MOVING SOCIAL NETWORKING APPLICATIONS INTO THE CLOUD

A Thesis Submitted to the College of

Graduate Studies and Research

In Partial Fulfillment of the Requirements

For the degree of Masters of Science

In the Department of Computer Science

University of Saskatchewan

Saskatoon

By

RADHIKA RAMASAHAYAM

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science

University of Saskatchewan

Saskatoon, Saskatchewan

Canada

S7N 5C9

ABSTRACT

Social networking applications that are developed using traditional software and architecture have scalability issues. One way to overcome the high cost of scaling social applications is to use Cloud Computing (CC). There are various cloud computing platforms available. One very interesting CC platform is Google App Engine (GAE). This research focuses on using the “free” GAE as a way to re-implement existing social networking applications.

The research focuses on how to move social applications into the cloud and on the evaluation of their performance. The thesis investigates the GAE platform, and its features. The study shows how to re-implement a social networking application using GAE cloud with limited code approximately 600 lines and evaluates the scalability of the applications.

ACKNOWLEDGMENTS

I take this opportunity to acknowledge and extend my gratitude to the people who helped me in successfully completing my Master's Degree.

First, I would like to sincerely thank my supervisor Dr. Ralph Deters for his excellent guidance, encouragement, patience, and financial assistance. I thank my committee members Dr. C. Rangacharyulu, Dr. Ralph, Deters, Dr. Julita Vassileva, Dr. John Cooke, and Dr. Mark Eramian for their valuable suggestions and comments.

My special thanks to the MADMUC'ers for their friendliness and co-operation. In particular I want to thank our Graduate Correspondent, Ms. Jan Thompson who has been very helpful throughout my study here. I am grateful to the technical and office staff, professors and students of the Computer Science Department for their assistance.

Most importantly, I thank my parents and in-laws for their unconditional love and encouragement. I especially appreciate my most beloved husband Dr. Laxman Reddy Nadithe for his support and belief in me.

CONTENTS

LIST OF FIGURES

FIGURES PAGE NUMBER

LIST OF TABLES

LIST OF ABBREVIATIONS

AMI	Amazon Machine Image
API	Application Programming Interface
AWS	Amazon Web Services
CC	Cloud Computing
CSS	Cascading Style Sheets
CURD	Create, Update, Retrieve, Delete
DS	Data Store
EC2	Elastic Compute Cloud
GAE	Google App Engine
GFS	Google File System
GQL	Google App Engine Query Language
HaaS	Hardware as a Service
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IDE	Integrated Development Environment
IT	Information Technology
JDK	Java Development Kit
JDO	Java Data Objects
JDOQL	Java Data Objects Query Language
JPA	Java Persistence Interface
JSON	JavaScript Object Notations
JSP	Java Server Pages
JVM	Java Virtual Machine
MVC	Model View Controller
MySQL	My Structured Query Language
PaaS	Platform as a Service
PC	Personal Computers
PHP	PHP: Hypertext Preprocessor
PMF	Persistence Manager Factory

RDBMS	Relational Database Query Language
REST	Representational State Transfer
RIA	Rich Internet Applications
RPC	Remote Procedure Calls
SaaS	Software as a Service
SDK	Software Development Kit
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOC	Service Oriented Computing
SQL	Structured Query Language
SSTables	Stored String Tables
UDDI	Universal Description, Discovery, and Integration
URL	Universal Resource Locator
W3C	World Wide Web Consortium
WS	Web Services
WSDL	Web Service Definition Language
XML	eXtensible Markup Language

CHAPTER 1
INTRODUCTION

Service Oriented Computing (SOC) is a computing paradigm that utilizes services as fundamental elements. It supports rapid, low cost and easy composition of distributed applications in heterogeneous environments [1]. The SOC has evolved from legacy systems (in enterprises) that are linked together with business processes. Such systems contain code that is difficult to update and modify. Liu et al. [2] say that with the development of “services”, existing functionalities of the system can be combined with Web Services (WS) to build services that are ready to use and can be combined to create new systems without further modification.

As defined by Papazoglou et al. [1], the WS are used to develop applications that can communicate with each other over the Internet. The Simple Object Access Protocol (SOAP) is widely used protocol to exchange messages between service providers and users. The introduction of Web2.0 technology has increased the usage of the RESTful architectural style for developing services. The Representational State Transfer Protocol (REST) is an architectural style that guides development of applications based on Hypertext Transfer Protocol (HTTP) design principles. It is comparatively easy to develop and use applications based on REST. Thus many computing paradigms are proposing frameworks to develop, deploy, and maintain applications with minimum effort and better performance. One among them is the Google App Engine (GAE) that enables users to develop applications and upload them to the Google’s cloud. Using this framework the applications can be easily developed, deployed and maintained. As the demand for the application grows the applications can be scalable depending on our requirements.

The goal of this research is to study how to re-implement existing social networking applications using a cloud. The research work is organized as follows: Chapter two states the

problem description, Chapter three provides the literature review on WS, SOAP, REST, and popular the Cloud Computing (CC) approaches. Chapter four describes the CC and its features. Chapter five discusses the architecture and implementation of the experiments. The application testing and evaluation is described in chapter six. Chapter seven discusses the conclusions and the future work of this research.

CHAPTER 2
PROBLEM DEFINITION

WS were introduced to interact with the applications that are diverse in nature, build new applications and allow them to communicate with each other over the Internet. Thus the WS enable rapid application development with low costs based on the principles of SOC [1]. Menace [3] explained in his work, that SOAP is a widely used protocol to exchange messages in the form of XML regardless of the operating system or the computing environment. The interaction between users and WS providers is in the form of XML based SOAP messages which tend to be long and require parsers on both sides thereby reducing the performance of the applications. Litoiu [4] discussed about the client server based application model using WS, in which the server's performance is affected due to the scalability issue with increase in the number of users.

The introduction of Web2.0 technology and the RESTful architectural style has reduced the performance and scalability problems. REST is a stateless protocol that can handle interactions based on the HTTP verbs (PUT, GET, POST, and DELETE). Calcote [8] mentioned, the HTTP verbs are used to perform (Create, Retrieve, Update, and Delete) (CURD) operations by the WS designers. The table 2-1 describes REST verbs and its equivalent CURD semantics.

Table 2-1. Table showing REST verbs and corresponding CURD operations

REST Verbs	Function	CURD Operation
PUT	Replaces the entire URL with the content sent	Create
GET	Lists the URL and other details of it	Retrieve
POST	Updates the resources on the server with one or more entries	Update
DELETE	Deletes the entire content	Delete

According to Amazon, 80-85% of WS are REST based and the performance of the REST based applications is 6 times faster than the SOAP WS and do not require any additional security and standards [5][6]. Pautasso et al. [7] pointed out the services built using REST are light-weight, and scalable and that they can be used with an ordinary browser [7].

Traditionally, development of web applications in an organization incurs huge costs due to development of the applications, maintenance with the need to buy servers hosting their software, applications to rent servers to improve scalability. Usually this process is time consuming for the organization to maintain resources thereby producing resultant systems that are not reliable.

Wikipedia [9] states that “Cloud Computing (CC)” is a mechanism to cut down costs of hosting, and scaling an application. CC is a mechanism to improve reliability, security, sustainability, and location independence. CC incorporates three aspects, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). The SaaS model focuses on hosting the application by a service provider or vendor and making it available to its users over the network. SaaS is becoming popular with the support of existing technologies like the WS and the Service Oriented Architectures (SOA). PaaS delivers a computing platform as a service. PaaS provides the facilities required for developing the complete life cycle of applications, starting with building them to delivering web applications [10]. Examples of them are salesforce.com which provides the platform to build and deploy enterprise based applications. The Google App Engine (GAE) and Microsoft’s Azure are providing foundations upon which users can build more scalable, and robust web applications. The GAE provides a platform to build and host web applications on Google’s infrastructure. It uses multiple servers depending on the requirements to run applications and store data. It automatically adjusts the

number of servers to run the applications, depending on requests [12]. Microsoft's Azure provides a specialized operating system called "Windows Azure" that runs applications hosted on Microsoft's datacenters by managing resource allocation and storage. Microsoft [13] says that Azure uses Windows 2008 server and Hyper-V to provide virtualization. The IaaS model delivers computing environments to run the applications. Examples of the IaaS are Amazon's Elastic Compute Cloud (EC2). EC2 is among the list of Amazon's Web Services (AWS).The EC2 allows customers to rent computers to run their own applications and provides scalable deployment of applications with WS interface to create virtual machine instances. EC2 uses Xen [11] virtualization mechanism to create instances in three different sizes; small, large, and extra-large.

This research focusses on re-implementing social legacy applications in a way that they are scalable, and robust.

- This study focuses mainly on how to re-design and move a social networking application into the cloud
- It investigates possible design architectures within the cloud
- To evaluate in terms of its scalability?

This research focuses on problems with existing social web applications, architecture and implications of the cloud platforms, design options to enable rapid and easy development of social web applications within the cloud, and to evaluate the performance of the applications in the cloud.

CHAPTER 3
LITERATURE REVIEW

This chapter presents WS, their importance, types of WS, and various CC approaches.

Web Services

Vinoski [14] states that the underlying strengths of the WS are to integrate with applications that are diverse and heterogeneous in nature ranging from varied applications, operating systems, and hardware platforms. According to Gartner [15], "Web Service (WS) is a loosely coupled remote procedure call that would replace today's tightly coupled Remote Procedure Calls (RPCs) which require application and protocol specific Application Programming Interface (API) connections."

Features of the WS include platform independent technologies that can ease delivery of network based services over the intranet or the Internet. They can integrate personal computers (PCs), hand held devices, databases, and networks into one computing platform via web browsers so that services are run on web-based servers [15]. Vaughan-Nichols [15] [16] defines WS as a mechanism to utilize the existing IT infrastructure and allow the organizations to wrap their existing legacy applications in a standardized, consistent, and reusable format so that the companies can collaborate with their business partners to connect their internal applications in a cost effective manner. Dave Spicer of Flamenco Networks [16] says that, "Adoption of XML as a standard lead to the development of WS". The Extensible Markup Language (XML) is the most important WS standard basis for many other WS standards.

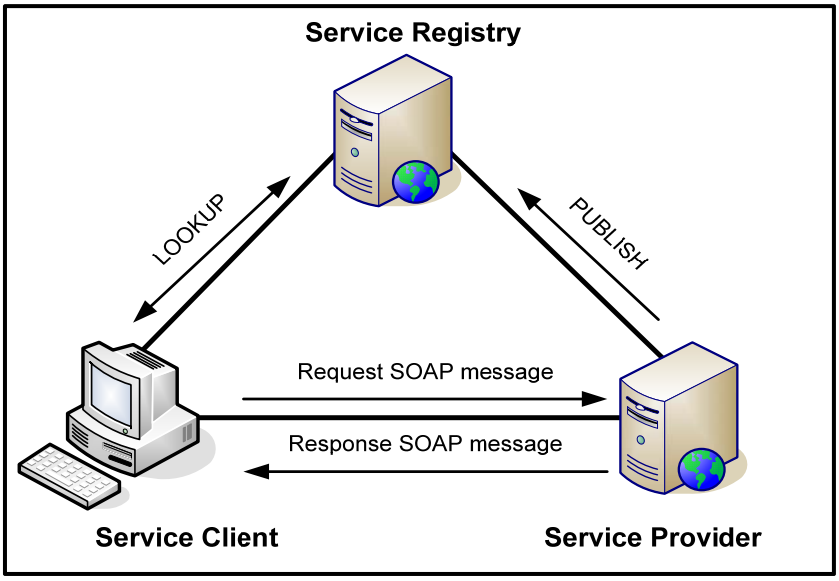


Figure 3-1. Service Oriented Architecture

Papazoglou [17] introduces the concept of WS as “A software system available via a network such as the Internet to complete tasks, solve problems, and conduct transactions on behalf of users or applications”. To accomplish a task, WS are used for discovering and invoking network available services rather than building new applications. SOA helps WS framework to implement publishing, discovering, and binding. According to Curbera et al. [18] the activities are identified by three different areas, the communication protocols, the service descriptions, and the service discovery. The Communication protocol (SOAP) enables communication among the WS, Web Service Description Language (WSDL) provides a description of the WS and the Universal Description, Discovery, and Integration (UDDI) provides the list of WS in the registry with their descriptions. SOAP and REST are the communication mechanisms for the WS. SOAP is a protocol that can be used in different architectures while the REST is an architectural style [19].

SOAP

To solve the problems of proprietary systems running on heterogeneous platforms, the WS popularized SOAP, an XML based communication protocol used for exchange of messages between computers regardless of the underlying operating systems, programming environment or object model framework [17]. SOAP allows programs to communicate using HTTP and XML documents [15]. SOAP sends an envelope containing address, header, and a body in the form of an XML document to the service. The services are described in the UDDI registry. WSDL provides a description of the services in terms of what a service does i.e., its operations, where it resides, i.e., details of the protocol specific information and how to invoke it, i.e., data format and the protocols necessary to access the service's operations in the online XML registry based on the UDDI protocol. The UDDI protocol allows companies to publicly make available the WS on the Internet or corporate networks [15] [20].The SOAP based services connect service providers and requesters through APIs in the WSDL which can also be used for invoking a component on the remote machine. WSDL separates the interface from the implementation and the interface must be defined in terms of input and output messages it supports for each operation. The service is later bound to an implementation at a particular location using a port and binding [20]. Shi [21] points out that SOAP uses serialization and deserialization of objects to translate application specific languages to SOAP protocols.

The Figures 3-2 and 3-3 given below are examples of SOAP request and response messages [22]. The SOAP envelope request consists of a header and the stock name. The price of that stock is responded back in the response envelope shown in the figure 3-3.

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
 <m:GetStockPrice>
  <m:StockName>IBM</m:StockName>
 </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

Figure 3-2. SOAP Request [22]

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
 <m:GetStockPriceResponse>
  <m:Price>34.5</m:Price>
 </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

Figure 3-3. SOAP Response [22]

Table 3-1. Advantages of SOAP

	Advantages of SOAP
1	SOAP is a communication protocol that allows the exchange of information regardless of the underlying hardware or computing environment [17].
2	The application and data integration is easier as the client needs to know only the description of the service in WSDL and does not need to know how it is implemented and how the data is stored.
3	SOAP is versatile in design as a client can combine data from multiple WS and present the user with updated information without affecting the service.
4	Code reuse is another important feature. A service can be used by multiple clients all of them employed to serve business functionalities.
5	It allows creating highly customized applications for integrating applications that are inexpensive.

In spite of the above mentioned advantages of SOAP, it has some drawbacks which led to the introduction of an architectural style (REST) for developing web applications by R.T Fielding.

Table 3-2. Disadvantages of SOAP

	Disadvantages of SOAP
1	SOAP based systems are often tightly coupled [20] [25].
2	Introduction of Web2.0 technology has increased the complexity of developing web applications using XML [23].
3	SOAP does not provide secure environment for delivering messages to their destinations. However, secure protocols like S-MIME, HTTPS, and Secure Socket Layer (SSL) provide security to directly interacting parties [2] [6].

4	SOAP based protocols do not support adhoc application integration known as “Mash ups”.
5	XML and SOAP are too verbose and thus it affects the performance of the interacting systems [24].
6	As the number of interfaces increase, the complexity of the systems increases leading to SOAP-RPC services making them not interoperable [25].

REST

R.T Fielding defines REST in his thesis dissertation [26] as an architectural style to describe a network of resources that are loosely coupled. According to him the term *Architecture* is used to design a system with a set of properties that forms a superset of systems requirements. The Architecture embodies both functional and non functional components like arrangement of components, data within a system, reusability of components. Fielding et al. [27] similarly defines “*Styles*” as a mechanism to categorize the architectures and defining common characteristics.

The acronym of REST stands for “*Representational State Transfer Protocol*”. Fielding [26] proposed the motivation for using the Representational State Transfer is the design principles and characteristics of HTTP. HTTP uses a concept of Universal Resource Locator (URL) to transport data between resources. Xu [28] states that, it treats entities in the world as resources connected to each other and supports the Resource Oriented Architecture (ROA). REST uses the “*resource identifier*” to identify components involved in interactions, “*representation*” to capture the current or intended state of a resource and to transfer the representation between components. REST has different types of connectors for component

communication, enhancing simplicity by separation of concerns and hiding the underlying resources and their communication mechanisms [26]. REST is not a standard, it is a style drawn from many pre-existing distributed paradigms, communication protocols and software engineering fields. According to Costello [29], REST is an architectural style that helps in designing WS based on the standards like HTTP, URL, XML, HTML, GIF, JPEG, text/xml, text/html, image/gif, image/jpeg, etc. Separation of clients user interface from its data storage improves the portability of user interface across multiple platforms and increases the scalability of the applications [26]. The second aspect is stateless communication between client and server. Each request from a client to server must contain all the information necessary for the interaction between client and server. Thus it leads to an increased reliability and scalability of applications. The third aspect that adds to the efficiency of REST is client-cache-stateless-server. If the data is cached for a particular client request it can be stored on the client side for any equivalent requests later. Advantages of client cache include reduction of latency, improved efficiency and scalability due to partial reduction of few interactions. REST's uniform interface between its components makes it distinguishable from other network based styles due to its simplicity and improved visibility of interactions. It is an approach to get information from the website by reading the information in the form of XML that describes the content. RESTful WS gained popularity in the development of distributed applications based on HTTP. These services can be easily integrated in to various applications such as mashups [2] [30] which is complex to create them using SOAP-RPC style.

Advantages of the REST based style, which allow it to be used widely in enterprise architecture and business applications are shown below.

Table3-3. Advantages of REST

	Advantages of REST
1	REST is simple, minimal use of tools; easy to build, maintain, and extend applications.
2	Requires less time to build a client for RESTful WS and can be tested with a normal web browser [7] [31].
3	REST allows usage of XHTML/XML formats to allow dynamic communication between the interacting parties. This increases adaptability and loose-coupling between the applications [28].
4	REST sends data which represents their state across network using resources [19].
5	Communication between client and server do not require protocol conversions. A business process represents all resources as URLs. These resources can be manipulated as PUT, GET, POST, DELETE operations that increases interoperability [28].
6	REST only expands those portions of the architectures that are needed for Internet-based distributed hypermedia interactions. The existing protocols fail to get the details required for protocol interaction and currently used semantics can be replaced with an efficient form without changing the architecture [27].
7	Since each resource has its own representation, scalability is improved by minimizing network delays and latency. REST based systems provides light weight access to operations due to its limited number of operations and unified addressing schema [19].
8	REST uses all types of data for representing resources such as HTML, GIF, PDF files. It recommends usage of standards like URL's for addressing, HTTP methods for communication of messages, MIME types, XML, XHTML, HTML and PNG for representation of data formats. The standards used by REST are all web standards [32].

Table3-4. Disadvantages of REST

	Disadvantages of REST
1	There is no common standard accepted for the REST service description [29] [5].
2	REST requests especially GET do not handle URL's that are lengthy (i.e., above 4KB)[7] [33].
3	REST style does not cover all WS standards like Transactions, Security, Addressing, Trust, and Coordination.
4	REST does not have any widely accepted specifications like WSDL. Developers have to use the XML because there are no tools and IDE's that generate it [31].

Pautasso et al. [7] compared SOAP and REST and concluded that in spite of the above mentioned disadvantages; REST is preferred by WS developers for its simplicity in the design of interfaces and developing resources, scalability, usage of intermediate components to reduce latency.

Buyya et al. [34] describes that with the rise of SOA, the essential basic computing services are made available to users depending on their requirements. The consumers can pay service providers for the utility services they used. The latest computing paradigm that emerged into the world of computing is the "Cloud Computing" which promises reliable services to be delivered through data centers, built on virtualized compute storage clouds by using the WS developed using SOAP or REST.

Cloud Computing

Traditionally, the development of web applications in an organization starts at the infrastructure level at which an organization creates its own websites. Initially, a small group of people interact with the website. As the demand for applications increases, organizations need to buy servers hosting their website or rent it to host on other severs to improve its scalability. Usually at this level organizations spend lots of money, time and resources to host a website and to keep it running all the time.

Hayes [35] summarizes that technology advancements in the past 50 years have changed vastly with the human needs. Time-sharing machines which had a central hub and individual users at the terminals communicated with the central site using telephone lines for computing and later personal computers appeared which focused on decentralization of data and programs and gave rise to client server model.

Armbrust et al. [36] states that today computing is offered similar to utility services like electricity, gas, water, and telephone where users can access the services based on their requirements. It is available to users with less costs and minimum delay. The users accessing the services need not know where the servers are located, how the services are delivered, or how to maintain the servers. Several computing paradigms have promised the vision of delivering utility computing and these include Cluster Computing, Grid Computing, and CC [34]. Among these CC has recently emerged where enterprises and users are able to access applications on demand. CC has developed a mechanism to cut down costs of hosting, scaling an application, improving reliability, security, sustainability, location independence. Thus the importance of CC is on developing the software and making it available as a service rather than running it on individual computers [34].

