IJETCR Open Access

ISSN: 2348 - 2117

International Journal of Engineering Technology and Computer Research (IJETCR)
Available Online at www.ijetcr.org
Volume 3; Issue 2; Page No. 107-112

# Performance analysis based on Google App Engine and Amazon Web Service

**Divya Hada , Abhidha Agrawal**

M. Tech. Scholar,

Department of computer science and Engineering

Jayoti Vidyapeeth Women's University, Jaipur, Rajasthan (India)

**ABSTRACT**

To intend a new application model which reduce the cost and increase the capabilities of the resources. These models include partition of an application into multiple components. These are the main features of this model. In this paper we define the features of Google App Engine which reduce the cost and capabilities of resources. It can be easily maintained. Amazon EC2 also used to enhance the capabilities of resources but Google App Engine is quite batter than Amazon EC2.Google App Engine offer platform as a service (pass).

**Keywords:** Elastic application, weblet, Google App Engine, Amazon Web Service, performance analysis.

## INTRODUCTION

Cloud computing offer several computing models for both services provider and consumers including infrastructure -as-a-services, platform- as-a-services, Software-as-a-services which enable business model such as resources on demand, pay as you go and utility computing [3].Cloud computing is the vast platform for service delivery. We consider cloud computing as to extend the capabilities of resources. There are several approaches, first one is the duplicate the runtime environment of device in the cloud or run the application on device or in cloud. The off device runtime environment is called a "surrogate" [5] a "clone" [8] or a cloudlet [9].Virtual machine technology is used to isolate and host the run time. To make this approach fit with emerging IaaS platform such as Amazon EC2 [1]. When device is running on cloud has attractive properties such as enhance CPU and memory resources. Applications are not need any modification the clone and the physical devices run identical binaries this application also have disadvantages. First, the application on clone needs physical hardware for access. To develop a new application model many challenges are arises in different areas, such as management of heterogeneous computing environment, data management, communications between web-let.

Secondly, replace one processor with another to take full advantages of cloud resources. Third, duplicating a device and run it on the cloud it increases the complexity. The above considerations focus on cloning a complete device environment. These applications are parallel to high-compute-to-communication ratio. For example data mining, media processing, search. Our goal is to design architecture to enable elastic application which consist multiple component called weblet. Each can be launched in cloud. To launch a weblet is based on the application configuration, battery level, and CPU. Offloading and delegating computing proposed by many researchers [12, 11, 10, 13].

### Background

The cloud was used as a metaphor for the internet especially for telephone networks and later used to describe the internet in computer network diagrams and the infrastructure. Nowadays, cloud computing has grown up to a mature industry standard supported by many company. There are many cloud computing platforms such as Google App Engine, Amazon Web Service, HP cloud enabled computing, IBM cloud computing etc.

### 1. Two Leading Cloud Computing Platforms

### A. Amazon Web Service

Amazon's cloud was initialized in 2002 and named Amazon web service. It is a web based remote computing collection. It is constructed based on four key services, Simple Storage Service (S3), Elastic computed cloud (EC2), Simple Queuing Service and Simple DB. In other words, Amazon now provides the storage service, computing service, queuing service and database access service through the internet. Other Services include Amazon Associates Web Services (A2S), Amazon AWS Authentication, Amazon Virtual Private Cloud (VPC) and etc.

### B. Google App Engine



*Corresponding author: Divya Hada*

On April 2008, Google released the beta version of the Google App Engine which allows the developers to develop the application based on python. The developers can also use Google's infrastructure to manage their developing process. For the excessive part, Google will charge 10-12 cents/GB on per CPU per hour basis. The key idea of GAE is to virtualize the apps across multiple data centers and servers.

**C. Comparison of Google App Engine and Amazon Web Service**

The main difference between Amazon Web Service and Google App Engine is that Amazon Web Service is IaaS while Google App Engine is PaaS.

