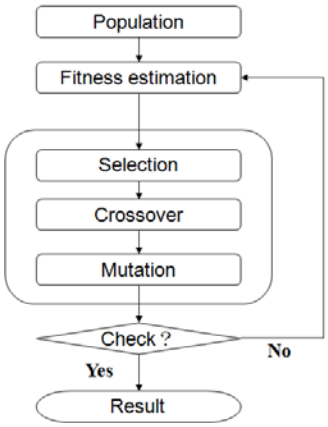

Fig. 3　Flowchart of Genetic Algorithms

```
http://localhost:8080/
```

(3) アプリケーションの公開　ローカル環境でアプリケーションの正常動作を確認した後，アプリケーションを Web 上に公開する．App Engine アカウントにログインし，アプリケーション ID，アプリケーションのタイトル，アプリケーションの公開範囲等を入力して保存する．その後，ローカルのコマンドプロンプトから以下のコマンドを入力する．

```
appcfg.py update (ディレクトリ名)/
```

ここで，(ディレクトリ名) はソースプログラムが格納されたディレクトリを示す．その後，Google アカウントを取得した際に登録したメールアドレスとパスワードを入力する．

公開されたアプリケーションは，web ブラウザを用いて以下の URL にアクセスすることで確認できる．

```
http://(アプリケーション ID).appspot.com/
```

4. Google App Engine での GA 実行

4.1. 関数最小化問題

解析例として，DeJong の標準関数 F1(Parabola)[9] の最小化問題を考える．DeJong の標準関数 F1 は次式で与えられる．

$$f(x) = \sum_{i=1}^{3} x_i^2 \quad (-5.11 \leqq x_i \leqq 5.12) \tag{1}$$

評価関数は設計変数 x_i の 2 乗和で定義され，最小値は 0 である．例として，設計変数 2 個の場合の関数を図 2 に示す．この関数はの解空間は単峰性で凸型である．

DeJong の標準関数 F1 は単峰性関数である．一般に，遺伝的アルゴリズムは，多峰性関数の解を求めることにより有利である．ここでは，GAE における実装の例を示すことが目的なので，式 (1) の関数を用いることとした．

4.2. 遺伝的アルゴリズム (GA)

DeJong の標準関数 F1 の最小化問題を解くために遺伝的アルゴリズム (GA)[4, 5] を用いる．GA は遺伝子情報による生物の進化過程を工学的に模擬したものであり，探索範囲の広い最適値探索問題に対して有効であることが知られている．

GA 処理の流れを図 3 に示す．演算の前に，問題の解を染色体と呼ばれる記号列に置き換える (コーディング)．これにより，1 つの染色体を有する個体はその問題に対する 1 つの解候補となる．この染色体に対して各遺伝子座をランダムに決定することにより複数の個体を生成し，初期集団を作成する．次に，設定した評価関数に基づき，各個体に対する適応度の計算を行う．適応度の最も高い最良個体は次世代まで保存される．それ以外の個体については，適応度を用いて染色体の選択を行い，選ばれた染色体のペアを親として交叉を行う．さらに，集団中のいくつかの染色体に対し，突然変異を施す．ここまでを 1 世代として，新しく生成された染色体に対し再び適応度の計算を行い，さらに遺伝的演算を施す．この操作を何世代か繰り返すうち，集団中には次第に適応度の高い染色体 (解候補) が多く含まれるようになり，より優れた染色体が生成されることが期待される．

GA のアルゴリズムでは終了条件を満たすまで何世代にも渡り繰り返し演算を行うので，適応度関数に非常に高い計算負荷がかかる．

4.3. Google App Engine 側の処理

Google App Engine 上では，大きく分けて以下の 3 つの処理を行う．

- パラメータ入力
- GA 演算
- 演算結果の表示

このうち，GA 演算のアルゴリズムについては 4.2 節で説明したので，それ以外について以下で説明する．

(1) パラメータ入力　GA 演算を行うためには，染色体数，交叉率，突然変異率等を入力する必要がある．これに加えて，本研究では GA の終了条件として，最大世代数を入力する．これらの条件は，Google App Engine の URL にアクセスした際表示されるパラメータ入力 Webpage から入力する．実際の入力 Webpage の例を図 4 に示す．続いて，計算ボタンをクリックすることで Google App Engine 上で GA 演算を開始する．