Armstrong et al. [36] and Barnatt [37] explain the term “Cloud Computing” as “the applications delivered as services over the Internet”. The hardware and software in datacenters provide services which are called as the SaaS [37]. In this paradigm a client computer on the internet can communicate with many servers at the same time while some of them are exchanging information among themselves [35]. The aim of computing in a cloud is to concentrate computation and storage in the core, where high performance machines are linked by high-bandwidth connections and all these resources are carefully managed.

Cloud Computing Models

IaaS, PaaS, and SaaS are three forms of CC [9]

1. ***The SaaS*** model focuses on hosting the applications by a service provider or vendor and making it available to its users over a network. The SaaS model is becoming popular with the support of existing technologies like WS and SOA [38]. It is different from other software models by avoiding the need to purchase or maintain computer hardware and infrastructure related to run the application [39]. The SaaS model generally prices the applications on a per-user basis or per-business basis. The revenues for the software vendors are initially lower than the traditional software license procedure but it is a recurring process. It is predicted to be similar to maintenance costs for the licensed software [39]. Benefits of SaaS include easier administration, limiting the infrastructure and installation, compatibility of the software among all users, automatic updates, global accessibility, and allowing easier collaboration with other parties. Examples of SaaS are the Google’s Gmail which scales to a large measure, and Fortiva’s email archiving service which addresses the need for email e-discovery [39].

2. ***The PaaS*** delivers a computing platform as a service. It provides all facilities required for developing a complete life cycle of applications from building the web application to delivering application [40]. Using tools developers build applications and deploy them without the need for specialized administration skills. The benefits of the PaaS model are the ability to develop, deploy and maintain the web applications oneself by overcoming problems with traditional development where there is a backend server development, front end client development and the web site administration [10]. Examples of PaaS model are force.com from the Sales Force infrastructure, Microsoft Azure, and the GAE from Google based on Python and Java languages.
3. ***The IaaS*** model uses the equipment leased by the service provider to support operations, storage, hardware, servers and the networking components. In this model, the service provider owns the equipment and is responsible for maintaining and running it. The resources can scale up and down based on the requirement and thus users pay for the services based on the consumption levels [41] [42].The IaaS model is in the form of a virtualized computing environment in which users can deploy their applications in a virtual image locally and then execute it within a remote environment without worrying about the underlying network infrastructure or the server. Examples of IaaS are BlueLock which is used to configure servers, storage and virtual machines, and EC2 [42].

Popular Cloud Service Providers

Cheow states [43] that, CC draws attention from experts in technology. The research studies done by the Gartner company in the year 2008 says that CC can be used for both large and medium scale companies. With the popularity of the CC, “Evans Data” [44] conducted a survey with over 400 software developers about their perceptions of leading CC vendors and

providers. Developers rated them based on completeness of offering, ability to execute, and their capabilities such as security, scalability, reliability, and cost to value. Among the top list of companies are EC2, GAE, IBM's cloud and Microsoft's Azure.

- ***Amazon*** provides a WS called EC2. EC2 allows users to purchase computer processing power online. It provides a virtualized environment for hosting the instances of servers and creating new server instances upon requirement.

  > *Amazon defines **instance** as a predictable amount of dedicated compute capacity that is charged in instance-hour* [11].

  Amazon presents the virtual server instances created to the user with the same degree of access as the administrator would have to their servers [45]. It is flexible because users can choose the configuration of their instances as small, large, and extra large. Users can create and destroy multiple virtual server instances upon requirement. Amazon creates the server instances by launching Amazon Machine Images (AMI) that contains application, data, and configuration settings that the servers need. The AMI can be created from scratch or using the pre-configured templates [37]. Servers are hosted on different geographical areas, if one server goes down another one can be used thereby making service reliable. Amazon charges its customers based on the instance hours. Currently it is $0.10 per instance hour. Depending on the amount of data moved in and out of Amazon's network, the charges vary between $0.10 to $0.18 per gigabyte [45]. The Amazon Web Services (AWS) [59] also provides a service called the Cloud Front for content delivery. It delivers the streaming content using the global network of edge locations. The requests for a particular object are routed to the nearest edge so the content is delivered with best possible performance. Miller[60] says that Amazon's data centers

(that help in caching of web content) are located in Ashburn (Virginia), Dallas (Fort Worth), Los Angeles, Miami, Newark (New Jersey), Palo Alto (California), Seattle, St. Louis, Amsterdam, Dublin, Frankfurt, London, Hong Kong, Singapore, and Tokyo.

- ***Google*** provides a platform to build and host web applications to Google's infrastructure known as the "Google App Engine". It uses multiple servers depending on requirements to run applications and store data. It automatically adjusts the number of servers to run applications, depending on requests [12]. The GAE provides a software environment centered on Python, and the Java programming languages using Bigtable for distributed storage [35]. The features and functionality of the GAE will be provided in chapter 4. Google's data centers are distributed geographically across the world to scale the applications. "Data Center Knowledge" [61] published the locations of the Google's data centers throughout the world. Google has 19 data centers in United States including the 3 which are under construction, 12 in Europe, and 1 in Russia. The data centers are in Mountain View, Pleasanton, San Jose, Los Angeles, Palo Alto in California, Seattle, Portland, the Dales in Oregon, Atlanta, Reston, Virginia Beach in Virginia, Ashburn, Houston (Texas), Miami (Florida), and Lenoir (North California). The international locations where data centers are located are Toronto, Berlin, Frankfurt, Munich, Zurich etc. Google reports that it spends $600 million dollars for each of the four new data centers with the expenses from computers to the construction [61].

  The "Royal Pingdom" blog [62] writes that Google's focuses on the following criteria in choosing the data centers

  - Cheap electricity.
  - Green energy with its focus on renewable energy resources.

- Closeness to rivers and lakes for cooling the data centers.
- Large land areas for its security and privacy.
- Distance to the other Google data centers for its operations, and tax incentives.

The figure 3-4 shows the location of the Google's world wide data center locations.

Figure 3-4. Google's Data Centers [62]

- ***Microsoft*** provides the Azure service platform for its customers to develop, deploy and manage distributed web applications. It supports existing web technologies like ASP, Internet Information Service, and Integrated Development Environments (IDEs) to create, and deploy the applications. It hosts the applications and storage of information through its datacenters [46]. Applications for Azure are written in .NET libraries and compiled to common language runtime, which runs on a platform independent environment. Thus, it can be viewed as an intermediate between frameworks like GAE and virtual hardware provided by EC2 [36].Google launched a similar service for

creating and uploading web applications on Google's framework which I will discuss in chapter 4.

Summary

Research on WS has enabled us to develop applications at a faster rate to overcome the problems with middleware based legacy applications that are crucial for an enterprise. Thus the introduction of the WS made web applications available on the Internet. Further web applications on the Internet developed with SOAP based technologies provided quick access to the applications, allowing applications to be accessed from any platform. But the developed applications were tightly coupled with each other and verbose. Examples are the services developed using SOAP. This lead to the development of applications based on the HTTP rules that govern the Internet. Roy Fielding's research suggests that RESTful approaches are used for developing scalable and reliable web applications that will be proved based on the experiments discussed in chapters 5 and 6.

The RESTful development of WS is currently followed by many organizations and enterprises because of its loosely coupled nature. It is easy to develop and maintain because it is based on the Internet based protocol using simple verbs, navigation through the resources is fast. CC has emerged recently that focuses on reduction of expenses on resources and thus the application can be developed in a pay as you go manner. Then the web applications can be uploaded to the cloud and maintained without any issues on the enterprise side. CC on the other hand has emerged as a solution to cut down the enterprises expenditures but there is a limited literature about how to use it.

The SOC uses the WS as a model to integrate business applications. With the growing demand for Internet and the web applications, I predict that RESTful WS are more appropriate

for designing applications that are suitable for business transactions. Thus I propose that CC is a model based on the REST based technologies to design applications that are reliable, scalable and providing virtualized computing and storage.

Among the CC paradigms I investigated (PaaS, IaaS and SaaS), I choose the PaaS model as this model provides the support for developing complete life cycle of applications. The PaaS model provides an environment to develop rich social networking applications that answers my research goal to move an existing social networking application into the cloud. There are currently many PaaS models that provide a development environment. GAE provides a "free" and attractive framework to the users to develop applications that can be uploaded to the cloud. Also the GAE uses the existing RESTful WS that are compatible to the Internet and use Web2.0 principles. The GAE scales enormously and is available to its client requests from any geographical area. Thus in chapter 4, I will discuss the GAE platform features, advantages and the implications of the GAE, its data storage the Bitable that enables scalable development, and address the key challenges.

CHAPTER 4
GOOGLE APP ENGINE

This chapter discusses the GAE. The three cloud models (SaaS, PaaS, and IaaS) discussed in chapter 3 provide different types of cloud services. Among them, based on the literature review I choose PaaS. The PaaS services I choose are provided by Google. Amazon differs from Google in terms of the services it provides.

In this research, I look into how to migrate legacy social networking applications using the "free" GAE framework. The applications developed using the GAE are scalable and reliable when compared to the existing social networking applications like "Our Wise Tales" [47] which is developed using the Content Management System (CMS) Drupal by Zina Sahib and Dr.Julita Vassileva, The NSERC/Cameco Chair of Women in Science and Engineering at the Prairies.

Features of Google App Engine

Google launched a service known as GAE in April 2008, which allows developers to run web applications on Google's infrastructure. The GAE applications are easy to build, maintain and scale with increased traffic and data storage [48]. GAE provides a new approach without dealing with web servers and load balancers but instead deploying the application on the GAE cloud by providing instant access and scalability shown in Figure 4-1. Applications can be developed using several programming languages. The languages supported are the Java standard technologies using the Java Virtual Machine (JVM), and the Python run time environment and libraries. Applications are developed using WS, based on the JVM or the Python libraries using the GAE Data Store (DS). Further they can be uploaded to the GAE cloud so that users can access them using a browser as shown in the Figure 4-1.

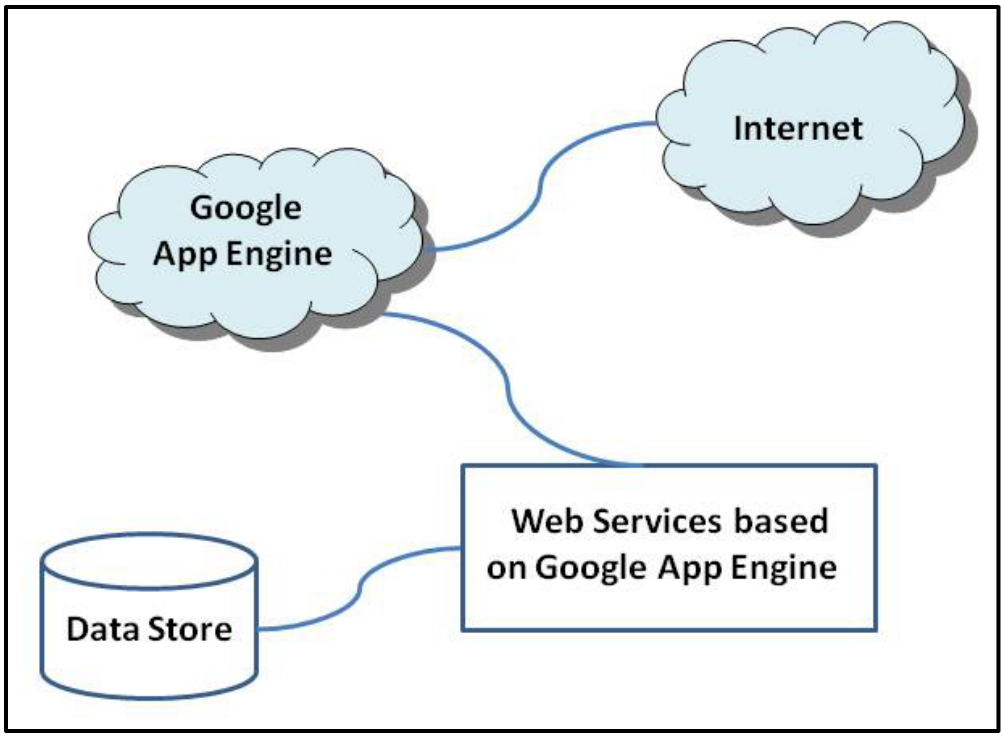


Figure 4-1. Architecture of Google App Engine

The GAE supports Python and Java libraries, and the Python and Java runtime environment. Users upload their applications and access them by using the free domain name“appspot.com” or their own domain [46] [48]. Google provides many cloud services like Gmail, YouTube, Spreadsheet, Word Processing etc.

GAE has some limitations as the applications are run with limited access to the underlying operating system. The advantages of the GAE are based on its ability to scale the applications which is mostly dependent on the data storage using the Bigtable. The pros and cons of using the Bigtable are discussed in the architecture of the GAE.

Table 4-1. Limitations of the GAE

	Limitations of GAE
1	If an application receives a web request, a response must be given within 30 seconds. If the request takes too long, process is terminated and server returns an error to the user.
2	GAE can return a maximum of 1000 query results each time. It can read information from a file but cannot upload data to the file unless it is specified within the application. It can only connect to the DS [48].
3	Python as a language supports extensibility but App Engine does not support code written in C or any other languages. Python environment provides rich APIs for DS, Google Accounts, and URL fetch and email services [49].
4	These applications can be run only with the Internet connection and using the URLs and APIs. Other computers can connect to them using HTTP or HTTPS on specified ports.

In spite of the disadvantages, it has many advantages that make it popular and promotes wide spread usage:

Table 4-2. Advantages of the GAE

	Features of GAE
1	GAE provides efficient and dynamic web application execution even under heavy loads and high data usage.
2	It provides automatic and on demand traffic and load balancing for the application by distributing it across multiple servers where each application has its own sandbox independent of the other applications to reduce resource conflicts [48].
3	GAE makes it easy to build web applications by providing a framework called Web App.
4	It provides a persistent storage system to perform transactions and queries.
5	It provides APIs for authenticating requests and sending emails to Google Accounts.

6	It also includes the Django [48] web application framework. Uploading third party libraries with the application is supported only if they are implemented in Python.
7	The GAE does not cost anything for 500MB of storage and 5 million page views per month is free [48]. Users can set maximum daily budget and allocate billing for each resource accordingly.

Architecture of the Google App Engine

The GAE Software Development Kit (SDK) provides Java and Python programming languages. The languages have their own web server application that contains all GAE services on a local computer. The web server also simulates a secure sandbox environment. The GAE SDK has APIs and libraries including the tools to upload applications. The Architecture defines the structure of applications that run on the GAE. Further the description about the architecture based on Python and Java languages is given in below sections.

Python

Python was the first one among the languages supported by the GAE. It was released in 1991, by Guido Van Rossum at National Research Institute for Mathematics and Computer Science (CWI), Netherlands. It is an interpreter based, general purpose programming language used for developing web applications [46]. Google has been using Python as one among the three languages used on production servers for system administration tasks along with C and Java. Python is used by Google for running automated tests, building and packing systems, pushing code to servers and some applications that are user visible like Google Groups and code.google.com.

The GAE allows implementation of applications using Python programming language and running them on its interpreter. The runtime environment for Python supports version 2.5.2.

The GAE provides rich APIs and tools for designing web applications, data modeling, managing, accessing apps data, support for mature libraries and frameworks like Django [48].

The main characteristics of GAE are its Bigtable or DS, configuration file app.yaml and how it serves an application [46].

Bigtable. The Bigtable is Google's distributed storage system for managing structured data and is being used to power search indexes and Google Earth. The Bigtable is a tabular NoSQL database that is designed to reliably scale to petabytes of data and thousands of machines. It is a sparse, distributed, persistent, multi dimensional storage map [50]. It is generally referred to as a "**map**" indexed with row key, column key and a time stamp.

According to the Wikipedia [64], a map is "*an abstract data type composed of collection of keys, and a collection of values where each key is associated with one value*".

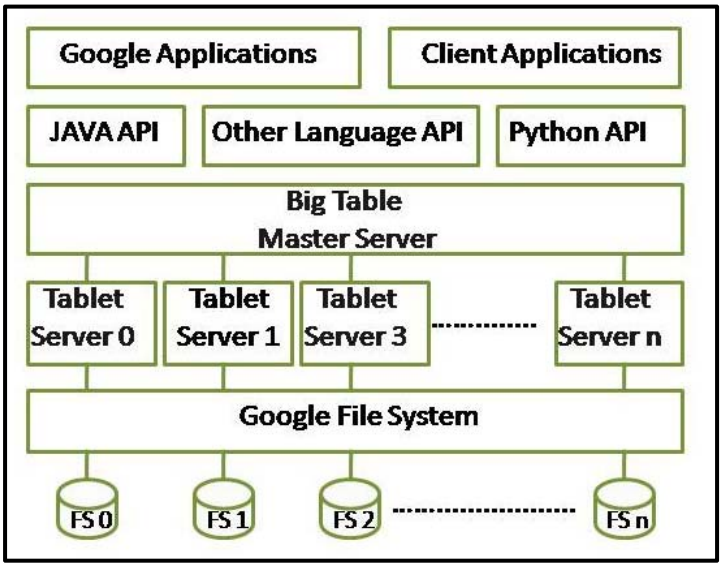


Figure 4-2. Bigtable Architecture and application development services [63]

Figure4-2 shows the Bigtable architecture and how it relates to the application services. Cuirana [63] defines the Bigtable has a master server that coordinates the large segments of a logical table called "tablets". The tablets are split across a row with an optimal size of 200MB

per each tablet for optimization purposes. The table contains rows and columns and each cell has a time stamp. So there can be multiple copies of the cells with different time stamps as shown in Figure 4.3. Chang et al., describes an example slice of a table for storing web pages in Figure 4-3. In the figure the row name is revered URL, and the contents column contains page content, the anchors column contains text of the anchors that referenced the page. The CNN's home page is referenced by both the sports illustrated and the MY-look home pages, so the row contains column names anchor: cnnsi.com, and anchor: my.look.ca. Each cell has one version so the column has 3 versions with different time stamps t_3, t_5, and t_6.

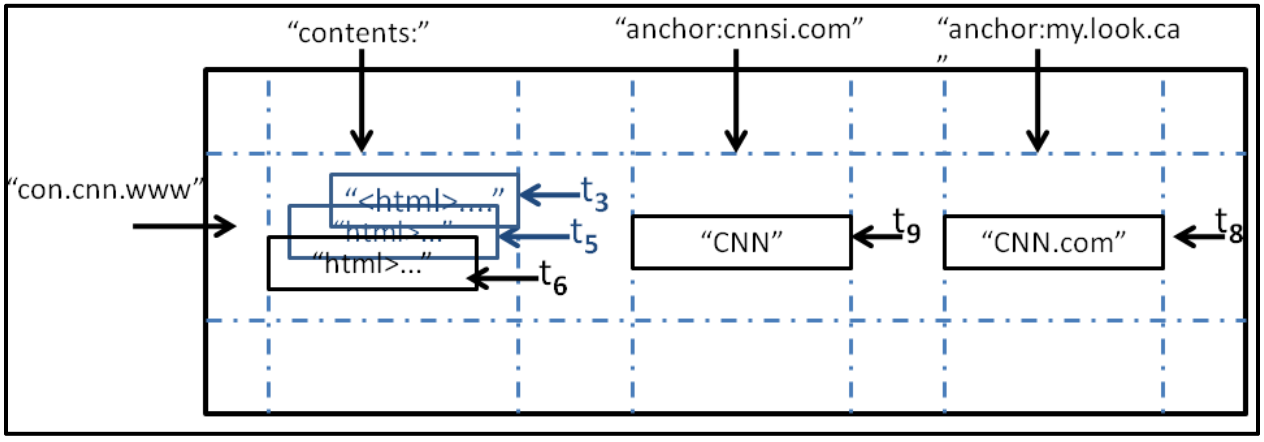


Figure 4-3. An example of a slice in a Webpage [50]

In order to manage the tables each table is split at a boundary and saved as a tablet. According to Hitchcock [66], the tablets are of fixed size (200MB) and each machine stores 100 of them in the Google File System (GFS). This setup allows load balancing by distributing the load to another tablet when one tablet receives lots of requests. It allows faster rebuilding when a machine goes down other machines take one tablet from the machine that is down so the load on each machine is very low. Tablets are stored on systems as immutable Sorted String Tables

(SSTables). The Google's SSTables provide a persistent map from keys to values where keys and values are arbitrary byte strings. It allows looking up the key value pairs by using operations.