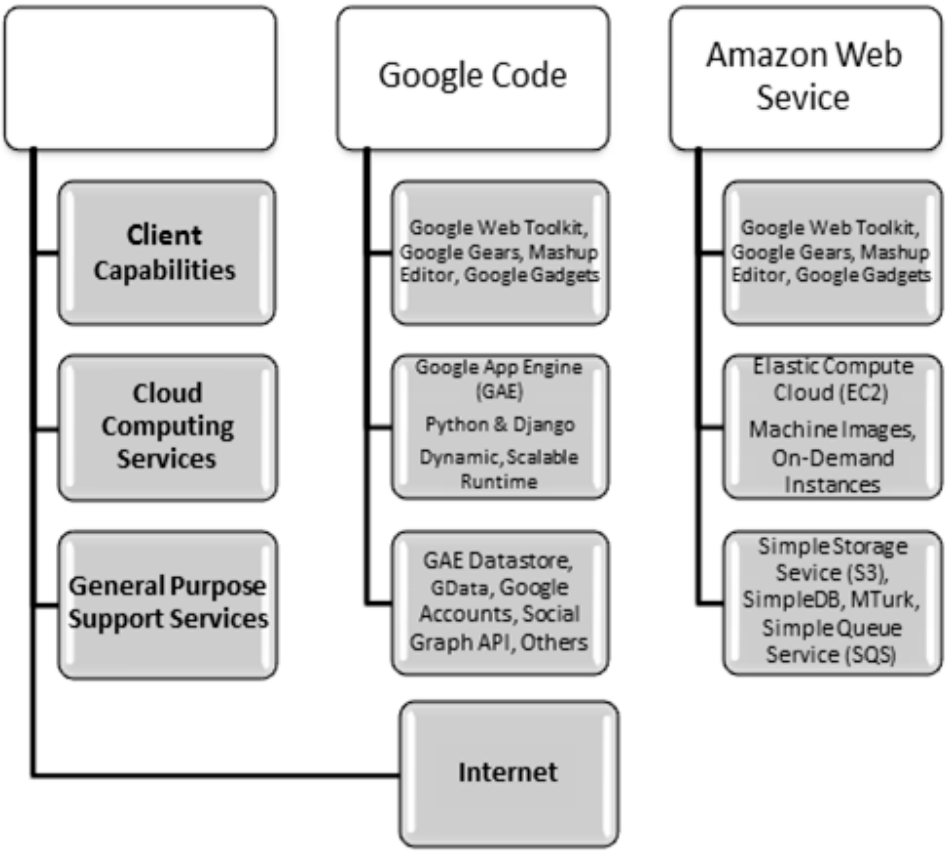


**Figure 1.1: Comparison of Amazon Web Service and Google App Engine**

**2. Performance analysis based on Google App Engine and Amazon Web Service**

**Analysis for Google App Engine, Measuring tools and implementation**

The httperf measurement tool [6] and Planet lab test bed [7] are used in this case. The httperf which was developed by David Mossberg and other staff at Hewlett-Packard Research Laboratories is a test tool used to measure the performance of the web servers. The Planet lab test bed is a virtual lab network established on March 2002, which consists of a bunch of lab machines distributed around the globe most of which are hosted by research institutions. Same tests are running on the Google App Engine and the traditional web servers.

- Metrics Selection

In these two important metrics are describe

- Round trip-time(RTT)
- Network throughput

Round trip time is the time it takes from the data being processed to reach the host and return back to the user. It is an important metrics for cloud computing area since it give better insight into comparison of the latencies of the Google App Engine with the traditional web server. RTT is measure in a second.

The network throughput metric measures the data transferred through the network connection for a period of time. It also responds for the bandwidth of the system. It can show the difference between the bandwidth of Google App Engine and the traditional web server. Network throughput is measure in kB/sec.