(2) 演算結果の表示　GA の演算結果の表示は，Google App Engine 側で Webpage に表示することで行う．このと

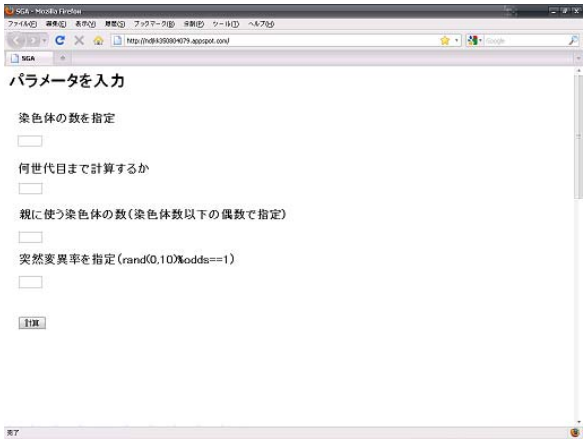

Fig. 4　Input homepage of GAE

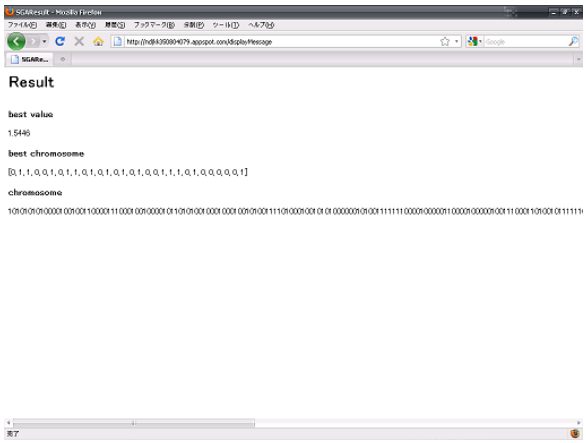

Fig. 5　Output homepage of GAE

き，アクセスする Webpage は以下のところである．

```
http://(ID).appspot.com/displayMessage
```

ここで，ID はアプリケーション ID を示す．

出力結果例を図 5 に示す．best value は母集団中最も評価の高い染色体の評価値を，best chromosome にはその染色体の遺伝子情報を示す．また，chromosome は母集団中の全ての染色体の遺伝子座情報である．

5. 計算環境の改善

5.1. Google App Engine の制約

GAE の利用にはいくつかの利用上の制約がある．無料で利用できる GAE の CPU の使用率，データ転送量，ハードディスク使用率を表 4 に示す．CPU 時間の単位は 1.2GHz Intel x86 プロセッサが同時間で実行できる CPU 時間 (秒) に相当

Table 4　Resource limit of free GAE

Resource	Limit (Daily)
Request	1,300,000
Bandwidth on sending	10 GB
Bandwidth on receiving	10 GB
CPU time	46 CPU hours
Data storage	1 GB