The GAE allows usage of the Bigtable in applications through the DSAPI [46]. Batty mentioned in his blog that Barry Hunter [65] states that, "Bigtable and the GAE DS are not same". The DS is built on top of Bigtable which is built on top of GFS. The GAE does not allow access to any external databases like SQL. The DS is the only database GAE supports for logging and storing data, including session data. It uses slightly different terminology inherited from the Bigtable. The Bigtable can be defined as a huge spreadsheet with unlimited number of columns and in the form of a string.

The DataStore API. The DS is responsible for the scalability of the GAE applications. The structure of the applications enables them to distribute the requests across the servers which should be compromised with relational databases. Unlike any relational database the GAE DS can create an infinite number of rows and columns that scales by adding servers to the clusters.

In the DS, tables are called "models" and are represented in classes. Records are called "entities" and are instances of the model, columns are called "properties" and are attributes of models or entities [46]. To access the DS we have to define a model class with some properties, then create entities and store them in database. Later queries can be run to retrieve the entities. The model class can be created by sub classing db.model. The GAE provides a variety of property types from strings and integers to Boolean, date/time objects, list objects, phone numbers, email addresses, geographic points like latitude, longitude etc.

The GAE allows queries to be made using Bigtable as a database from its services using the Google App Engine Query Language (GQL) or Java Data Objects Query Language (JDOQL). All the data is being stored in the cloud which could be at any location on Google's

servers. If data is to be stored on an external data base which is locally installed on our machine Google imposes strict constraints due to security issues which can be a potential problem if organizations put their secure data on servers located in remote locations.

Configuration File: app.yaml. The app.yaml file is a platform neutral and human readable file for representing data. It is created as an alternative to XML to represent structures in programming languages like lists and dictionaries. “Key: value” syntax represents items in a dictionary and “-“represents elements in a list.

The file represents a dictionary with 5 key elements. They are application, version, runtime, api_version and handlers [46]. The structure of the app.yaml file is shown in Figure 4-4.

The first key is application, it can be any name when run on a local server. But if it is uploaded to the Google’s server, then the key application value must be the Application ID value. The second key is “version” which is used to specify version number of application. Google uses “MAJOR.MINOR” format to represent application numbers. MAJOR version is the number user sets and MINOR version is nth upload of that version. The GAE saves last upload for every MAJOR version, and one among them can be chosen as the current one. For the third and fourth keys runtime and api_version are specified as Python. Newer versions of API will be available in future. Handlers specify mapping of URL patterns. Handlers are different key values which can be a static file, script file or a static directory.

```
application: application ID registered with App Engine
version: 1
runtime: python
api_version: 1
handlers:
    - url: /.*
    script: name of the .py file
```

Figure 4-4. App.yaml File

How the App Engine serves applications. Each application has an app.yaml file which tells how to handle URL requests. GAE provides a simple framework called webapp that helps to organize code. When a web browser sends a request to the Google's cloud it chooses a server near the users location, instantiates the application if it is not running and processes the users request. Therefore the cloud meets the demands by creating the instances when required and deletes them when they are not used [46].

Java

The GAE provides tools and APIs required for the development of web applications that run on the GAE Java run time. The application interacts with the environment using servlets and web technologies like Java Server Pages (JSPs) which can be developed using Java6. The GAE environment uses Java SE Runtime JRE platform 6 and libraries [48] which the applications can access using APIs. Java SDK has implementations for Java Data Objects (JDO) and Java Persistence (JPA) interfaces. To exchange email messages with GAE, it provides the GAE mail service through the Java Mail API. Support for other languages like JavaScript, Ruby, or Scala is also provided by GAE with the use of JVM compatible compilers and interpreters [49].When

GAE gets a web request that corresponds to the URL mentioned in the applications deployment descriptor (i.e., web.xml file in WEB-INF directory) it invokes a servlet corresponding to that request and uses Java Servlets API to provide requested data and accepts response data.

Bigtable comparison with the SQL databases

Bigtable promises high scalability, and availability of the applications using the Google's servers on which the applications are hosted. Some of the important design principles of the GAE are its scalability and availability of the applications, the storage (i.e. DS) which is built on top of the Bigtable a distributed storage, and the MVC architecture that it follows for a thin client interaction(discussed in chapter 5). The special features of the GAE are its ability to provide "free" and attractive platforms for the development of the applications and that are easily uploaded into the cloud where the applications run 24x 7's. These special features make the GAE attractive so that any kind of web applications could be easily developed and uploaded.

Further, Sarrel [69] points out that the Relational Database Management System (RDBMS) that was once revolutionary in 1970's by separating the organization of database from its physical storage laid the foundation to databases like Ingres, Sybase, MS SQL Server, IBM DB2, and Oracle. In 1980's the Structured Query Language (SQL) has become the standard for its performance, scalability, caching, and replication. The Internet has achieved a tremendous growth in the past in government, education, military, and in communication media with its transactional capacities supported by the relational databases. The sites are heavily loaded with content used by relational databases back ends. But there is a need to scale up the back ends for concurrent user support. The traditional relational database offers advantages to transactional data but there is a tremendous difficulty storing and retrieving unstructured data. In the 1980's and 90's the maximum number of entries in a table were 100s with two and three way joins, but

now there are thousands of attributes in a table with seven way joins. As a result, searching is more complex with the arguments and relationships involved. The select operation is acting on all the attributes causing delay by fetching fields that are not required as there are links between information stored. Later indexing became popular with the ability to process complex queries parallel but it took place in vertically scaled RDBMS environment which was unacceptable with the requirements placed by the high performance applications like Facebook, Twitter, Yahoo, Google etc.,

The emergence of Web2.0, social networking and user contributed content has moved RDBMS aside due to the need for scalable databases. Dogan says [71] that a RDBMS has limits on its performance with its inability to scale to millions of concurrent reads or writes. It is seen that companies like Yahoo, Google, Amazon, and LinkedIn have observed the problems and started using NoSQL databases. Unlike traditional databases NoSQL databases are built to quickly scale horizontally with the support of map reduce algorithms for parallel computations on multiple server clusters [70]. The table 4-3 lists out the features of NoSQL databases and relational databases.

But there are some drawbacks about GAE because of its limited support to the data base and the programming languages available. The DS provides the flexibility of storing user information in the cloud. But where does the DS information get stored, the GAE adheres to the US Safe Harbor privacy principles [48].But the information is not available anyone except Google but it is encrypted. The GAE is still is in early stages of development where it provides limited support for application development. But the Bigtable DS that Google is using for most of its applications does not have features that traditional data bases have.

Table 4-3: Table showing the features of NoSQL and traditional databases [69]

Data Store	Use Cases	Advantages	Disadvantages	Key Products		
Key- Value	In-memory cache, web Site analytics, Log file analysis	Simple, Small set of data types, limited transaction support	Simple, small set of data types, limited transaction support	Redis	Scalaris	Tokyo Cabinet
Tabular or Columnar	Data mining analytics	Rapid data aggregation scalable, versioning, locking, web accessible, schema-less, distributed	Limited transaction support	Google Bigtable	Hbase or HyperTable	Cassandra
Document Store	Document management CRM, Business continuity	Stores and retrieves unstructured documents, Map/reduce, web-accessible, schema-less, distributed	Limited transaction support	CouchDB	MangoDB	Riak
Traditional	Transaction processing, typical corporate workloads	Well documented and supported, mature code, widely implemented in production	Cost, vertical scaling, increased complexity	Oracle	Microsoft SQL Server	MySQL Cluster

Bigtable's DS stores data in columns so that it can rapidly fetch the information without the need for multiple tables with less input and output [69].

However there are a few limitations of Bigtable

Limitations of the Bigtable

- A query does not return more than 1000 rows.

- The inequality filters (<,>, <=,>=,!=) operators cannot be applied on more than one property in a query []

```
Example: this query is allowed
SELECT * FROM Person WHERE birth_year >= :min
AND birth_year <= :max

But not this
SELECT * FROM Person WHERE
birth_year >= :min_yearAND height >= :min_height     (ERROR)
```

- If the query has inequality operators and sort order comparison, the query must include a sort order for the property used in inequality and the sort order must appear before the sort orders on other properties.

```
Example: this is not a valid query
SELECT * FROM Person WHERE birth_year >= :min_year
ORDER BY last_name            (ERROR)
```

Similarly, this query is not valid because it does not order by the filtered property before ordering by other properties

```
SELECT * FROM Person WHERE birth_year >= :min_year
ORDER BY last_name, birth_year  (ERROR)

This query is valid
SELECT * FROM Person WHERE birth_year >= :min_year
ORDER BY birth_year, last_name
```

Summary

This chapter showed that GAE allows building and designing scalable applications without any initial costs using Python and Java. But currently GAE does not provide any design principles and guidelines that suggest a design for its applications. I therefore investigate possible architectures that fit GAE and understand their performances in chapter 5 and 6.

CHAPTER 5
ARCHITECTURE AND IMPLEMENTATION

The current generation of Web2.0 applications focuses on rich interfaces, interactive user support, and user collaboration. These features are seen in Google docs, Del.icio.us, Wikipedia, Flickr, and MySpace. The common features the above mentioned applications are sharing stories, searching, tagging which help the users to find efficiently the information. Tagging also helps in structuring the information in a customized way. In some applications providing user interaction and collaboration is the final goal of many tools. The web application development according to the conventions of Web2.0 is quite challenging. A number of languages provide support for developing the web applications but it is difficult to host and maintain the application on a web server and ensure key aspects like security and scalability.

The social networking website, “Our Wise Tales” http://www.ourwisetales.com was developed using the Content Management System (CMS), Drupal [47]. The website Our Wise Tales main functionality is to develop a community for women in science and engineering that allows users to share stories related to their experiences, frustrations, and inspirations. The community aims to bridge across space and generations to build supportive networks. People who visit the community can register, view stories of others, comment on their stories, tag stories. The website visualizes interactions between the users who posted stories and comments made on the stories by other people in the community. The problems with Drupal are that it cannot strictly maintain the differences between the client, server, and the data structures. It often leads to tightly coupled interactions with the database and its interface components. As the website expands due to its popularity, new features will be added to the site to attract the people. At this point, the complexity increases due to the interactions between the increase in size of the community, interactions between the interfaces and database increase. It is difficult to track the

users visit to webpages, and interactions of users with the website. It is difficult to maintain and update the system. To increase the scalable of the websites so that they could withstand the growing demand be accessed from different geographical locations, we need to purchase more storage space and backup the server at regular intervals. This process often complicates the process and incurs more expenditure. The application itself cannot scale above a certain number of requests. In spite of the increase in computing space sometimes the application may not be reliable. To maintain the application it often requires human resources to check with the available storage resources, upgrading the software, installing the updates on the system, buying more storage space if the application has to withstand the growing demand, install software on those machines.

The table shown in 5-1 summarizes the problems with Our Wise Tales application developed with Drupal CMS.

Table 5-1. Problems with Our Wise Tales application [47]

Problems with OUR WISE TALES website
Developing applications using Drupal CMS is complex to learn and implement
Complexity with code as it does not differentiate between the interface, business logic, and data base interactions
Need to update the software on the computers
With the popularity of the website the scalability has to be increased by buying server space
Complexity of inserting new code in to the application when there is a need for additional features

All the above mentioned key challenges had to be faced in addition to the development costs. My research proposes a flexible architecture where scalability, reliability and maintenance issues will be resolved. The architecture uses the GAE frame work and the database as the DS built on top of the Google's database the Bigtable. Using this architecture there is a clear separation of concerns between the clients interface, services and the database functionalities. The Figure 5-1 shows clear separation of concerns between the client and server of the GAE.

The Model View Controller Architecture

The Model View Controller (MVC) architecture is a pattern used in software engineering to separate the domain logic (referred to the application logic of the user) from the input and presentation permitting independent development, testing, and maintenance [67].

In most of the applications the presentation layer is very rarely designed. It is usually coded with the business logic of the applications and works for small and medium size web applications but performs chaoticlly for larger applications. Nowadays, there is an increasing demand for sophisticated web applications with a need for clients to carry out transactions. This requires the server to have an idea of client's state and boundaries which is not possible with the normal client server architecture where the client state is changed with a couple of forward and backward moves on the browser.

Anderson [68] describes in his paper that the presentation layer is the server side code of the user interface. The term presentation layer is used to distinguish between the client's interfaces often called the user interface. For most of the web applications the code is generally written without much thought on design and server side code is written to process HTML pages with. This prototype is allowed to communicate with the back end with little or no additional

design. This works for small and medium scale websites but becomes problematic for large scale applications where the design documentation has to be maintained. Knowing the state of a user helps during the transaction processing of an application for logical transactions.

When a client sends a request through its browser to the server, the request may be either a HTTP GET or POST. The request is usually sent to the server's presentation layer. At this point the server has to decide how it has to respond to the client. This process involves interaction with the business logic of the application. At this stage the business analysis is carried out at the Model layer where persistence is achieved. In this level the presentation layer interacts with the problem domain code and persistence code in order to evaluate the HTTP Request. This interaction is usually carried out in a 3- tiered model which separates it from the traditional client/server model which is 2-tiered. The server presentation layer is also responsible to send output messages to the client. Each time it sends a response to the client it has to interact with the business information layer which in turn talks with persistence storage where a list of outputs are stored and sent according to the requested information. The Figure 5-1 shows the MVC architecture diagram with the different layers needed for the communication between the client and server.

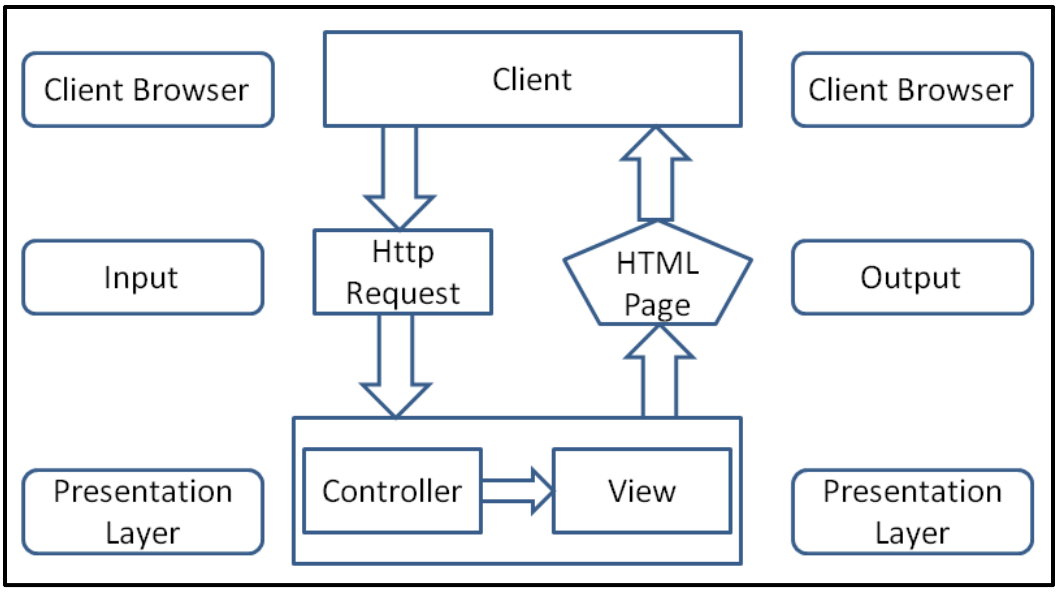


Figure 5-1. Model View Controller Architecture

Architecture of the GAE applications

The Figure 5-2 shows the architecture of different clients connecting to the GAE using the MVC architecture using RESTful WS over the Internet. There is a separation of concerns between the three interacting parties, the client, the server and the DS as shown in the figure. Separation of the different components makes the code transparent and can interact with multiple services.

In the architecture the central part is the server which controls the interactions between the client and the DS. It is also known as the “Controller”. In Figure 5-2 the controller is located between the client and the DS and is interacting with the client view and the DS using the HTTP protocol and the JDOQL/GQL query language. On the client side, different types of clients are accessing the services present on the server using the HTTP protocol. Here the client acts as a “View” which presents the information to the users on the screen. The DS shown in the Figure 5-2 acts as a “Model”. The model fetches information from the DS based on the requested parameters. The GAE allows development of web applications in the pattern shown in Figure 5-2

by exposing its services and also clearly maintains strict separation of concerns using the MVC architecture.

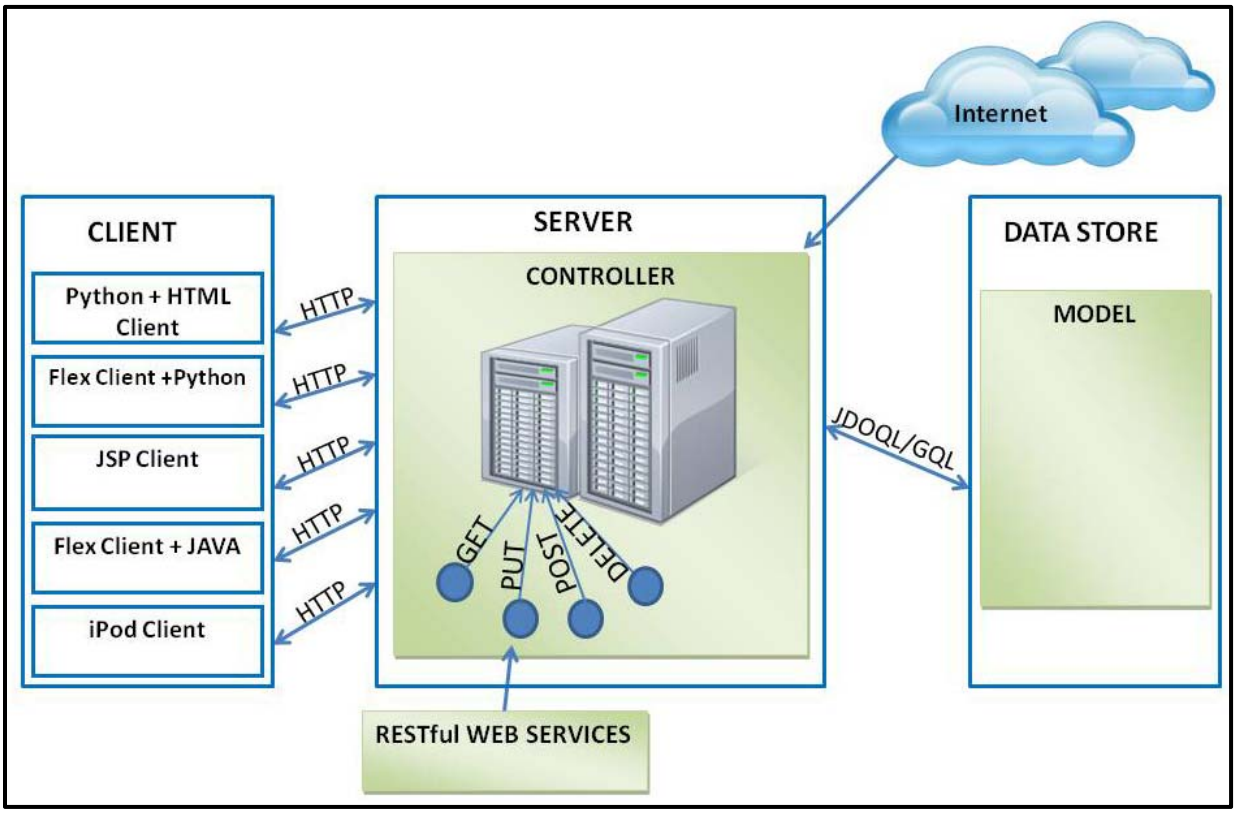


Figure 5-2. Architecture of the application using the Google App Engine

Experiment based on the GAE Architecture

This section describes the experiments designed for the GAE framework with Python and Java languages. The primary goals of the experiments are to develop the services using the "free" GAE framework and MVC architecture. Some of the goals that are to be accomplished based on the architecture diagram in Figure 5-2 are listed in the section below.

Goals of the Experiments

The goals of the experiments are to investigate the GAE cloud architecture and its design principles.

- Develop the application with the basic services based on the features of Our Wise Tales website.
- Investigate accessing the applications using different clients.
- To use the MVC architecture guidelines in experiments.
- To evaluate the performance of the applications in Python and Java languages.

The HTML Client with Python services

A Python HTML application is the first among the different clients connecting to the GAE. To overcome problems related to scalability and for ease of maintenance using traditional programming languages, we moved to a new approach using the GAE. In this application, a prototype of the web site is built using the GAE Python2.5.2 framework with limited features using RESTful WS. The services are exposed as resources and can be accessed as URLs. Figure 5-3 shows the block diagram of the design and the services used in the experiment. The users can login to the website based on URLs and have the view provided in the form of Hypertext Markup Language (HTML). The users register with the application using the register service and they can login. Once logged in they are redirected to the main page which stories the list of stories already posted by other users and an option to post a story as shown in Figure 5-3.