**3. Performance Analysis**

The experiment results for the RTT are shown in Table1. The data in Table 1 are collected from *Network-based Measurements on Cloud Computing Services*. [1]

**Table 1.1: RTT results for GAE and traditional web host, data are collected from Network-based Measurements on Cloud Computing Services [1]**

| Image Size | 12kB | | | 350kB | | | 1MB | | |
|---|---|---|---|---|---|---|---|---|---|
| # of req/Planetlab node | 1 | 10 | 100 | 1 | 10 | 100 | 1 | 10 | 10 |
| RTT for GAE(sec) | 1 | 5 | 47 | 1 | 10 | 40 | 1 | 15 | 43 |
| RTT for TWS(sec) | 1 | 13 | 62 | 1 | 50 | 510 | 1 | 120 | 1380 |

Using the analysis methods introduced in *The Art of Computer Systems Performance Analysis* [4], this experiment design can be considered as a 2*3*3 design. The factors are platform, image size and number of request per Planet lab node. The factors and their levels are listed in Table 2.

**Table 1.2: Factor and levels for Google APP engine study**

| Symbol | Factor | Level 1 | Level 2 | Level 3 |
|---|---|---|---|---|
| P | Platform | Google App Engine(GAE) | Traditional Web Server | |
| I | Image Size | 12kB | 350kB | 1MB |
| N | Number of Request per Planetlab Node | 1 | 10 | 100 |

Since the max/min in Table 1 is 1380, a log transformation should be performed for this case. The Analysis of Variance (ANOVA) is shown in Table 3.

From the results we get from Table 3, the RTT is highly affected by two factors, the platform and the number of requests per Planet lab node. Since the dominant effect, the number of requests per Planet lab node, explains 79.428% of variations while the second most significant effect, the platform, only explains 7.448% of variations, as a matter of fact, we can say that the cloud computing can get reasonable performance if the resources in the infrastructures are appropriately distributed.

**Table 1.3: ANOVA table of RTT for Google App Engine Study.**

| Component | Sum of Squares | Percentage of Variation | Degree of Freedom | Mean Square |
|---|---|---|---|---|
| y | 39.754 | | 18 | |
| y_bar | 22.925 | | 1 | |
| y-y_bar | 16.829 | 100 | 17 | |
| Main effects | 15.265 | 90.708 | 5 | 3.053 |
| P | 1.253 | 7.448 | 1 | |
| I | 0.645 | 3.831 | 2 | |
| N | 13.367 | 79.428 | 2 | |
| First-order interaction | 1.299 | 7.722 | 8 | 0.162 |
| PI | 0.305 | 1.814 | 2 | |
| PN | 0.669 | 3.978 | 2 | |
| IN | 0.325 | 1.930 | 4 | |
| Second-order interaction | 0.264 | 1.570 | 4 | 0.066 |
| PIN | 0.264 | 1.570 | 4 | |

**Table 1.4: Throughput result for GAE and traditional web host, data are collected from network-based measurement on cloud computing services [1]**

| Image Size | 12kB | | | 350kB | | | 1MB | | |
|---|---|---|---|---|---|---|---|---|---|
| # of req/Planetlab node | 1 | 10 | 100 | 1 | 10 | 100 | 1 | 10 | 10 |
| RTT for GAE(sec) | 50 | 36 | 32 | 275 | 220 | 175 | 250 | 260 | 135 |
| RTT for TWS(sec) | 42 | 22 | 26 | 120 | 70 | 65 | 75 | 65 | 60 |

Using the same method applied in the previous analysis, we can also get the ANOVA results in Table 5. Since the ratio of max/min is 11.9, we should try the log transformation as well.