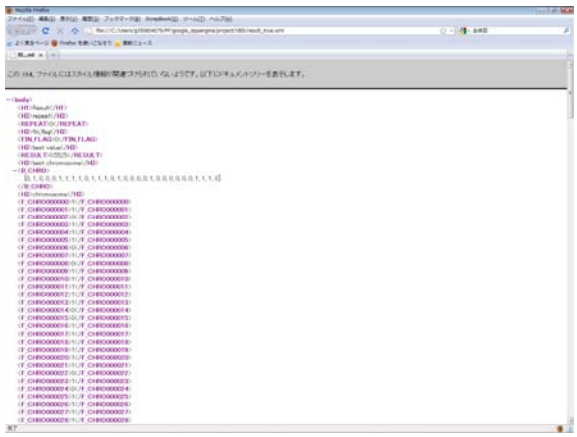

Fig. 6　Webpage of successful output results

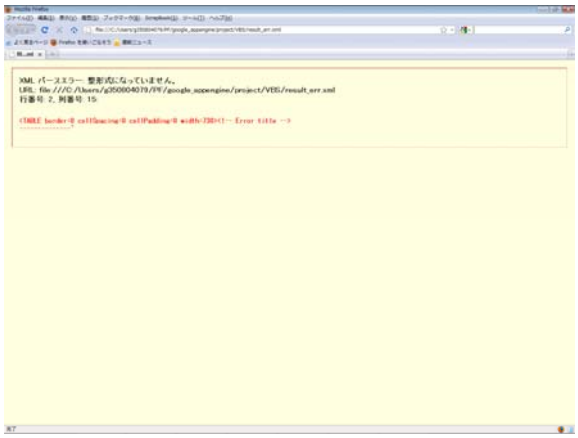

Fig. 7　Webpage of unsuccessful output results

する．また，GAE には有償コースもあり，料金を支払うことでこれらのリソースを追加購入できる．

この他に，先の GAE において GA を実行した結果，GAE には，マニュアルに記載されていない制約のあることが明らかとなった．それは，1 リクエストの処理時間を CPU 時間で 30 秒程度以内としなければならないことである．これを，以下では，“1 リクエスト 30 秒ルール” と呼ぶことにする．この制約は，GAE のマニュアルに明記されておらず，いくつかの数値実験から確認された．本研究で扱う GA の計算環境では，この制限が最も厳しい制約となるので，これを回避できる制御プロセスを開発する．

5.2. ローカルホストの処理プロセス

4.3 節で構築した環境では，正常に処理された場合，図 6 の Webpage が表示される．一方，最大世代数が十分大きい場合，1 リクエスト 30 秒ルールによってエラーとなり，図 7 のような Webpage が表示される．

この制約を回避して計算を継続させるシステムを開発する．Google App Engine へはブラウザを通してジョブをリクエストするので，Internet Explorer で開かれた入力フォームへの入力を容易に実現できる Visual Basic Script(VBS) を用いてローカルホスト上のシステム開発する．全体処理の流れを図 8 に示す．

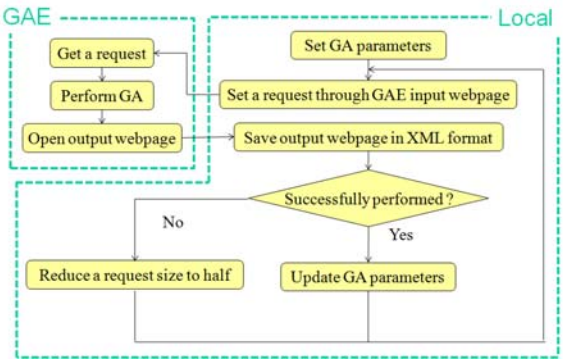

Fig. 8　Simulation process

1. ユーザーが以下の処理を行う.
   (a) ローカル側のシステムを実行する.
   (b) 演算している世代数, 交叉率, 突然変異率, 最大世代数 (終了条件), 1 ジョブあたりの最大世代数をローカルシステムに入力する.
2. ローカル PC のシステムが以下の処理を行う.
   (a) ブラウザを起動し, GAE のデータ入力 Webpage を開く.
   (b) ステップ 1b で入力されたパラメータを自動的にデータ入力 Webpage へ入力する.
   (c) 計算ボタンをクリックする.
3. Google App Engine が以下の処理を行う.
   (a) データ入力 Webpage からデータを得る.
   (b) GA 演算を行う.
   (c) 演算結果を出力 Webpage に表示する.
4. ローカル PC 上のシステムが以下の処理を行う.
   (a) 出力 Webpage の<body>〜</body>を XML 形式で保存する.
   (b) ブラウザを閉じる.
   (c) 保存した XML ファイルを解析する.
   (d) XML ファイルが正常 (図 6) ならば, 世代数と全個体の遺伝子情報を取得する. XML ファイルが不正 (図 7) ならば, 成功終了したジョブでの全個体の遺伝子情報を保存し, 1 リクエストあたりの終了条件の世代数を 1/2 とする.
   (e) 世代数が最大世代数未満であればステップ 2 へ戻る. 世代数が最大世代数に等しければ結果出力し, プロセスを終了する.