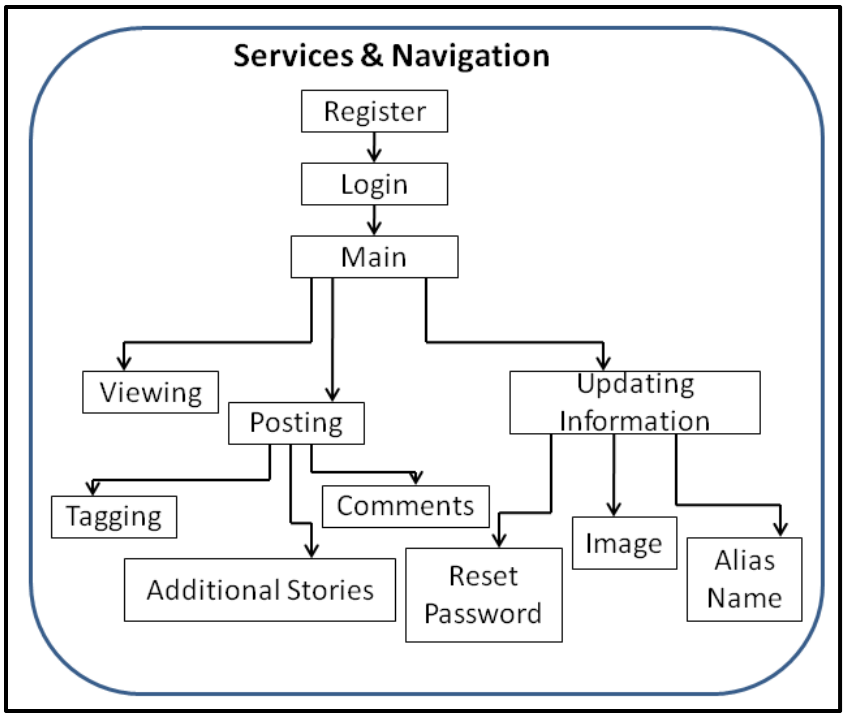


Figure 5-3. Workflow of the HTML client view using Python services

From this screen the users can be redirected to different services. The users can read a story by clicking on the hyperlink under the story name where they can see the date when it is posted, author's name, tags associated with the story, and provide comments to the story using a service called viewing. The users can post their views about a story by using the comments service. If the user is not the author of that story he/she will not be able to append any content to it. If the user is the author they can append the content to the story by using the fields under it. The service that checks whether a user is an author and enables appending more stories is "addmorestories". The users can update stories in many chunks and add tags to them. In this application apart from posting and viewing stories users can personalize their information. The users can change their passwords using the "resetpassword" service. Other personal information of the users like changing the address, phone number, nickname, and images can be done using information service. The table 5-1 shows the DS models used by the services in Python.

Table 5-2. The Data Store models with HTML client and Python services

Data Models	Purpose
IDs	To assign unique ID to users
Stories	List of Stories
Tags	List of Tags associated with stories
Users	User Information
Comments	List of comments associated with stories
Stories Database	List of stories updated in chunks

In the Python application the services provided are for user registration, login, postingstories, commenting, viewing available stories, posting stories in parts, resetting passwords, loading images and updating account information. This application was built using Python version 2.5.2. The application can be accessed from any geographical location using HTTP or HTTPS requests at, https://poststories.appspot.com.The interface is designed using HTML templates, and the data structures are stored in the GAE DS. The data structures used are for maintaining user information, creating stories, commenting stories, and tags. Figure 5-3 shows communication between the HTML client and the navigation of the application using services and the communication between the DS models. This experiment provides a HTML view for the services with clean and compact code using GAE framework that are easy to develop using POST, GET verbs which is based on the RESTful design approach.

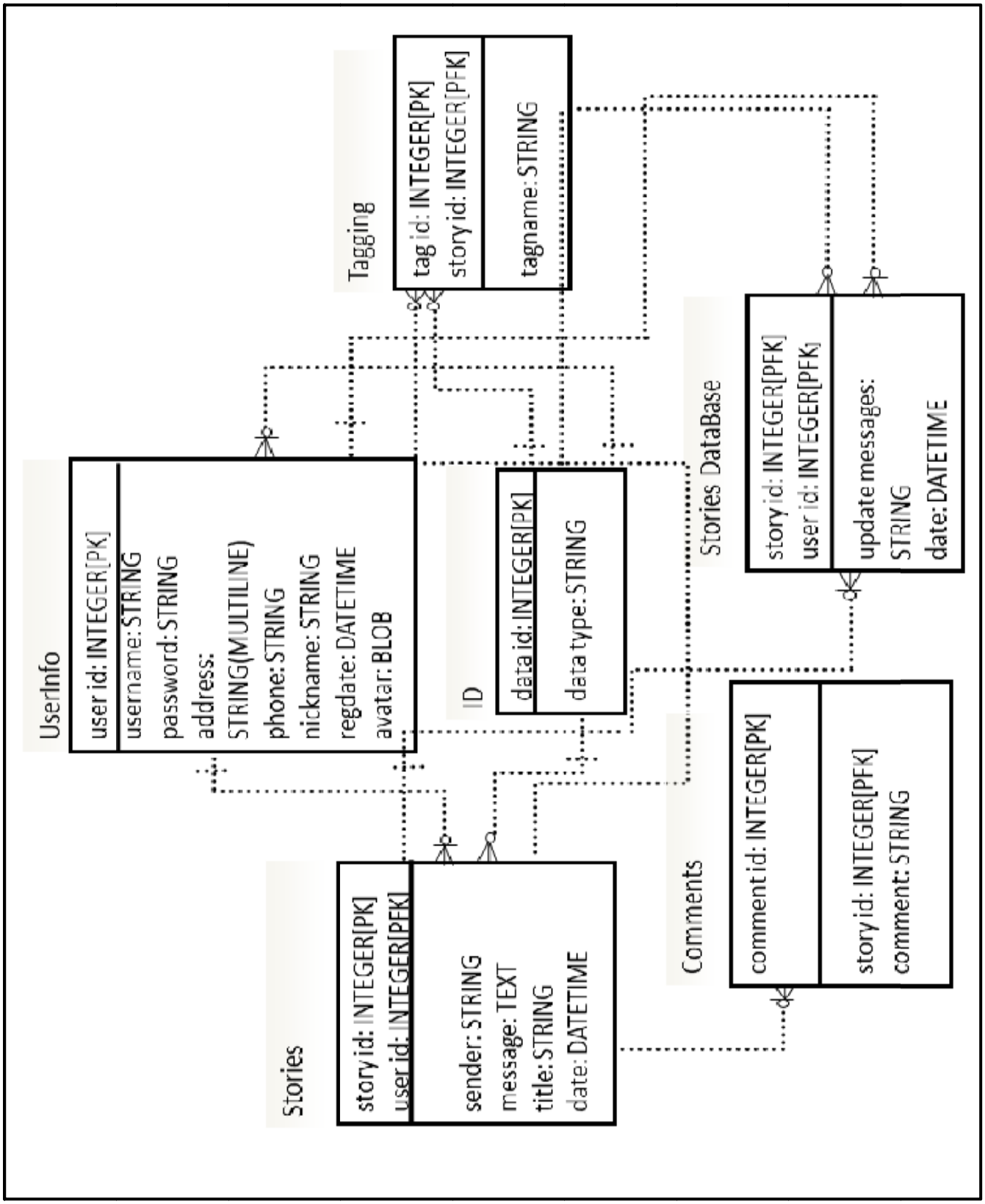


Figure 5-4. E-R Diagram of the data structures in Python

```
class UserInfo(db.Model):
      userid = db.IntegerProperty()
      username = db.StringProperty()
      password = db.StringProperty()
      address = db.StringProperty(multiline=True)
      phone = db.StringProperty()
      nickname = db.StringProperty()
      regdate=db.DateTimeProperty(auto_now_add=True)
      avatar = db.BlobProperty()
```

Figure 5-5. Data Structure for UserInfo in Python

```
users = db.GqlQuery("SELECT * FROM UserInfo WHERE
                                    username='"+self.request.get('user')+"'")
  if users.count() == 1:
    self.response.out.write("""  <h3>User exists already.
                                Please change to a new  username.</h3>""")
    self.response.out.write("""<a  href="/new"> Register Again</a>""")
        else:
            userinfo = UserInfo()
            userinfo.userid = getid('userid')
            userinfo.username = self.request.get('user')
            userinfo.password = self.request.get('pwd')
            userinfo.address = self.request.get('addr')
            userinfo.phone = self.request.get('phone')
            userinfo.account = self.request.get('account')
            avt =  images.resize(self.request.get("img"), 50, 50)
            userinfo.avatar = db.Blob(avt)
            userinfo.put()
            self.response.out.write("<h3>You are Registered</h3>")
            self.response.out.write("""<html><body><br><br>
                <a href="/"> Back to Home </a><br><br> </body></html  """)
```

Figure 5-6. Sample code for Useraccount service in Python

Unlike Drupal, the GAE uses simple functions to define its services and DS models. Figures 5-5 and 5-6 provide the sample code for creating a data structure and user account using Python based services. This application proves that using the reliable GAE applications can be developed using the MVC architecture for a thin client interaction. Thus this framework helps in developing reliable and scalable Web2.0 applications within less time. The code clearly shows separation of concerns where Figure 5-5 is used to store the information to the DS while Figure 5-6 is to fetch the information from client view to controller layer. All the interactions in Python language are based on MVC architecture with separation of concerns. The maximum length of each resource in the application is 60 lines. There are approximately 11 files including the app.yaml and database interactions. The total length of the entire application is collectively approximately 660 lines.

The Flex Client with Python services

The services developed in Python are based on REST and can be accessed with any client application platform. For this application I have chosen Flex3.0 due to its rich internet applications (RIAs), interface layout, layout, and interactive debugging. Adobe labs describe Flex [55] as a powerful Eclipse based IDE that includes editors for Action Script, MXML, and Cascading Style Sheets (CSS). It allows previewing user interface layout, appearance, and behavior using a rich library of built-in components. It allows exchange of data using WS using HTTP protocol, request XML and responses.

The Flex client application was developed using Flex3.0 to provide client interface, the GAE with Python services, and the DS of the GAE. The flex client provides rich interface and

accesses services using URLs associated with the GAE. The Figure 5-7 describes the navigation of the application and its available services using a workflow.

When a user registers using the Flex client interface, the request is sent with the parameters of the users and the GAE service verifies with the available information in the DS using the login http request service, and hence returns users success or failure back to the client. If the client request succeeds it proceeds to the next step where client can view the available stories and can further post a story. The services available with the Flex client are user login, user registration, posting stories and viewing stories.

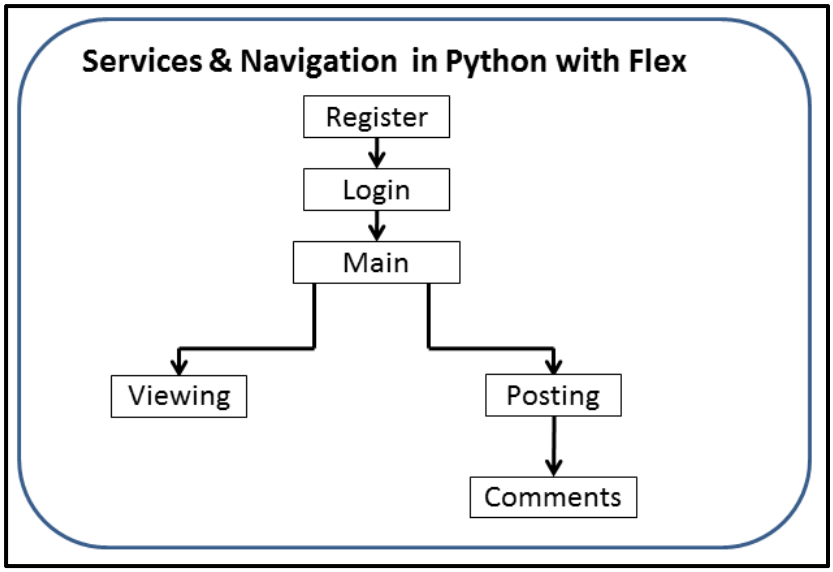


Figure 5-7. Workflow of the Flex client view using Python services

The main goal of this application is to connect the Flex interface with Python services and to provide a rich interface using the Flex development environment. The appendix shows the client interface screens using the Flex3.0. The raw services and application developed quickly by the GAE can be connected with the rich Internet applications development environment like Flex to develop the web applications with clear separation of code between the client applications,

and the business logic. The problems described by Anderson [68] about the integration of client and server code are not encountered using the separation of code features with the GAE environment. Thus the applications can be developed quickly and easily without wasting the developer's time. The Python client with HTML describes the application developed in Python and has a basic client interface developed using HTML. From this experiment it is evident that using the GAE users can develop the applications that are user friendly, easy to develop and provide rich and interactive client interfaces.

Table 5-3. The Data Store models with Flex Client and Python services

Data Models	Purpose
UserInfo	User Information
Stories	List of Stories
Tags	List of Tags associated with stories
Comments	List of comments associated with stories

The JSP Client with Java services

The GAE launched Java as the second language next to Python on its framework. Using Java the applications can be developed using the service oriented approach where URL is used to navigate through the services. In this thesis, I used Java as a language to create the services and JSP as a client to provide the view for the services. The servlets are used to create the services using Java6.0 and access them with JSP client.

The application has been created with Eclipse builder using Google Plug-In for Eclipse. The access to the DS is provided with the Java Database Objects (JDO) using a query language

JDOGQL. The GAE Java applications use Java servlets standard to interact with the web server. The application files, compiled classes, JAR and static files are arranged in a directory structure using the WAR layout for Java web applications. The application has HTTP servlet classes that can process and respond the web requests.

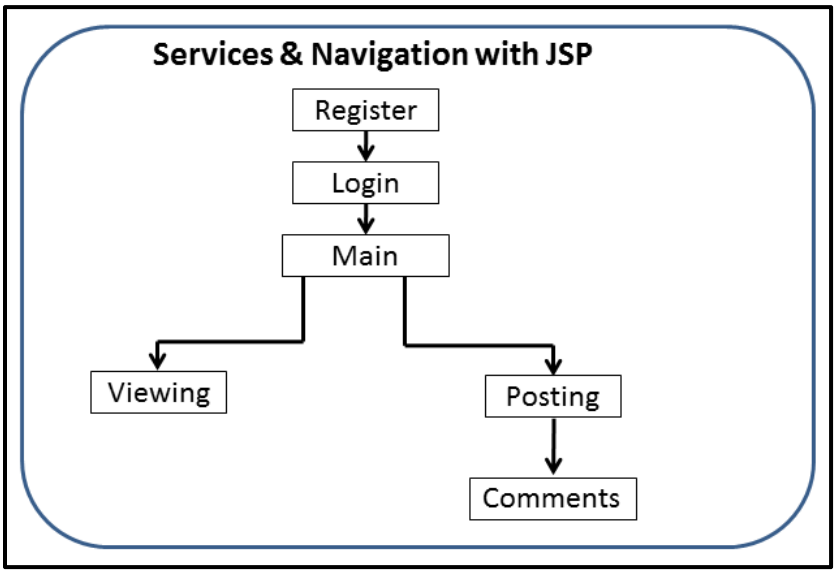


Figure 5-8. Workflow of JSP client view with Java Services

Servlets can also give output in the form of HTML but it is complicated to maintain them. It is better to use a template system that provides the functionality separately in files with place holders to insert data provided by the application. There are many template systems supported by Java, but I used JSP's as they are part of the servlets and the GAE compiles JSP files in the applications WAR automatically and maps them to the URLs. The Figure 5-8 shows the workflow of Java services using JSP client.

I developed the services using Java servlets API and data is stored using JDO. Using JDO, instances of the classes are stored in GAE DS and retrieved as objects. Also each request that uses DS creates a new instance of the Persistence Manager Factory (PMF) Class. As the

instance needs time to be created, it can be stored in a static variable to be used in multiple classes and files. The table 5-3 shows a list of DS objects used with JSP client and Java services.

Table 5-4. The Data Store Objects with JSP client and Java services

Data Objects	Purpose
Users	User Information
Stories	List of Stories
Tags	List of Tags associated with stories
Comments	List of comments associated with stories

Figure 5-9 shows the code for Java based UserServlet developed using JSPs. The DS objects used in this experiment are for storing users, tags, stories. Some of the services developed for the JSP client are user registration, login, viewing stories, posting stories. This code shows services are re-implemented in Java, with the services rendered in the form of JSP pages. These services are developed using the RESTful mechanism and users can navigate between the pages using URL.

This application was developed to evaluate the two development languages that Google is providing for its GAE framework. Java is a widely used language throughout the world for application development. The GAE Java provides a JSP servlets and CSS to develop the rich user web applications. Using the Java environment with the GAE, the applications can be developed and uploaded easily using the Eclipse builder tool. But the code for the applications in Java is longer than the code for the applications in Python. The Java servlets are also designed with GET and POST requests with separation of code for DS interactions. Java application takes 1500 lines of code. The length of the code is due to lengthy DS queries defined by JDOQL. In Python each service is developed using the GET and the POST methods. The interface can be

developed using Django or HTML templates that provide user interfaces. The services should be registered with the app.yaml file in order to on the GAE local host. The application can be uploaded to the cloud with 2 or 3 commands. The Java GAE provides JSPs for the client view and the business logic maintained on the servlets that interacts with the database using the JDOQL query language.

```
public void doGet(HttpServletRequest req, HttpServletResponse resp)
throws IOException {
PersistenceManager pm =PMF.get().getPersistenceManager();
      String username = req.getParameter("username");
      String password = req.getParameter("password");
      Userinfo user=getUserByUserName(username,pm);
      try{
      if(user!=null)
      {     if(user.validatePassword(password))
         { HttpSession session = req.getSession(true);
            session.setAttribute("user", username);
            resp.sendRedirect("/posting.jsp");
            PrintWriter out =resp.getWriter();
           out.println("Successfully LoggedIn");
           out.println("<user>");
           out.println(user.getUserName());
          out.println("</user>");
         }
         else errorMessage(WRONG_PASSWORD,resp);
      } else {
       errorMessage(NO_SUCH_USER,resp);
      } } }
```

Figure 5-9. Sample code for UserServlet in Java

The Flex Client with Java services

To develop a Flex client with the GAE Java, the Flex builder plug-in and the Google plug- in with Eclipse has to be installed. The services are developed using HTTP Servlet but the client here is a Flex3.0. It has the entire interface layout and the communication between Flex client and Java servlets using HTTP service. The client and the server respond to each other based on URLs provided in the WEB-INF folder. The Figure 5-10 shows the workflow of a Java services with Flex client.

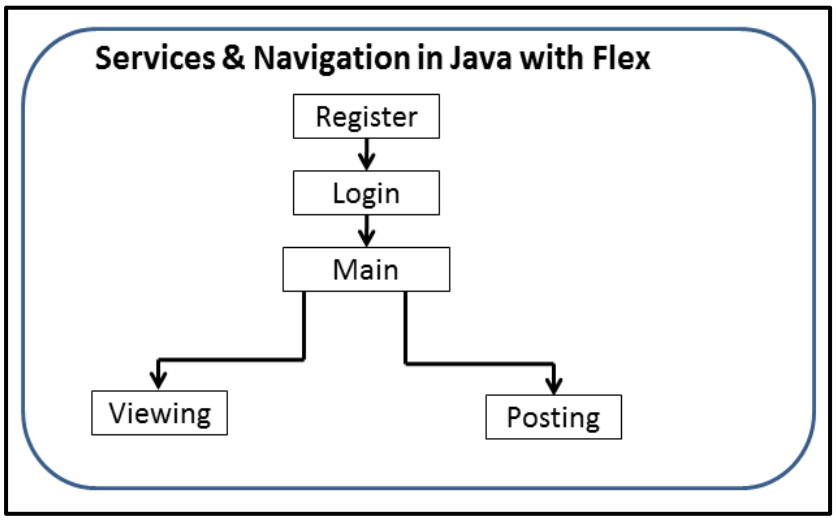


Figure5-10.Workflow of the Flex client using Java services

The table 5-4 shows the DS objects used with Flex client and Java services.

Table 5-5. The Data Store Objects with Flex client view and Java services

Data Objects	Purpose
Users	User Information
Stories	List of Stories
Tags	List of Tags associated with stories

The services designed for the Flex client interaction with Java Servlets are user registration form, login, stories display page and story posting page. The DS tables are user information, stories, and tags that store the information of the users and stories with their tags. The services used in this prototype application are used for login, and posting a story. If a user wants to tag a story they can provide tags separated with commas. The aim of this experiment is to see how the Flex communicates with the raw services developed in the GAE.