**Table 1. 5: ANOVA table for throughput for Google App Engine**

| Component | Sum of Squares | Percentage of Variation | Degree of Freedom | Mean Square |
|---|---|---|---|---|
| y | 67.581 | | 18 | |
| y_bar | 65.408 | | 1 | |
| y-y_bar | 2.173 | 100 | 17 | |
| Main effects | 2.008 | 92.408 | 5 | 0.402 |
| P | 0.549 | 25.283 | 1 | |
| I | 1.329 | 61.155 | 2 | |
| N | 0.130 | 5.970 | 2 | |
| First-order interaction | 0.156 | 7.197 | 8 | 0.020 |
| PI | 0.115 | 5.278 | 2 | |
| PN | 0.018 | 0.837 | 2 | |
| IN | 0.023 | 1.081 | 4 | |
| Second-order interaction | 0.009 | 0.395 | 4 | 0.002 |
| PIN | 0.009 | 0.395 | 4 | |

Notice that the major effects for the network throughput study are the platform (explains 25.283% of the variations) and image size (explains 61.155% of the variations). This will lead to the conclusion that the throughput for Google App Engine and traditional web servers strongly depend upon the data size transmitted through the network. Compared to the effect of image size, the effect of the platform can be neglected. Therefore, it is possible that Google App Engine can get reasonable performance by wisely choosing the data size.

## 4. Analysis for Amazon Web Services

**(a)** Machine specification and Instances for Amazon Web Service

The Machine specifications and instances for Amazon Web Service and Camillus are given in the Table 6.The data are collected from *A Quantitative Analysis of High Performance Computing with Amazon's EC2 Infrastructure: The Death of the Local Cluster?* [2]

**Table 1.6: Table of machine specifications and instances for Amazon Web Service and Camillus, data are collected from A Quantitative Analysis of High Performance Computing with Amazon's EC2 Infrastructure: The Death of the Local Cluster? [2]**

| Specification and Instance Type | CPU | Memory(GB) | Disk(GB) | I/O Performance |
|---|---|---|---|---|
| M1.Small 32bit, 1 core | 1*ECU | 1.7 | 160 | Moderate |
| M1.Large 64bit, 2 cores | 2*ECU | 7.5 | 850 | High |
| M1.XLarge 64bit, 4 cores | 2*ECU | 15 | 1690 | High |
| C1.Medium 32bit, 2 cores | 2.5*ECU | 1.7 | 350 | High |
| C1.Xlarge 364bit, 8 cores | 2.5*ECU | 7 | 1690 | High |
| Camillus | 64-bit dual-CPU Intel Xeon E5345 Quad-Core | 16 | -- | High |

**(b) Metrics and parameter selection**

The metric for studying Amazon Web Service is chosen to be the memory bandwidth since nowadays applications consume a lot of memory bandwidth based on the data stored in the memory. The parameter is chosen to be the actions of the application which include four levels: copy, scale, and add and triad.

**(c) Performance Analysis**

The experiment result is shown in Table 7. All the experiment data are collected from *A Quantitative Analysis of High Performance Computing with Amazon's EC2 Infrastructure: The Death of the Local Cluster?* [2]

**Table 1.7: experiment result for Amazon web Service study.**

| | 1 Thread Bandwidth in GB/s | | | |
|---|---|---|---|---|
| Machine Specifications | Copy | Scale | Add | Triad |
| M1.Large | 2.058 | 1.777 | 1.868 | 1.725 |
| M1.XLarge | 2.551 | 2.394 | 2.434 | 2.178 |
| C1.Medium | 2.865 | 2.852 | 3.114 | 3.097 |
| C1.Xlarge | 2.849 | 2.840 | 3.126 | 3.120 |
| Camillus | 2.834 | 2.830 | 3.171 | 3.160 |

Notice that the observations in Table 7 can be considered as the paired observations for each level of the factor. Then using the method introduced in *The Art of Computer Systems Performance Analysis* [4], we can compare the performance differences between each Amazon Web Service instance and the Camillus. The analysis results are listed in Table 8.