ローカル PC 上のシステムにおいて, XML ファイルが不正 (図 7) ならば, 1 リクエストあたりの終了条件の世代数を 1/2 としているが, このサイズを変更することは可能であり, それによって計算コストを改善することができる.

6. 実験

実験に用いるローカルホストの仕様を表 5 に示す. この実験用 PC と GAE を用いて, 最大世代を 10 万世代とした場合の CPU 時間を比較する. ここで, Google App Engine で 10 万世代分の演算を行うと, 終了前に CPU 時間が 30 秒を超えるため演算が終了しない. そこで. 今回開発したローカル側システムを用いて演算を行い, その結果をローカルホストだけで行った結果と比較する.

Table 5　Local host specification

OS	Windows Vista Business SP2 x86
CPU	Intel Core2Duo L7800 2.00GHz
Main memory	1GB

Table 6　GA parameters

Max. generations	100,000
Max. generations per job	50,000 , 25,000 , 12,500 , 10,000
Length of chromosome	30
Crossover rate (%)	40
Mutation rate (%)	10

シミュレーションに用いるパラメータを表 6 に示す. ローカルホストだけで行った結果を表 7 に, GAE で行った結果を表 8 に示す. これらを比較したものを図 9 に示す. 表 7 と表 8 における分割数とは, 10 万世代をいくつのジョブに分割したかを示している. 各分割数において GA 演算を 5 回ずつ行い, それぞれの演算時間の平均を示している.

これらの結果より以下のことがわかる. 今回比較に用いたローカルホストの演算時間と比較すると, GAE の方が約 25 %早く演算が完了している. GAE の CPU 時間は 1.2GHz Intel x86 で評価しており, ローカルホストの CPU である 2.0GHz Core2Duo よりも遅いにもかかわらず, GAE の計算時間は短くなっている. また, ローカルホストと GAE を比較すると, 計算時間のばらつきは GAE のほうが小さく, 安定した計算性能を示している.

次に, 1 リクエストあたりの世代数を 50,000 世代, 25,000 世代, 12,500 世代, 10,000 世代として計算を行った場合の CPU 時間を図 10 に示す. これにより, 1 リクエストあたりの世代数を小さくすると CPU 時間がわずかながら増加することがわかる. これは, 1 リクエストあたりの世代数を小さくすると分割数が多くなり, 演算結果の出力回数が増加, つまり通信コストが増大するためである.

Table 7　CPU time of simulations on personal computer (Unit:Seconds)

Number of jobs	1	2	4	8	10
Simulation 1	84.1	97.5	101.9	135.1	155.1
Simulation 2	102.9	85.1	104.8	131.8	146.9
Simulation 3	99.7	76.2	93.6	125.1	145.1
Simulation 4	83.0	84.9	113.3	128.9	146.3
Simulation 5	91.2	76.9	98.4	132.1	153.6
Average	92.2	84.1	102.4	130.6	149.4

Table 8　CPU time of simulations on Google App Engine (Unit:Seconds)

Number of jobs	2	4	8	10
Simulation 1	65.1	80.2	108.6	123.3
Simulation 2	64.7	79.3	108.6	123.4
Simulation 3	63.9	79.9	108.2	122.6
Simulation 4	64.4	78.8	108.1	123.1
Simulation 5	64.3	79.1	108.7	122.6
Average	64.5	79.5	108.4	123.0