The iPod Client with Java services

The Internet is not only accessed on desktops and laptops it is also accessed by mobile phones. Serhani et al. [54] states that, the mobile phone companies increased profits as the proportion of global population using mobile devices has increased in the recent years especially in developing countries. Internet is extensively used on small screen devices like smart phones. In April 2009, the iPhone accounted 43 percent of mobile web usage and 65 percent of HTML usage. It is expected that wireless subscriber rates will reach 2 billion by 2013. Among all smart phones, the leading competitors are Apple, Android, Black Berry Curve, and Palm Pre. Apple currently has sold 4 million devices in the second quarter and expects the numbers to increase up to 5 million units in 3rd quarter, and 7 million units in the 4th quarter.

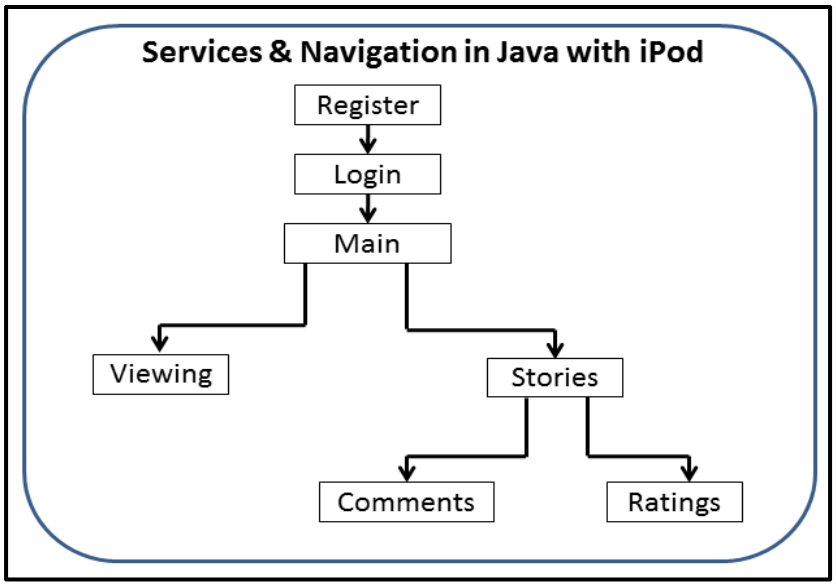


Figure 5-11.Workflow of the IPod client view using Java services

Table 5-6. The Data Store Objects with IPod client view and Java services

Data Objects	Purpose
Users	User Information
Stories	List of Stories
Tags	List of Tags associated with stories
Comments	Stores list of Comments associated with stories
Ratings	List of Ratings provided for the users stories

Originally the application was designed with JSP but I extended the design of the application to be accessed on mobile clients like IPod touch. In terms of the functionality rather than posting stories the application has services that can view stories, rate, and comment them. The Figure 5-11 shows the workflow of the Java services with IPod client view. The table 5-5 shows the DS objects used with Java services for IPod client.

The IPod client application has a rich interface developed using CSS that provides an enhanced view of the application for mobile or smart phone users. In this application there are REST based services where the clients register and login and they can view the existing stories and post comments or ratings instead of posting the story itself. Due to the IPod's limited screen size it is difficult for a user to post a story using smart phones instead they can read stories and post comments to it. The IPod client application uses JSP as a client where the CSS are used to edit the view. The JSP pages interact with the DS using the query language GQL.

CHAPTER 6
APPLICATION TESTING AND EVALUATION

This chapter evaluates the performance of the applications with various loads. The evaluation of applications is based on checking the time taken to process certain number of requests. I choose to evaluate our applications by checking the performance of services that are accessed by URLs. The table 6-1 shows the list of languages and their corresponding interfaces.

Table6-1. List of languages and client interfaces

Languages	Interface
Python	HTML
Java	JSP

Evaluation Plan

I used Apache JMeter to run the tests. JMeter is a Java desktop application designed to load test its functional behavior and measure the performance. It is used to test web applications for many server types including web requests (HTTP, HTTPS). The experiments are conducted on a machine with the following configuration 2.66GHz CPU, 3.00GB RAM, 100Mbps network card running over a 400Mbps Ethernet Hub. The operating system is Microsoft Windows XP SP3. The languages used are Python and Java with the GAE engine runtime. This chapter discusses the load tests performed on the application.

The evaluation for testing the Python and Java services is to test the performance of the applications under different workloads. The attributes like verification, validation that define the performance of the application are scalability, reliability, and resource usage which demonstrate

whether a system meets performance criteria. The load tests are modeled to simulate the expected number of users accessing the WS concurrently. The stress testing is done to test the applications performance beyond normal users to determine the stability of the application, and break the application by overwhelming its resources [56].

For evaluating the performance of the applications developed with Python and Java languages different test beds are designed that are discussed in sections 6.2 and section 6.3.Each test plan has to answer the following questions.

- What is the anticipated normal workload?
- What is the anticipated peak number of users?
- What is the good time to load test the application? This may sometimes crash the servers.
- What is testing intended to achieve?
- What is the sequence for the test?
    1. Functional (low –volume of users)?
    2. Benchmark (average number of users)?
    3. Load test (maximum number of users)?
    4. Test destructively (the hard limit)?

The test bed for testing the performance is based on load tests with varying workloads. The tests are performed for low, normal, and high volume of users. The low and normal users are for 10, and 50 users. The tests handle a peak load of 100 users concurrently for high volume of users. The tests were usually performed everyday in the morning for N days (where N=5). The sequence of the tests is used to handle 3 types of user's functional users that handle less number

of requests. The bench-mark for the users is to test for an average of 50 users, and high flow of user requests of about 100.

The reasons to carry out the tests with a limit of 10, 50 and 100 users are based on the limited community of users who concurrently access the application. So the tests are conducted with those limited values to mimic the real time user scenario. The applications are tested during Monday to Friday, morning 9am till 12 noon. The N value is chosen as 5 because the tests are carried during the week days to see the performance of the GAE servers.

The test bed for evaluating Python and Java services is similar. The tests are conducted using Apache JMeter as a preliminary evaluation mechanism even though it is believed that results may not be 100% accurate due to minor Java timing errors. However, to test the reliability of the services and JMeter the tests are repeated for 5 days. The reason for testing on weekdays is to check the performance of the GAE services when there is traffic on the network to simulate the requests on the web. The tests are conducted on a university network (University of Saskatchewan) where it is believed that traffic is shaped. The traffic is shaped due to requests on the university network where there may be people be watching videos, connecting to heavy audio and video files during the weekdays. This argument is true but the other networks which may be used for commercial or networking purposes may also be shaped by the network providers. Thus I conclude that these results are preliminary tests to check the scalability of the applications using Python and Java languages.

The Apache JMeter can be used to test applications using the HTTP or File Transfer Protocol (FTP) that can create the test plan based on the requirements. The JMeter has a web test plan that has two important components the Test Plan and a Work bench. The Test plan is a container to perform tests and the Work bench is a container for any test to be performed or a

portion of the test to be moved in to the test plan. The test plan has many sub components that can be added to it. When the test plan is right clicked a context menu appears to add the items to a test plan.

The test bed for this experiment creates a load on servers and tests the performance of the services accordingly. A JMeter test creates a loop and a thread group. The loop simulates sequential requests to the server with a delay and a thread group is designed to simulate concurrent load. A load test using the JMeter test plan is to execute a sequence of operations.

The important components of a test plan are Thread group, Controllers, Assertions, Listeners, Timers, and Configuration elements. The **"Thread Group"** tells the users number of users to simulate, how often the user requests need to be sent, and how many requests they need to send. There are two types of **"Controllers"** samplers and logical controllers. The samplers tell the JMeter to send a request and wait for the result. There are many samplers like HTTP Request, FTP Request, and JDBC Request etc. Logic Controller enables to customize the logic of that JMeter follows. The **"Assertion"** allows assert whether the results returned from the server are as according the results that we expected. The **"Listeners"** provide the information that JMeter gathers when a test is run. The **"Timers"** are used to pause between each web request that JMeter send to the server. By default the timer is off. The **"Configuration Element"** is used to add or modify the requests and works with the samplers [58].

Goals of Evaluation

- To evaluate the services by varying number of requests (10, 50, and 100).
- To perform repeated requests for N days and calculate the difference between them.

- To calculate the time taken for each request individually for N days.
- To calculate the time taken by the workflows.
- To test the performance of the services and workflows.
- To study the scalability of services developed in Python and Java.

Phase1: Test Bed for the experiments in Java and Python

The test bed for the services in Python and Java is to evaluate the performance of the services over a certain time period. In each of the languages, a set of services with Python and Java experiment are considered for evaluation. Initially each service from both the languages is evaluated for a fixed period of N days where (N=5). The services are evaluated for varying number of client requests with a fixed time difference between each request.

As discussed earlier, the phase 1 experimental test bed services developed both in Python and Java are evaluated with increasing loads for 5 days. Each of the measurements is recorded everyday in the morning 9 am till 12.

Test Bed

The test bed is a simple HTTP web request using Apache JMeter and defined in 4 steps. In step1 a thread group is created. The thread group tells JMeter the number of users, how often the requests have to be sent, and how many requests have to be sent. These properties are explained by the fields number of threads (users), ramp-up period (in seconds), and loop count. The field number of threads tells the JMeter the number of user simulations to be created, the ramp-up period in seconds indicates the time delay between each thread. For example if the number of threads is 6 and ramp-up period is 12 seconds then JMeter would send each request with a delay of 2 seconds. The number of loops indicates number of times the requests have to be sent. The Figure 6.1 shows thread and its fields.

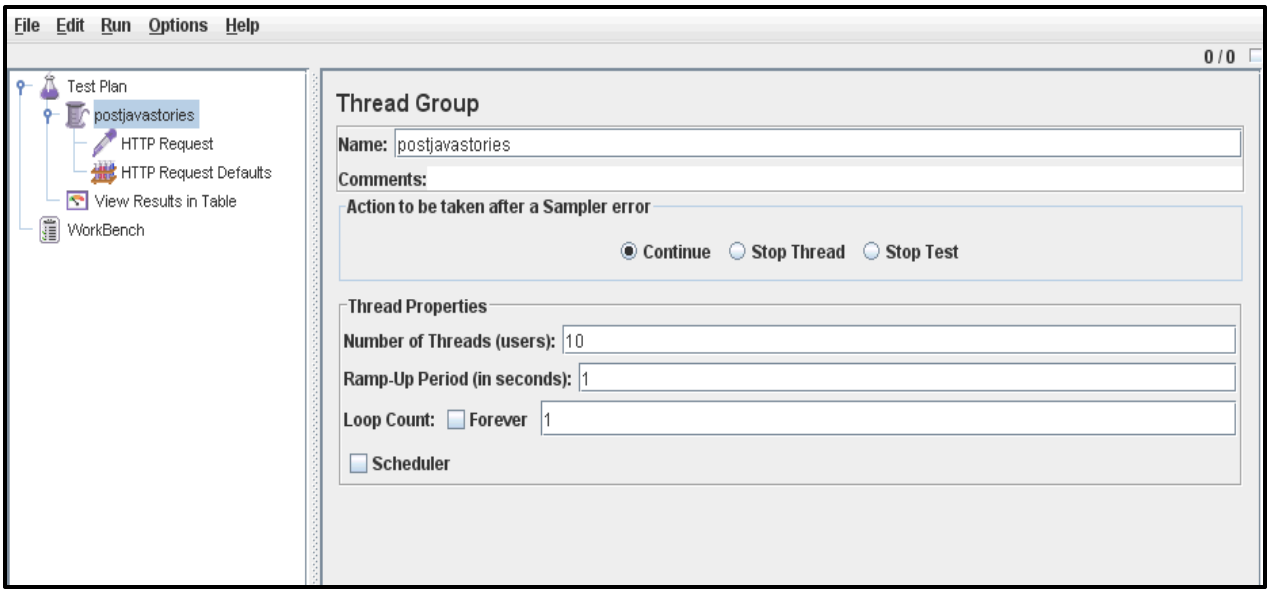


Figure 6-1. Thread group and its properties

In step2, the tasks to be executed by the JMeter are defined. The thread group is selected and mouse right click option is chosen to add a config element the “HTTP Request Defaults”. The Figure 6.2 shows the “HTTP Requests Defaults” page with values.

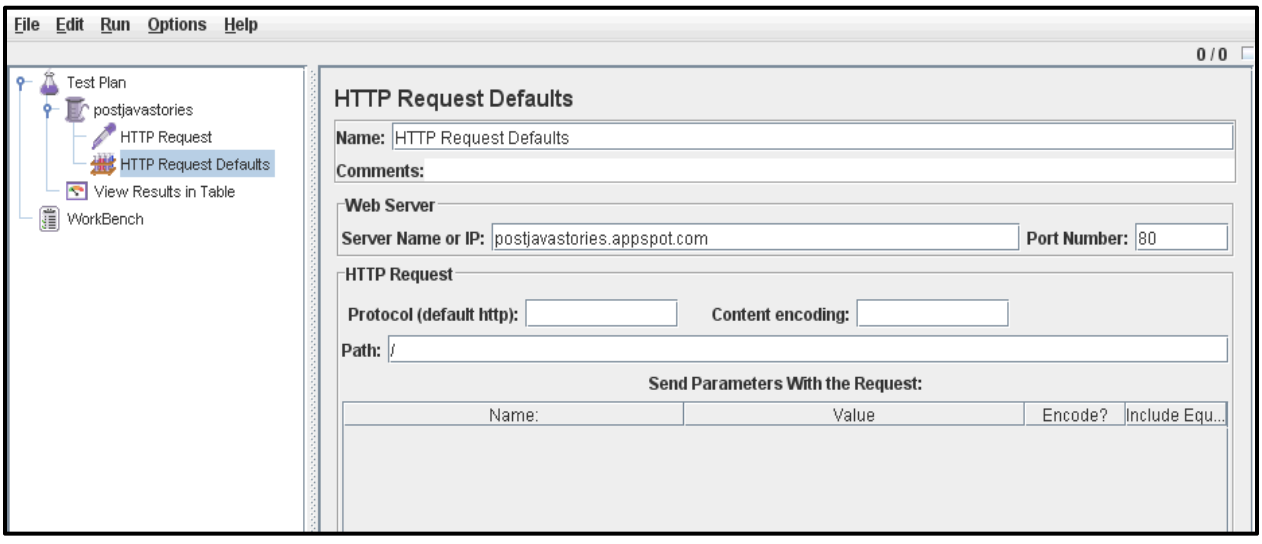


Figure 6-2. The HTTP Request Defaults and its properties

In step3 a HTTP Request is defined by selecting the thread group and choosing a sampler with the name HTTP Request. In this page the name, path, port, and method filled. The name, path and port numbers are the same as default “HTTP Request Defaults” but the method is changed accordingly based on the type of request GET or a POST. The lists of parameters are sent along with the request depending on the type of request. The HTTP Request is shown in Figure 6.3 with its attributes.

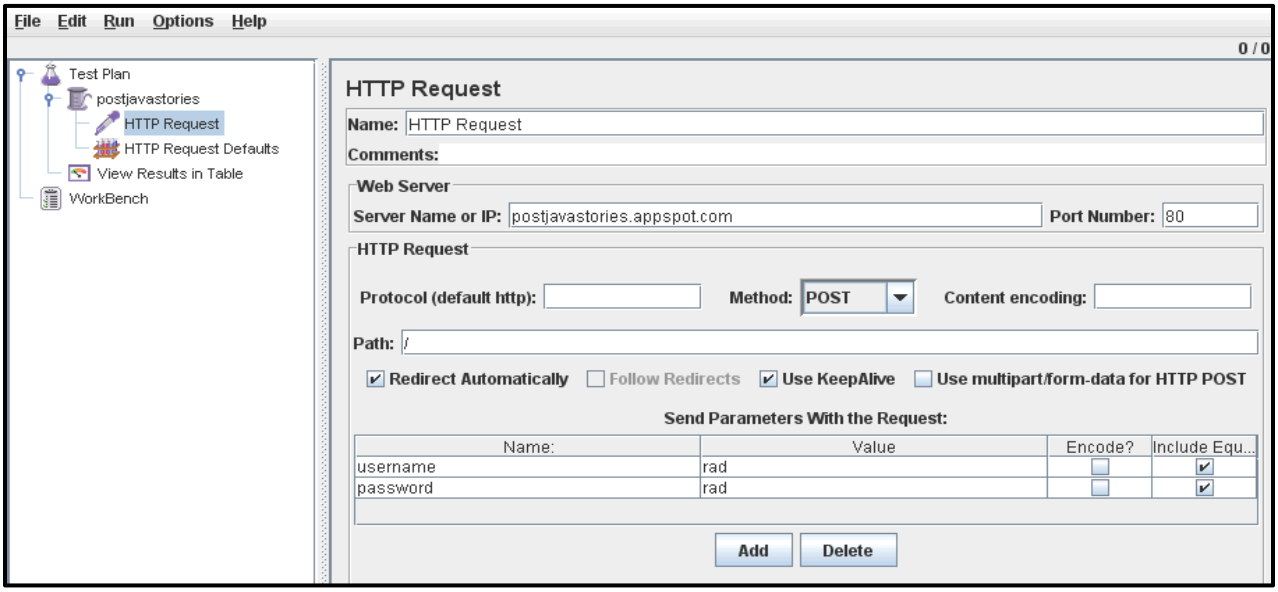


Figure 6-3. The HTTP Request and its attributes

In step4, the results of the test are viewed by adding a listener to the test plan. This element is used to store the results of the HTTP request. The listener is added by selecting the test plan an adding a listener and adding an element to view results in a table. These results show the number of the requests, the thread group they belong to, time taken in milliseconds, the result of the request either success or failure.

The Figure 6.4 shows results in a table for the login service.

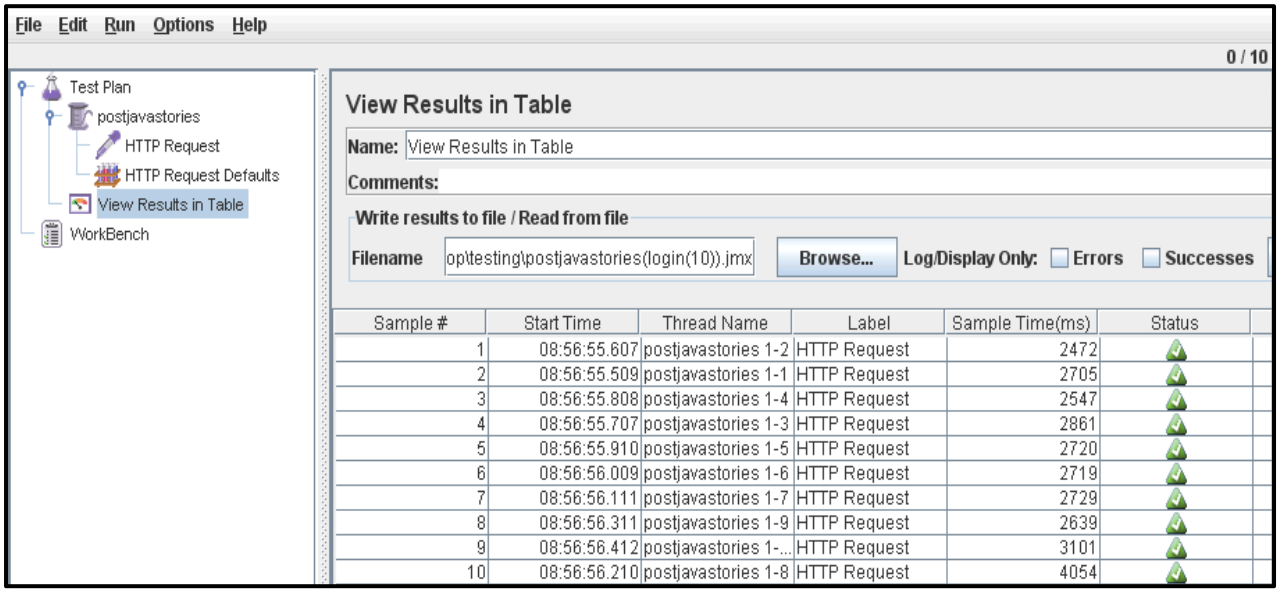


Figure 6-4. The View Results in a table with its fields

Results of the Phase1 Experiments

This section discusses the results for the Python and Java based services that are evaluated using the test bed in section (described for phase1 experiments). The services are evaluated for varying number of client requests 10, 50, 100 for 5 days. The tables in the 6.2 and 6.3 shows the services evaluated with varying client requests for Python and Java languages. The individual services are evaluated for 5 days for the best and worst performances with the workloads. The graphs for each of services are shown as in the sequence of the tabulated service names. Also, for the tests on the services in Python and Java, maximum, minimum, and average values for the time taken in milliseconds are calculated. Based on these values the Delta is calculated.