**Table 1.8: Performance differences analysis for Amazon Web Service study**

| | Sample Mean | Sample Variance | Sample Standard Deviation | 90% Confidence Interval for Mean |
|---|---|---|---|---|
| M1.Large & Camillus | 0.285 | 1.062 | 1.031 | (-1.115,1.686) |
| M1.XLarge & Camillus | 0.152 | 0.376 | 0.613 | (-0.681,0.985) |
| C1.Medium & Camillus | 0.004 | 0.003 | 0.052 | (-0.067,0.075) |
| C1.Xlarge & Camillus | 0.003 | 0.001 | 0.034 | (-0.043,0.051) |

From the results we get from Table 8, we can say that the performances are not different for each of these four observation pairs since the 90% confidence intervals all include zero. In other words, the performances on the 1 thread bandwidth study for the Amazon Web Service and the traditional machine within 90% confidence interval are the same. If we properly construct the architecture of the cloud computing infrastructure, we can get reasonable performance compared with the traditional machines in the memory bandwidth aspect.

**5. Conclusion and Future work**

The experiment designs in this only a small amount of metrics that should be measured for the cloud computing infrastructure.

Cloud computing is one of the most innovative technique and redefine the way of communication. It allows the end users to store and load their data or do the complex computing task anytime anywhere with a single device which can access the internet.

In this, performance analysis on two popular cloud computing platform

By comparing them to traditional web servers is introduced. The analysis method in The Art of computer system performance analysis [4] is conducted. As we can see from the analysis above, for the study on the RTT of Google App Engine, the dominant effect is the number of request per Planet lab node which explains 79.428% of the variation while the second main effect is the platforms which only explains 7.448% of the variation. In the second experiment on Google App Engine, the network throughput is affected by the image size (explains 61.155% of variations) and the platform

(explains 25.283% of variations). This tells us that the platform is not a significant bottleneck in the RTT and throughput aspects of cloud computing. For the test on Amazon Web Service, by comparing one tread bandwidth of Amazon Web Service and Camillus, the results indicate that neither of these two platforms is superior to the other. In a conclusion, the cloud computing infrastructure can get quite reasonable performance compared to the traditional web servers depending upon the service delivered. This may provide a better insight on how to construct the cloud computing infrastructure and platforms.

**REFERENCES:**


1. Vinod Venkataraman, Ankit Shah, Yin Zhang, "Network based Measurments on cloud computing Services",http://www.cs.utexas.edu/~vinodv/files/cc-measure.pdf
2. Zach Hill, Marty Humphrey, "A Quantitative Analysis of High Performance Computing with Amazon's EC2 Infrastructure: The Death of the Local Cluster?", the 10th IEEE/ ACM International Conference on Grid Computing (Grid 2009). Oct 13-15 2009.
3. M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the clouds: A Berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.
4. Raj Jain, "The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling", Wiley, 1991,pp 203-220,pp 381-389
5. O. R. J. Porras and M. D. Kristensen. Dynamic Resource Man- agement and Cyber Foraging, chapter Middleware for Network Eccentric and Mobile Applications. Springer Press, 2008.
6. "httperf Wikipedia, the free "encyclopedia",http: // en.wikipedia.org/wiki/Httperf.
7. "Planetlab testbed" ,http://www.planet-lab.org.
8. B.-G. Chun and P. Maniatis. Augmented Smartphone applications through clone cloud execution. In USENIX HotOS XII, 2009.
9. M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies. The case for vm-based cloudlets in mobile computing. IEEE Pervasive Computing, (4), 2009.
10. R. Balan, J. Flinn, M. Satyanarayanan, S. Sinnamohideen, and H. Yang. The case for cyber foraging. In Proc. of the 10th ACM SIGOPS European Workshop, 2002.
11. R. K. Balan, M. Satyanarayanan, S. Park, and T. Okoshi. Tactics- based remote Execution for mobile computing. In Proc. of the 1st.
12. G. C. Hunt, M. L. Scott, G. C. Hunt, and M. L. Scott. The coign automatic distributed partitioning system. In Proc. of the 3rd Sym- posium on Operating Systems Design And Implementation, pages 187–200, 1999.
13. J. S. Rellermeyer, G. Alonso, and T. Roscoe. R-osgi: dis- tributed applications Through software modularization. In Proc. of the ACM/IFIP/USENIX International Conference on Middleware, 2007.