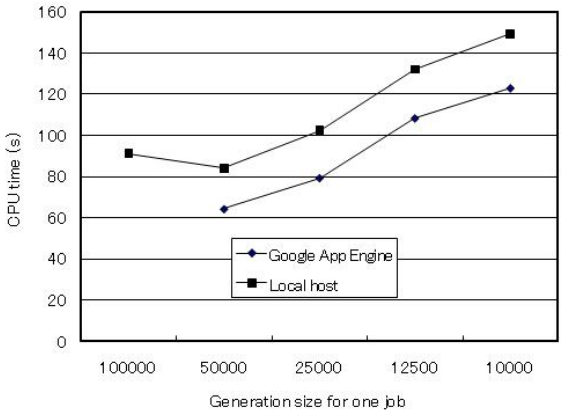

Fig. 9　Comparison of GAE and local host (Unit: seconds)

7. 結論

本研究では，クラウドコンピューティングサービスの 1 つである Google App Engine (GAE) のコンピュータシミュレーションへの有効性を検討するために，GAE 上で遺伝的アルゴリズムを実行する環境を構築した．実験の結果，GAE の利用にいくつかの制約があることが確認された．その中でも，1 ジョブを CPU 時間で 30 秒以内としなければならない制約のあることがわかった．そこで，これを解決するために，全演算を 30 秒以内のジョブにわけて実行するシステムをローカル PC 上で構築した．このシステムで実験を行い，GAE による結果をローカル PC での結果と比較した．この結果，GAE のほうがローカル PC よりも CPU 時間が短く，ばらつきが小さくなることがわかった．GAE のほうが同一 CPU を有するローカル PC よりも CPU 時間が短くことは，GAE で用いられている仮想化システムなど管理システムの優秀さを物語っている．また，ローカル PC では，計算以外にバックグラウンドで動作するプロセスがあり，これらの負荷が最適化計算にかかる CPU 時間を増減させている．GAE は大規模なクラスタシステムであり，そこで動作するロードバランスシステムが優秀なため，計算コストのばらつきが小さくなったものと想像される．

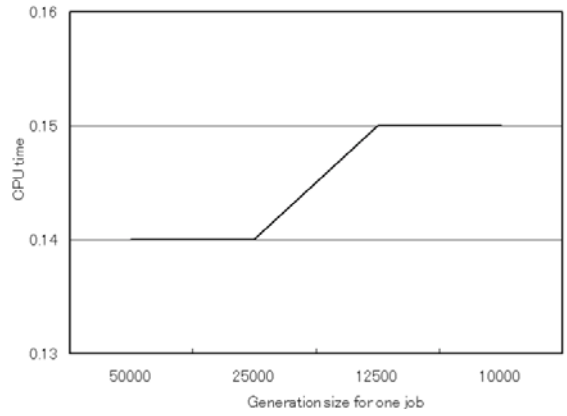

Fig. 10　CPU efficiency

最後に，今後の課題についても確認しておきたい．1 ジョブを CPU 時間で 30 秒以内に抑えるためには，処理全体を細分しなければならないので，細分化するとローカル PC と GAE のデータ転送に要する時間が増加することが確認された．従って，大規模な計算を行うためには，一般のネットワーク環境の改善が必要である．また，ローカル PC 上のシステムにおいて，XML ファイルが不正ならば，1 リクエストあたりの終了条件の世代数を 1/2 とするが，適切なサイズを推定することは可能であり，今後それが可能なようにアルゴリズムを改良する予定である．

参考文献

(1) 中田敦他. クラウド大全—サービス詳細から基盤技術まで. 日経 BP 社出版局, 2009.

(2) 城田真琴. クラウドの衝撃. 東洋経済新報社, 2009.

(3) エリック松永. クラウドコンピューティングの幻想. 技術評論社, 2009.

(4) J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1 edition, 1975.

(5) D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison Wesley, 1 edition, 1989.

(6) Google App Engine: スタートガイド. `http://code.google.com/intl/ja/appengine/docs/`.

(7) Googel App Engine: ダウンロード. `http://code.google.com/intl/ja/appengine/downloads.html`.

(8) Python: Python Japan User's Group. `http://www.python.jp/Zope/`.

(9) Kenneth A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* PhD thesis, University of Michigan, 1975. Dissertation Abstracts International 36(10), 5140B; UMI 76-9381.