The Delta is the difference between maximum and minimum values. The graphs display variance values for each service in Python and Java.

Delta (A) =Maximum (A)-Minimum (A)
Where A is the column in the table

Results of the Python services. Based on the test plan all the services in Python are evaluated for different number loads of client requests (10, 50, 100) for N days where (N=5). The services in python are evaluated for checking the best and worst performance among the five days with workloads and also for calculating the overall variance. The table 6-2 shows the services in Python and the graphs for the python services in the same order as tabulated.

The first part of the evaluation shows the performance of the services for the workloads 10, 50, and 100 over a period of 5 days the graphs are recorded to calculate the best, worst, and average time taken. The Figure 6-5, 6-6, and 6-7 show the time taken for login service for 10, 50, and 100 workloads. In each graph the best and worst time taken is observed as the best time defines the least time taken to process the workload. The worst time indicates the maximum time taken to process the workloads.

Table 6-2. Table showing the list of services evaluated in Python

Service name	Request Type
Login	POST
Main	GET
Account	GET
Poststories	POST
Comments	POST

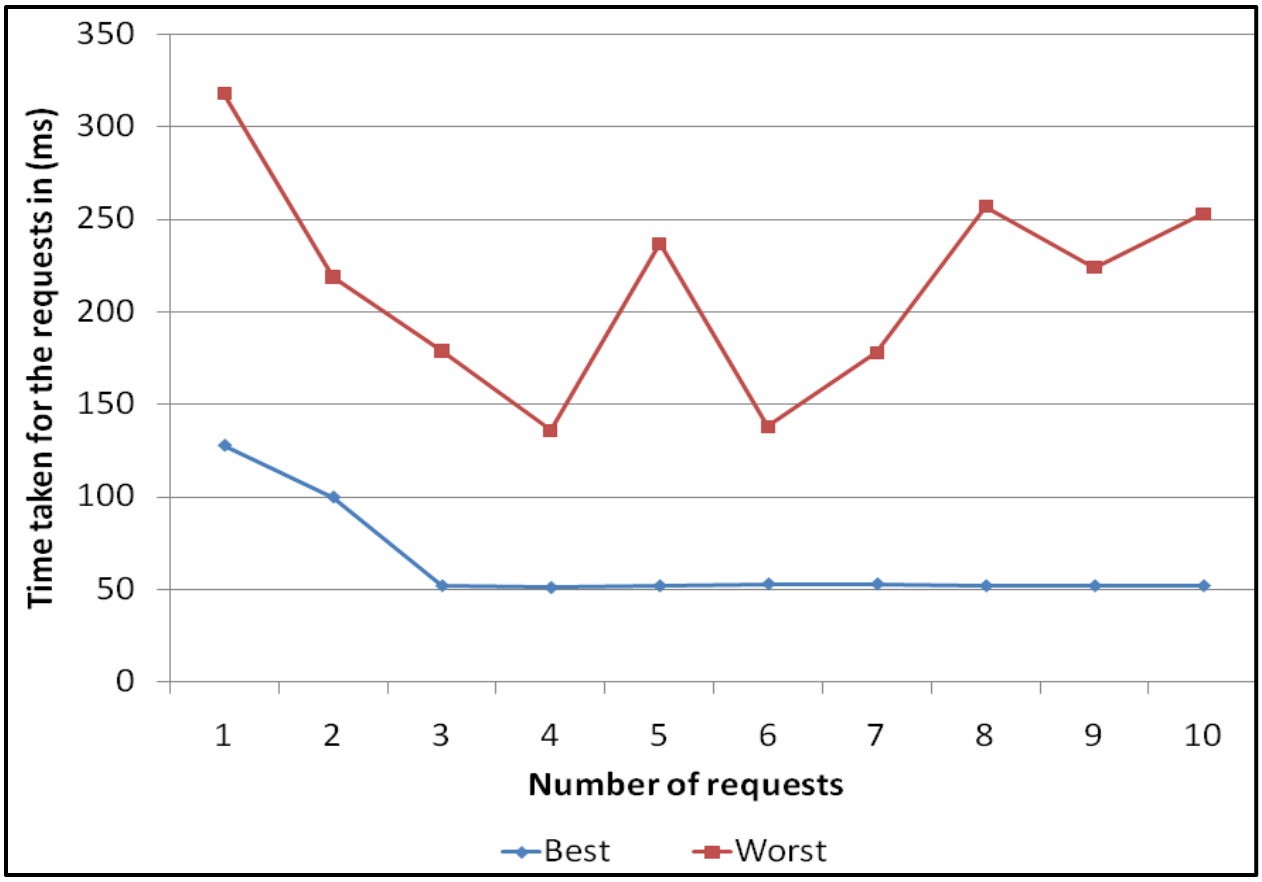


Figure 6-5. The performance of Python login service for workload (10)

The Figure 6-5 shows the best performance as the least time taken to process the workload of 10 requests. Figure 6-6 shows the huge difference between the best performance and the worst performance. It is understood that the average performance for the services should be between the best and the worst performance graphs. The Figure 6-7 shows the performance graph for login service for a workload of 100. In this graph the worst performance is indicated by the highest time taken line. It is indicated as worst performance because of increase in workload, the time taken is also increased after 85 requests.

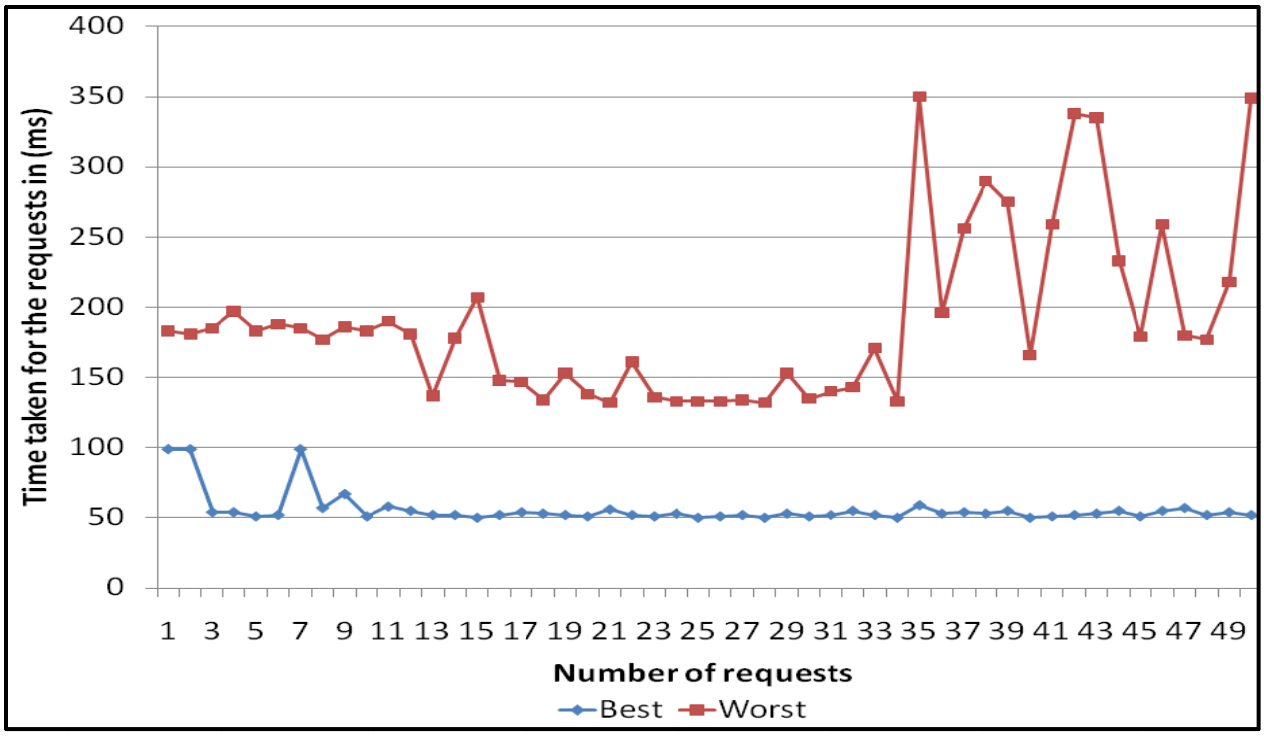


Figure 6-6. The performance of Python login service for workload (50)

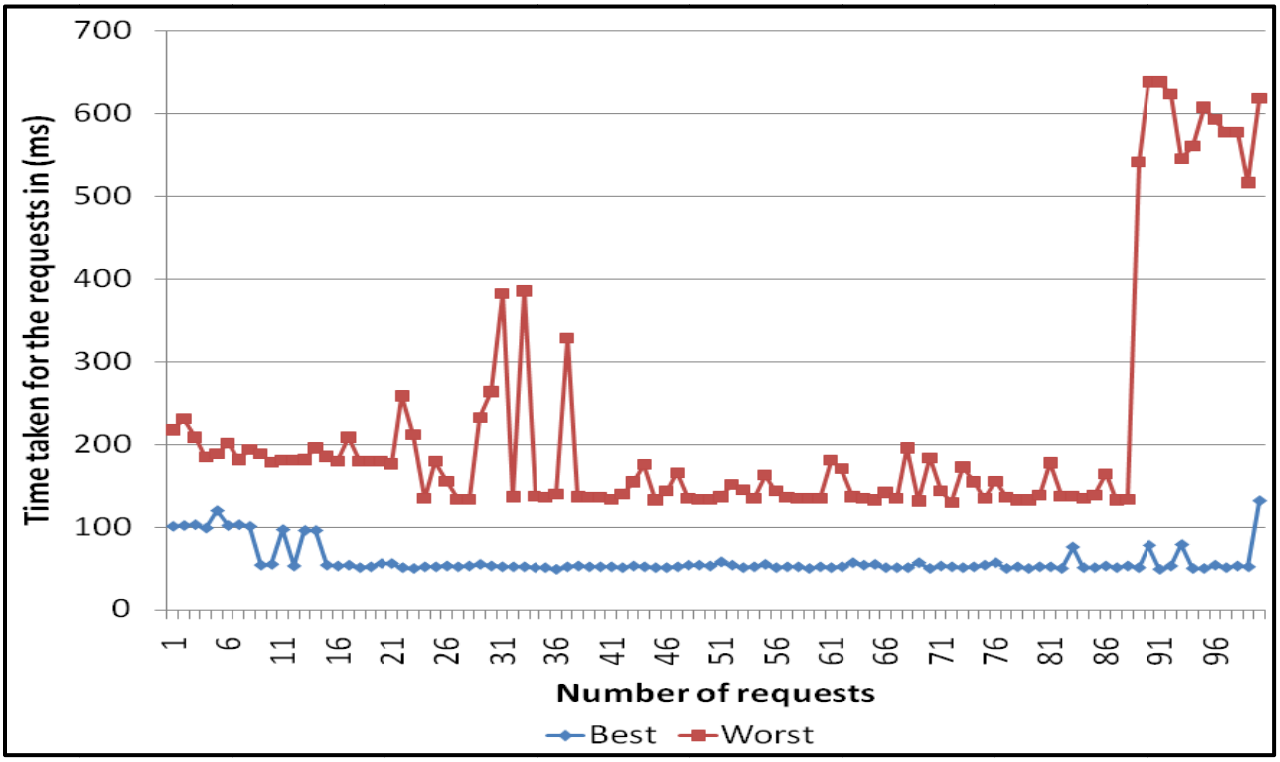


Figure 6-7. The performance of Python login service for workload (100)

The second part of the evaluation shows the variance of the services. For each day the average time taken and its Delta is calculated. As discussed earlier, the delta is calculated as the difference between maximum and minimum time recorded for a day. Similarly the averages and variances for the 5 days are taken and graph is drawn with the variances recorded from day1 till day5 (from Monday till Friday). The Figure 6.8 shows the time taken for login service in Python based on the variances. The 3 different lines indicate the time taken for varying loads with the variances as indicated.

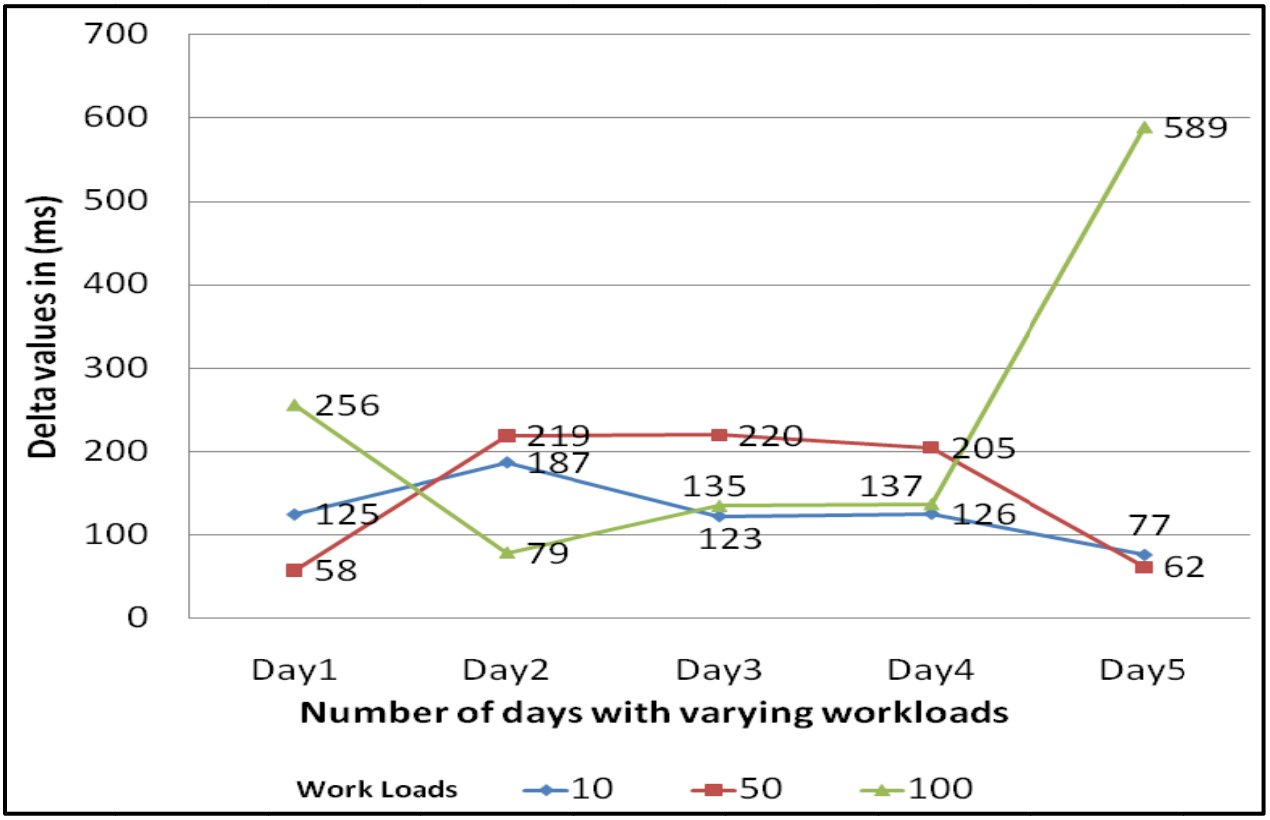


Figure 6-8. Difference of the times taken (ms) using login service in Python for workloads

The Figure 6.8 shows the time taken for login service with different workloads. This service performs a POST operation. With increased workload the delta value also increases. For example for the workload 100 the delta value of time taken each day starting with day1 is high and gradually decreases with the GAE balancing the load by running the application on many

servers. When the load on the servers comes down, time taken for the requests also decreases as a result the graph shows rise and fall at certain locations.

Similarly the performance graphs for the main service are shown in Figure's 6-9 till 6-12. The figure 6-9 shows the performance graph for 10 requests, Figure 6-10 for workload of 50 requests, and Figure 6-11 for workload of 100 requests. It performs a GET operation.

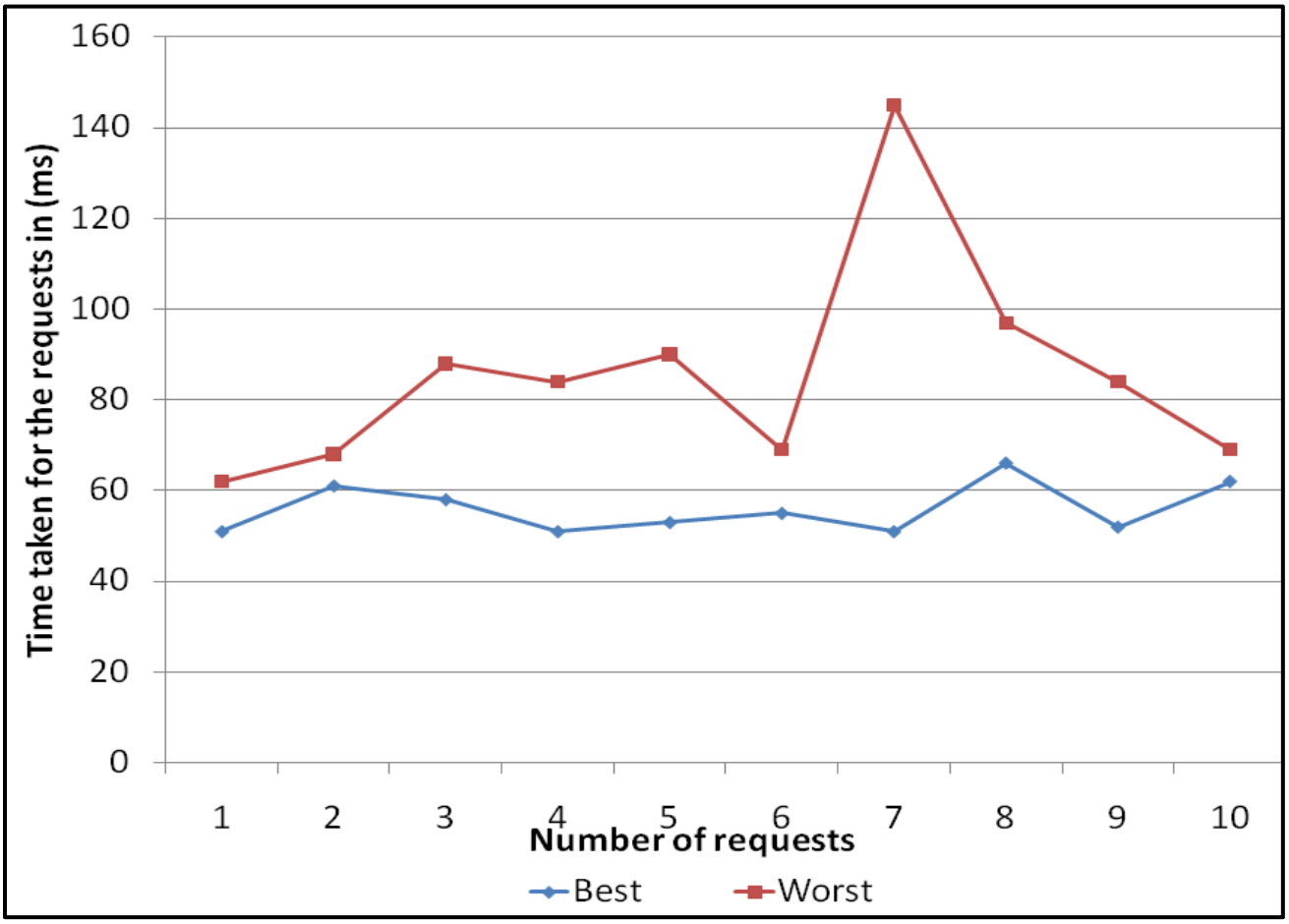


Figure 6-9. The performance of Python main service for workload (10)

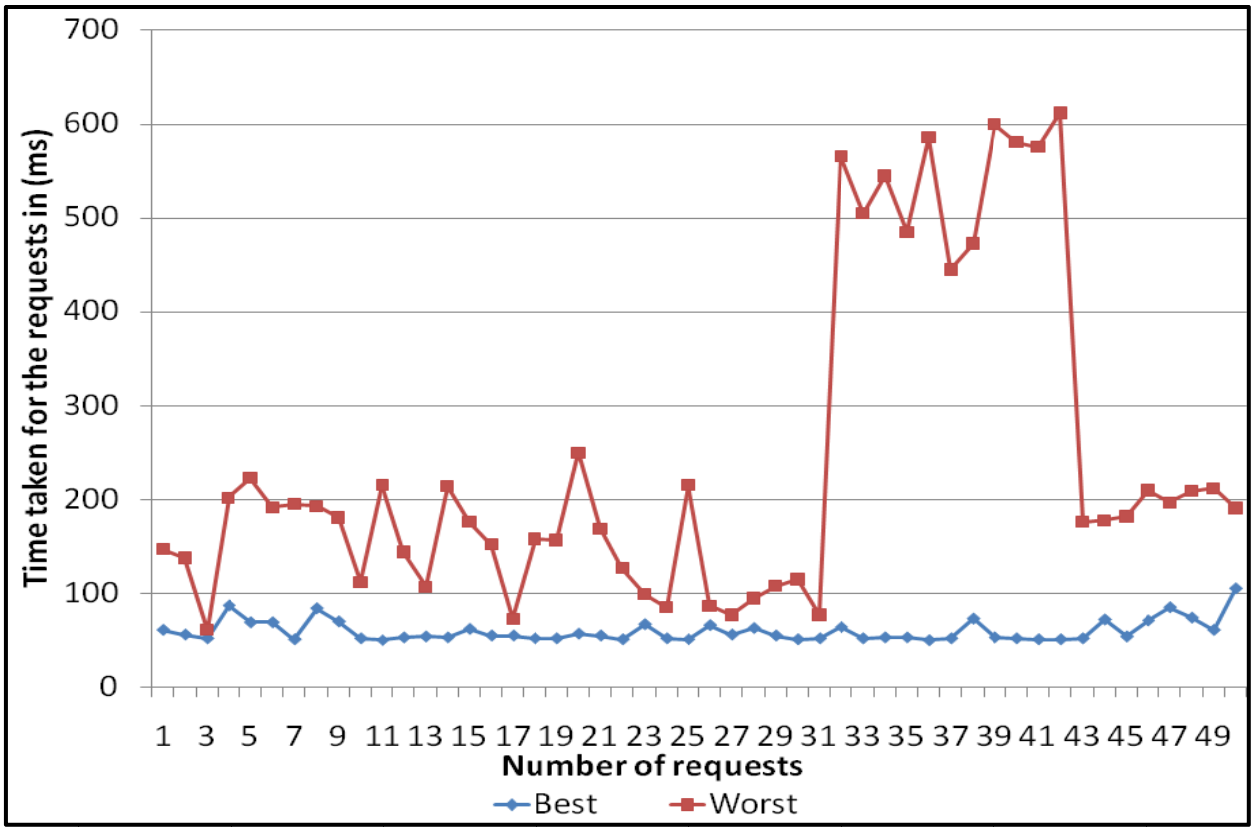


Figure 6-10: The performance of Python main service for workload (50)

In the Figure 6-9, the best and the worst performances does not overlap where as in Figure 6-10 and Figure 6-11 shows the overlap with a difference in time taken in each of the graphs. But for the workload 50, worst and the best graphs show a minimum difference between them. And for the workload 100, the Figure 6-11 shows that the time taken for worst performance line shows many fluctuations in the graph indicated with high volume of requests and maximum time taken. But the increasing loads in both the figures are balanced by the GAE server that runs the applications. The servers running the applications have balanced the load by sharing the load with the servers next to it thus a lowering line indicating the decreasing load is shown.

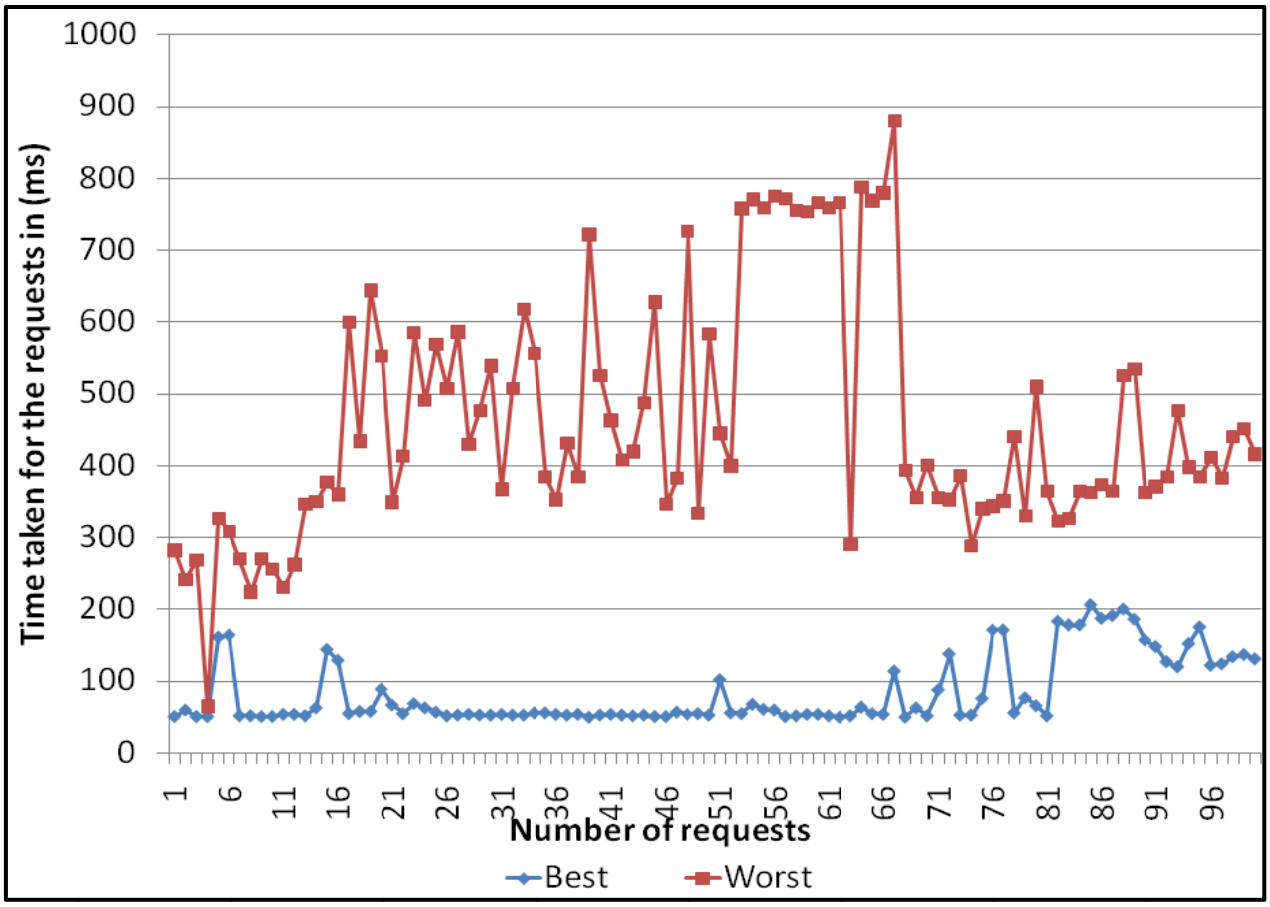


Figure 6-11. The performance of Python main service for workload (100)

The Figure 6-12 shows the delta values for main service in python for the workloads (10, 50, and 100).In this figure it is observed that the delta values for processing 50 requests on day4 is more than the delta values for 100 requests. For 100 requests workload starting day3 the delta values drops each day. This is due to the GAE servers automatically balancing the load after a certain threshold where they distribute the load to the corresponding servers automatically. During this time the overall load on the available servers reduces and the time taken for processing the requests reduces. Using this approach there is a low failure of the requests and fewer burdens on the server and ultimately the time is reduced.

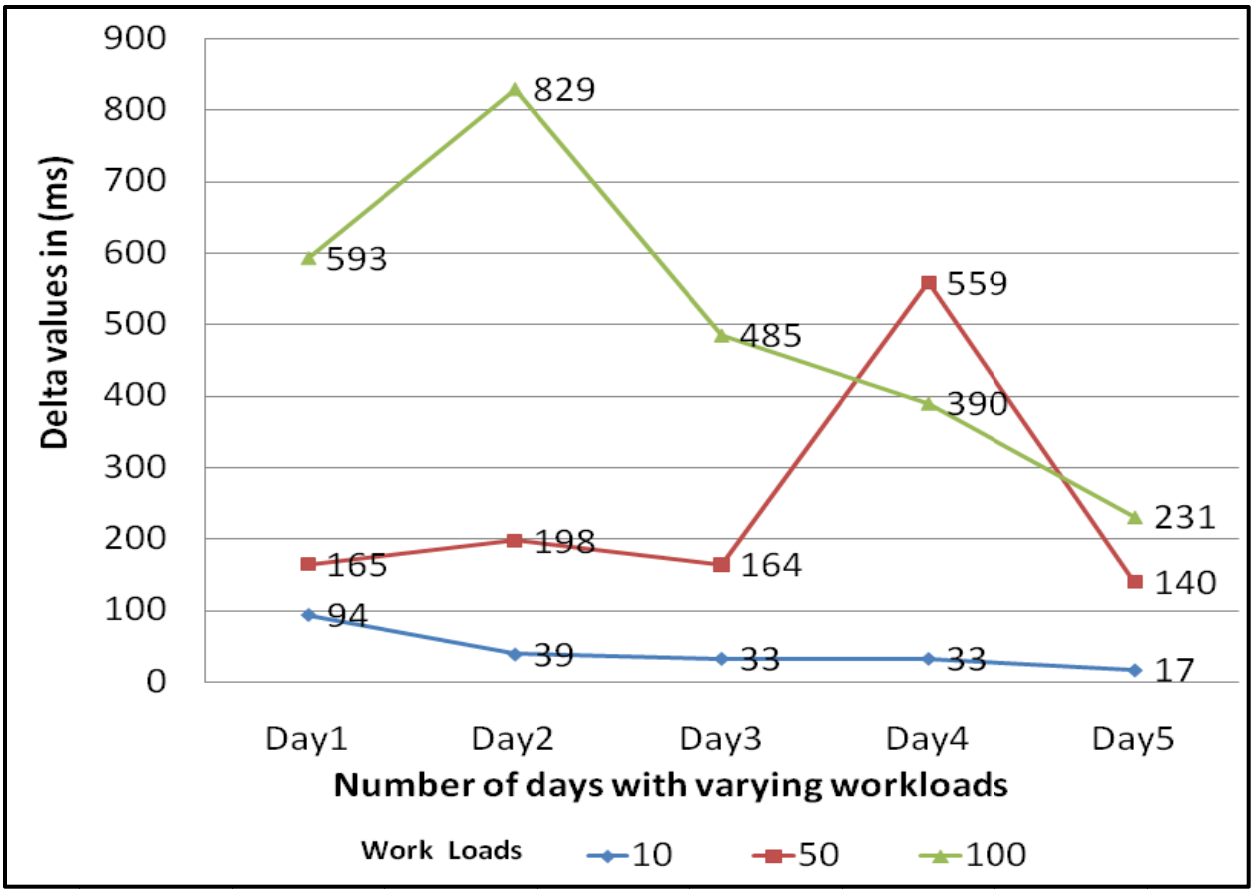


Figure 6-12. Difference of the times taken (ms) for main service in Python for workloads

Similarly the Figures 6.13, 6.14 and 6.15 show the performances of the accounts service using workloads. The Figure 6.16 shows the delta values for accounts service for all the workloads. For the accounts service, the peaks in the graphs are observed similar to main service for the workload 100 shown in 6.15. These loads are immediately balanced by sharing the load with the additional servers which loads the application and scales the requests.

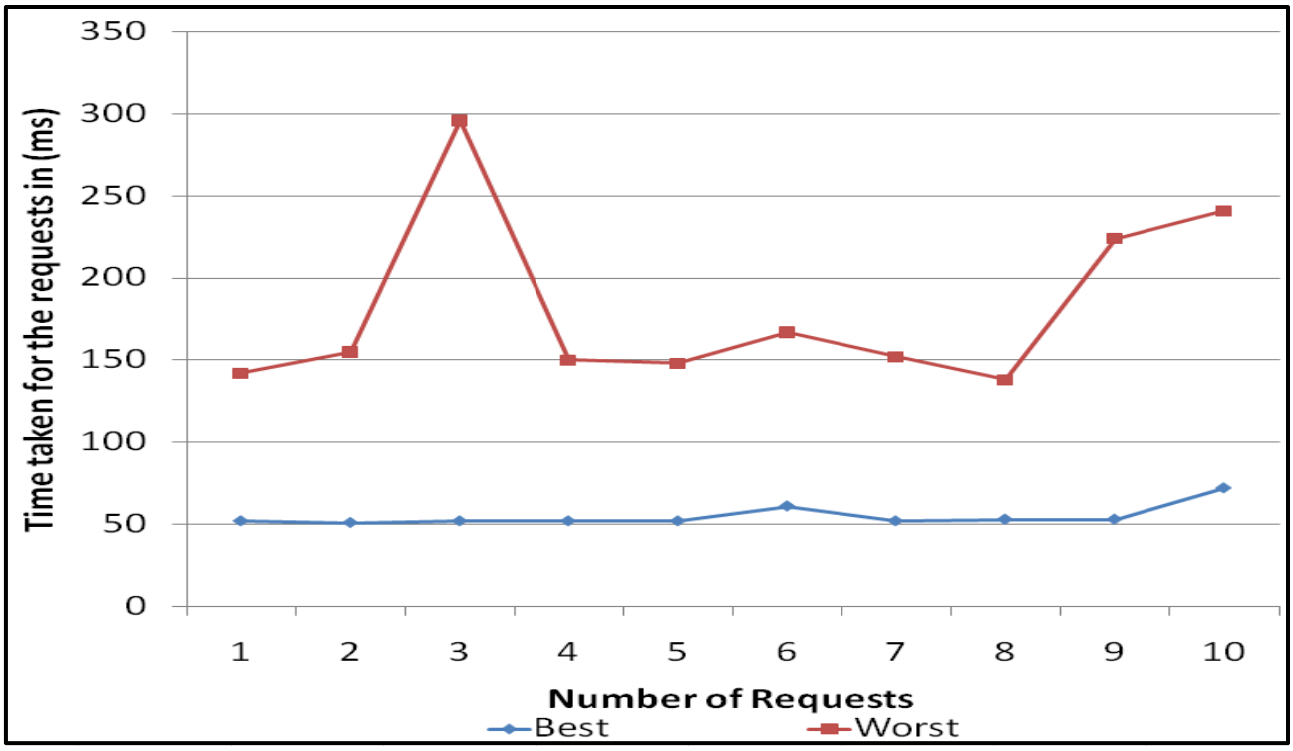


Figure 6-13. The performance of Python accounts service for workload (10)

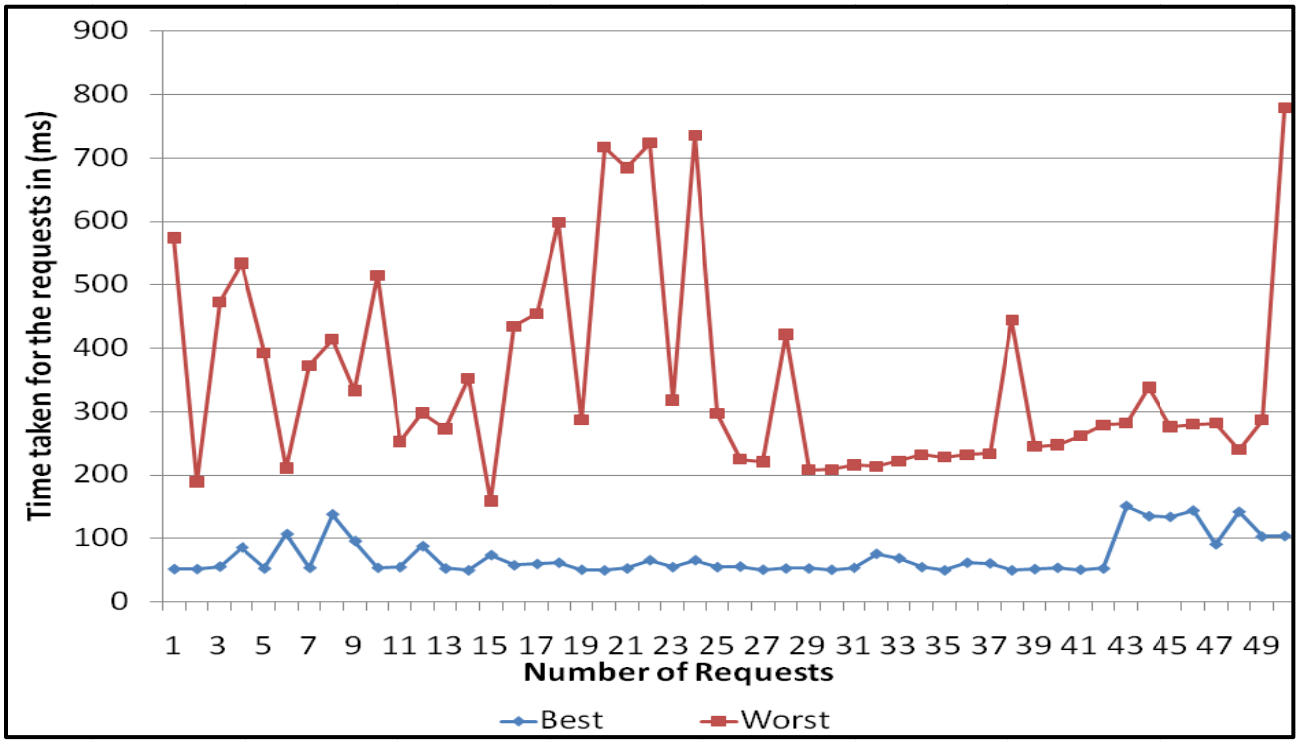


Figure 6-14. The performance of Python accounts service for workload (50)

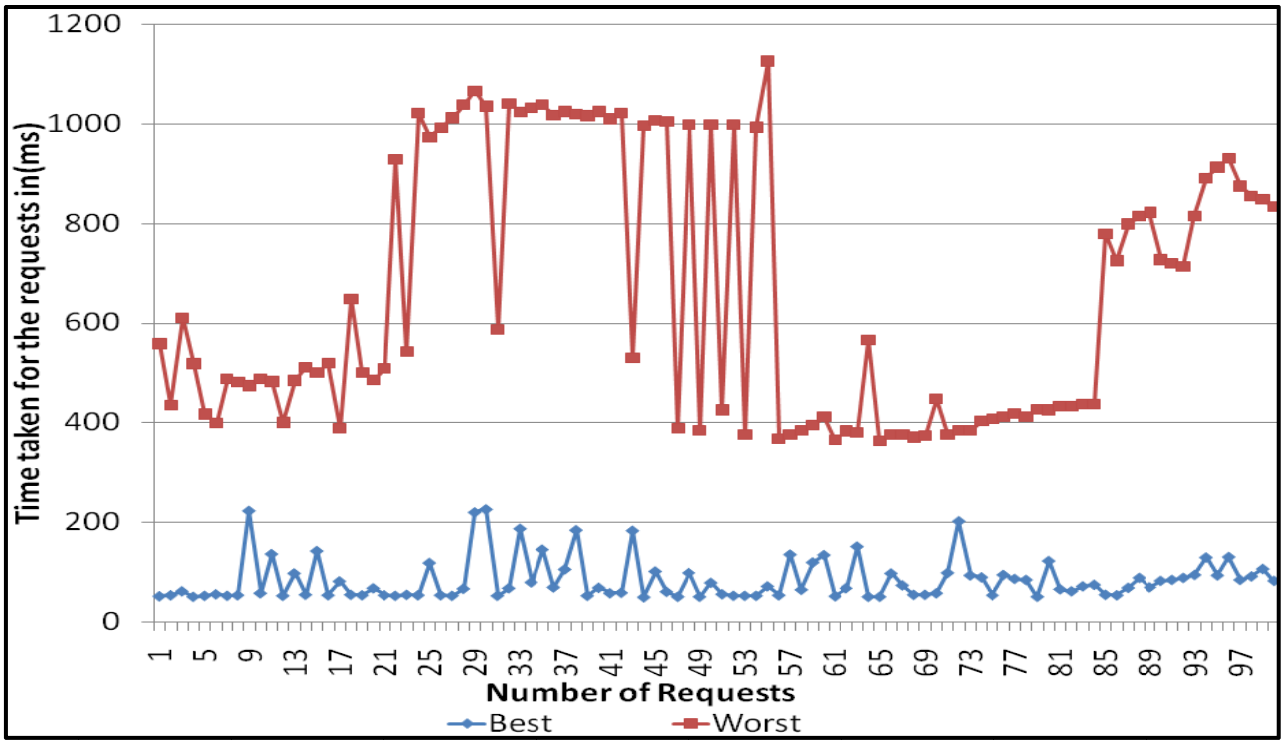


Figure 6-15. The performance of Python accounts service for workload (100)

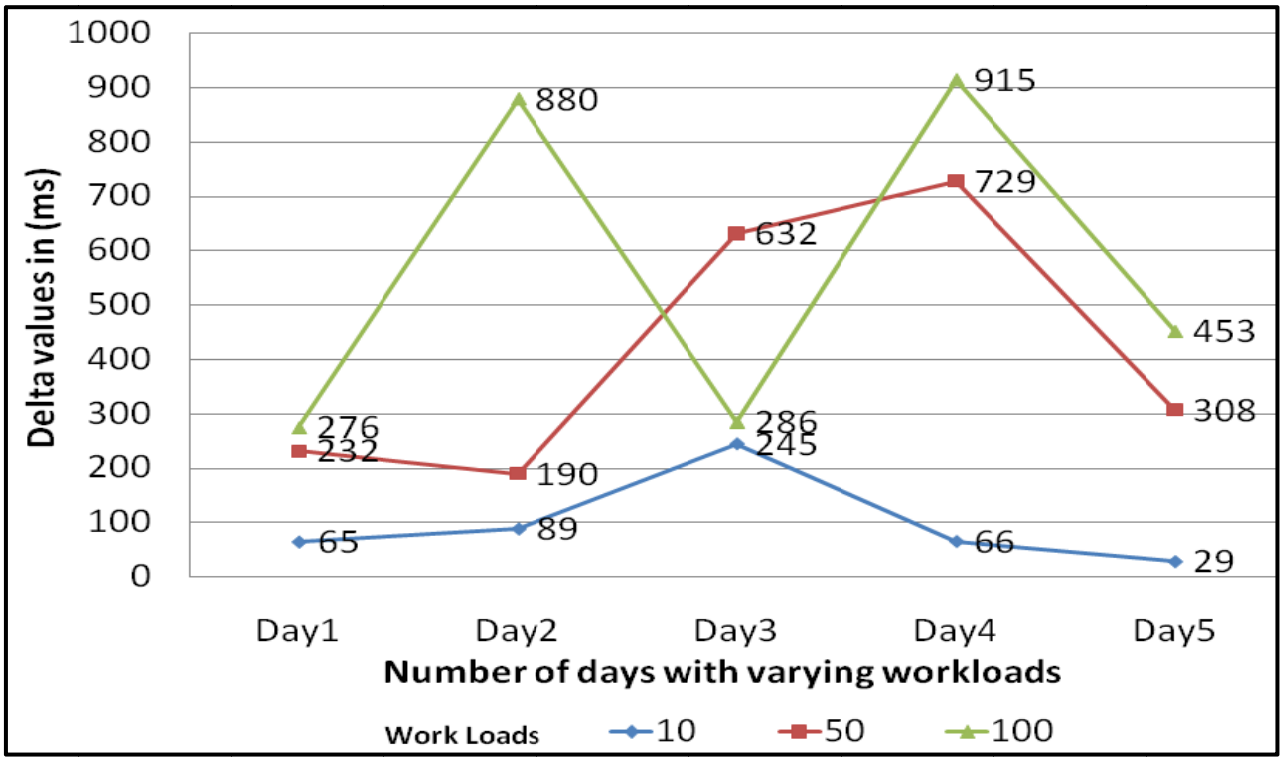


Figure 6-16. Difference of the times taken (ms) for accounts service in Python for workloads

The Figure 6-16 shows the delta values for the accounts service. The accounts service performs a GET request operation. In this graph the delta values show increase in numbers with increase in workloads but gradually decrease as new servers are fired up to balance the loads. Similarly the graphs are recorded for poststories service. The poststories service performs POST operation. The Figures 6-17 till 6-20 are the graphs for poststories service.

Figure 6-17. The performance of Python poststories service for workload (10)

The Figure 6-17 shows the time taken for the requests in worst line is in the same shape as the time taken for the best. This indicates that they are taking constant time with the worst graph taking twice the time of best.

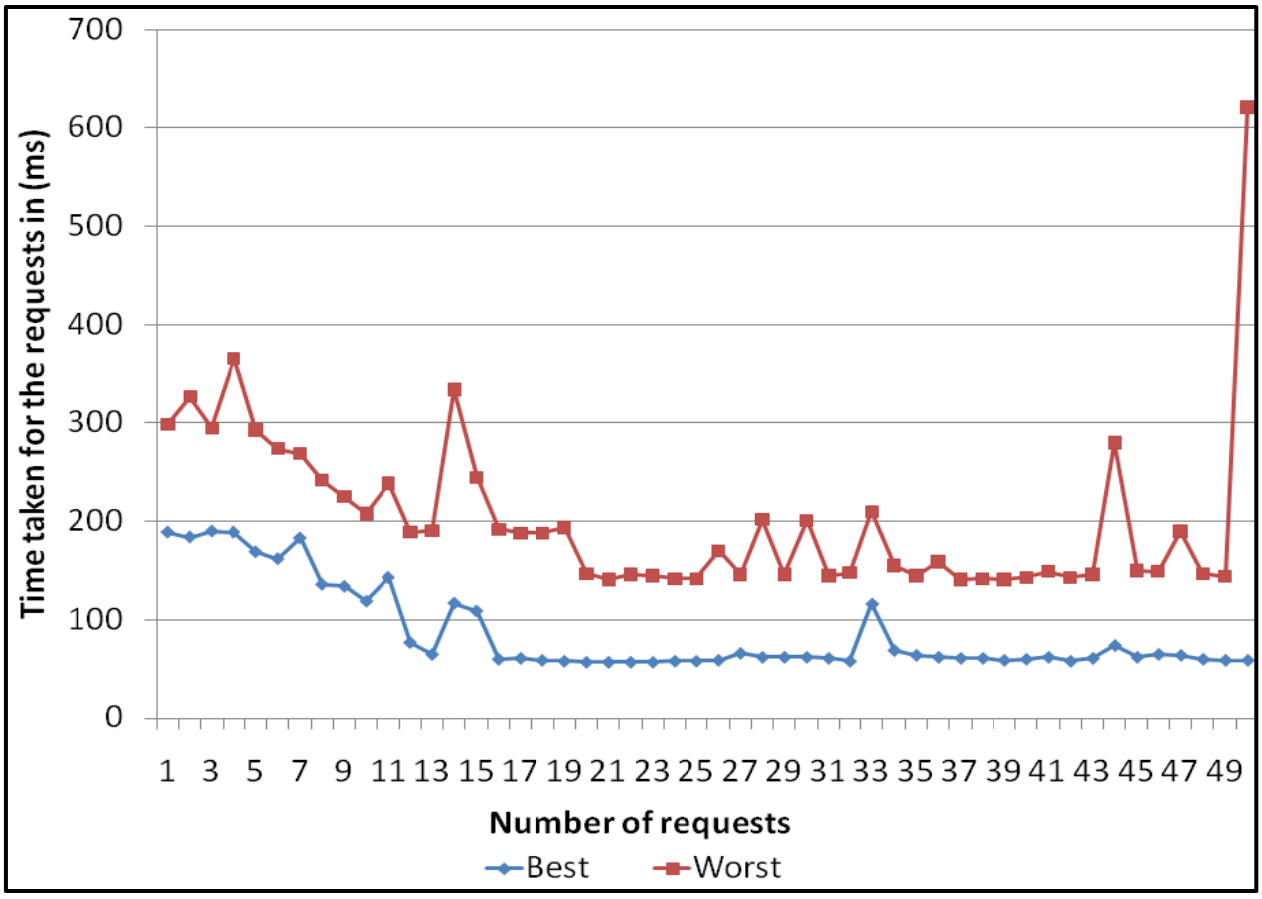


Figure 6-18. The performance of Python poststories service for workload (50)

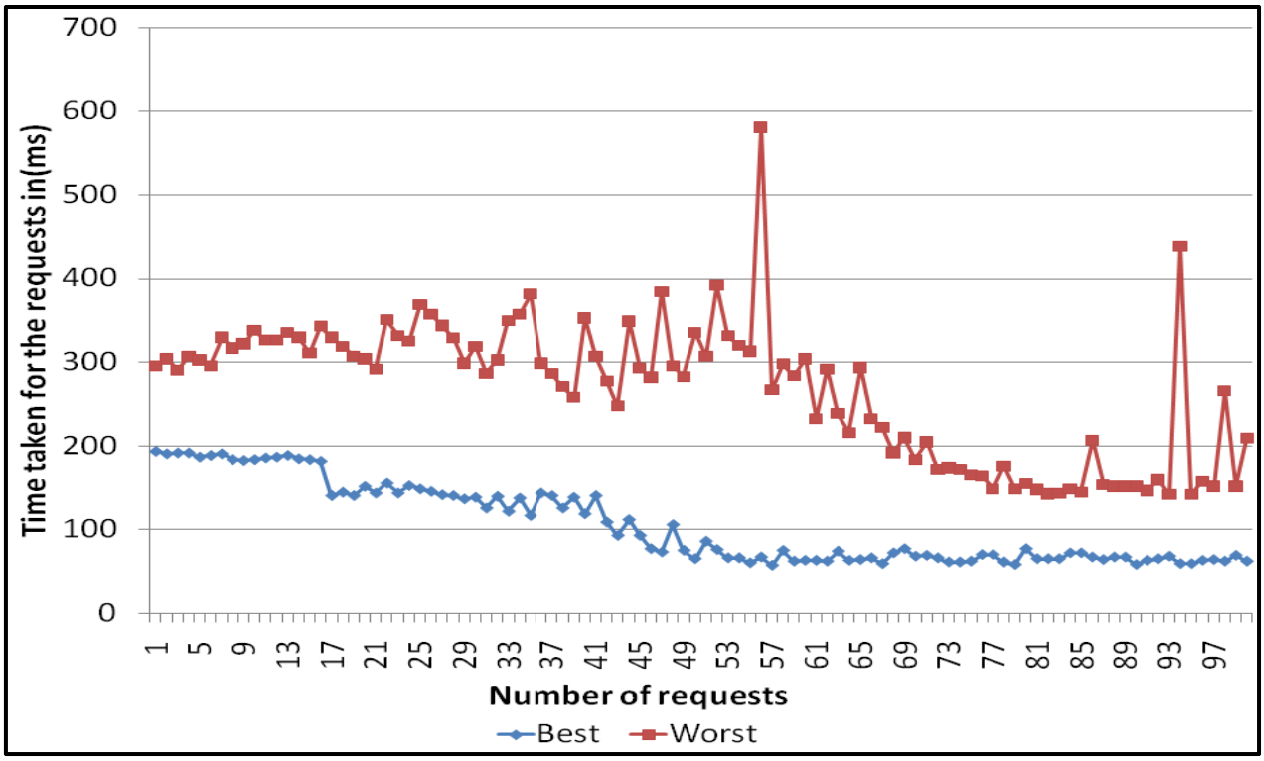


Figure 6-19. The performance of Python poststories service for workload (100)

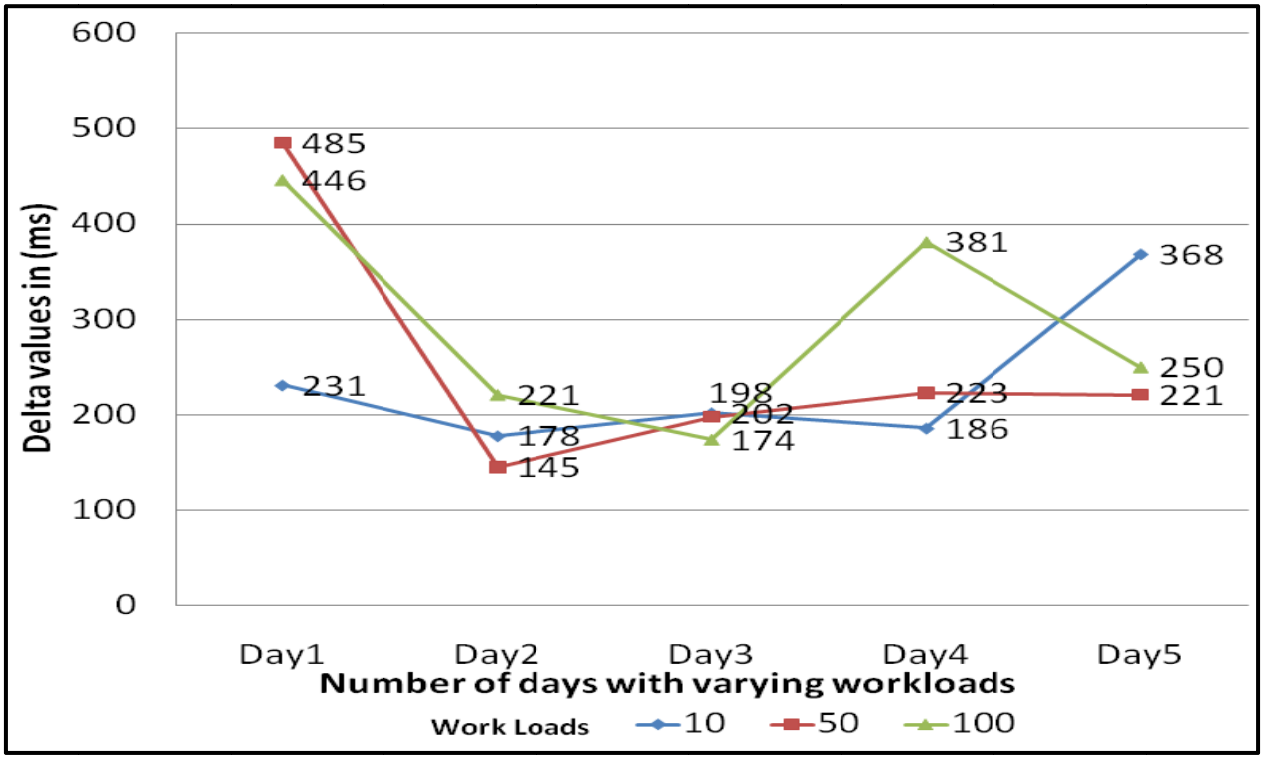


Figure 6-20. Difference of the times taken (ms) for poststories service in Python for workloads

The graphs for the comments service in Python are similar. The comments service is a POST service. The Figures 6-21 till 6-23 are the graphs indicating the best and worst performances of the graphs for workloads. The Figure 6-24 indicates the difference between the performance of the comments service for workloads on each day.

Figure 6-21. The performance of Python comments service for workload (10)

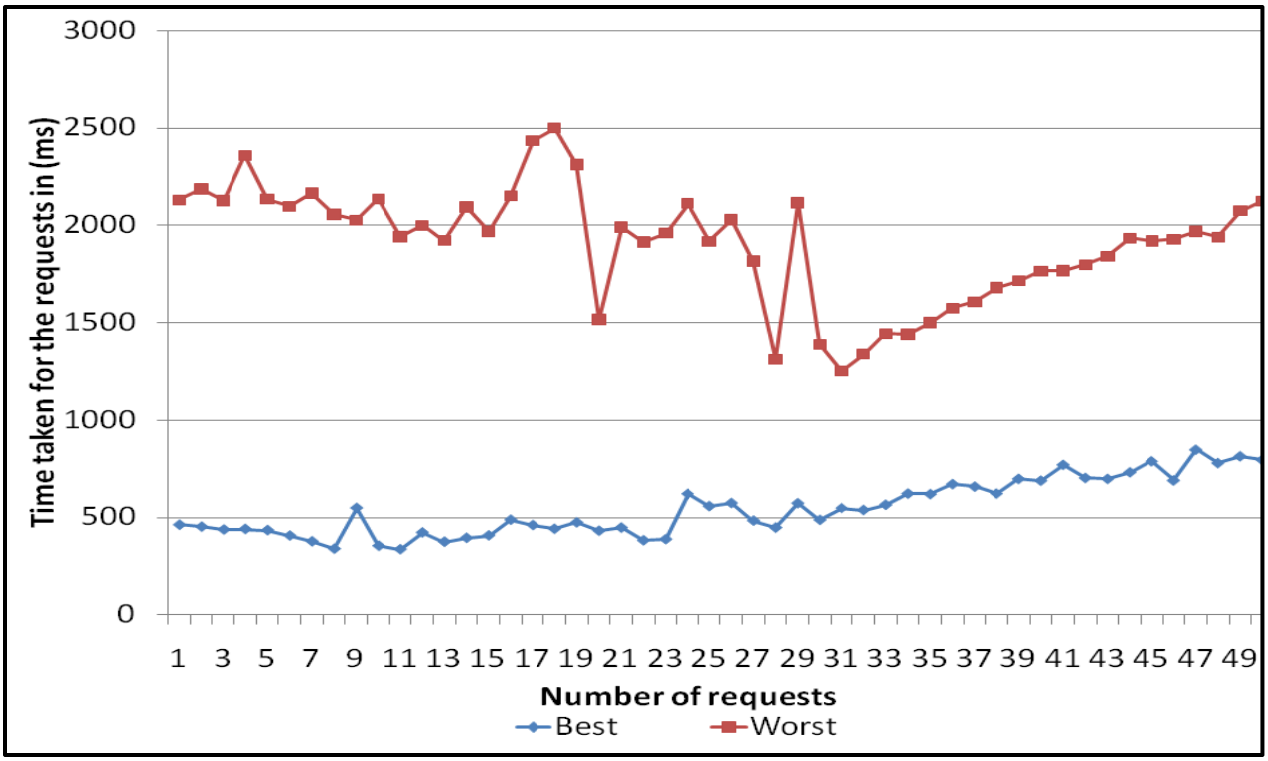


Figure 6-22. The performance of Python comments service for workload (50)

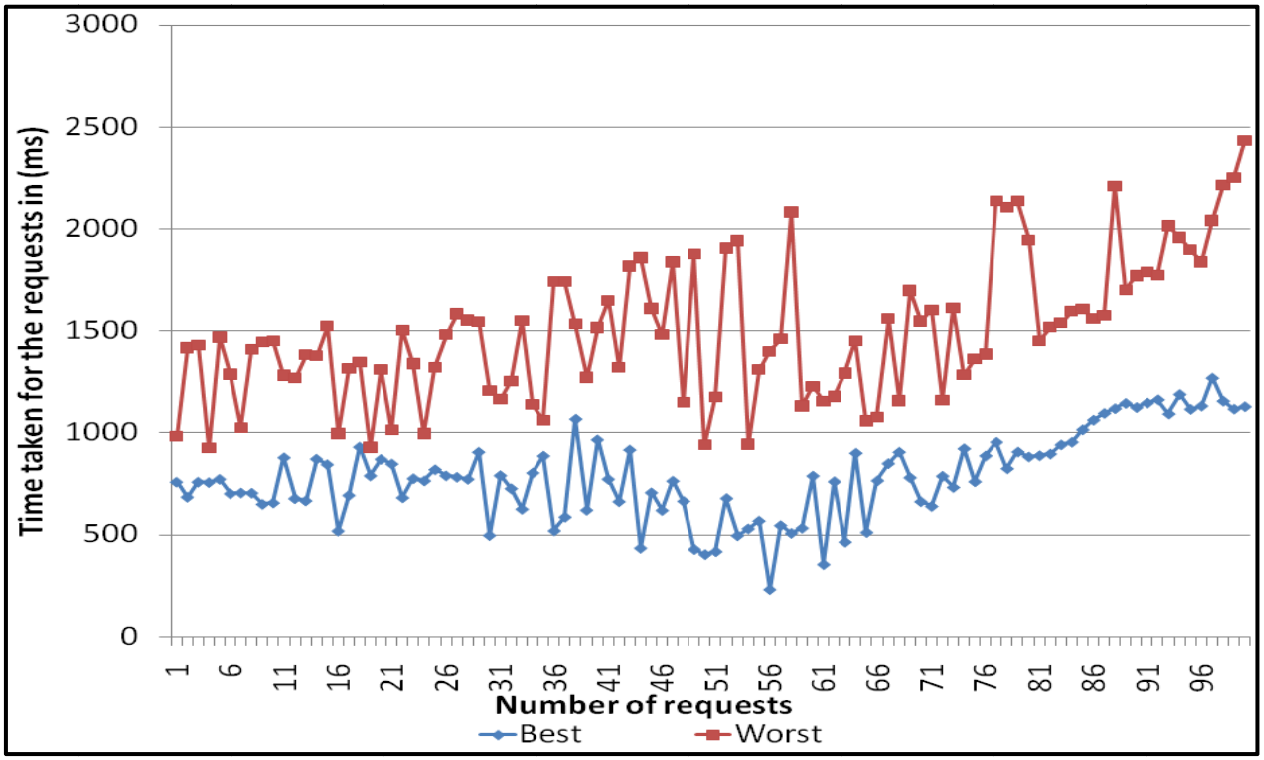


Figure 6-23. The performance of Python comments service for workload (100)

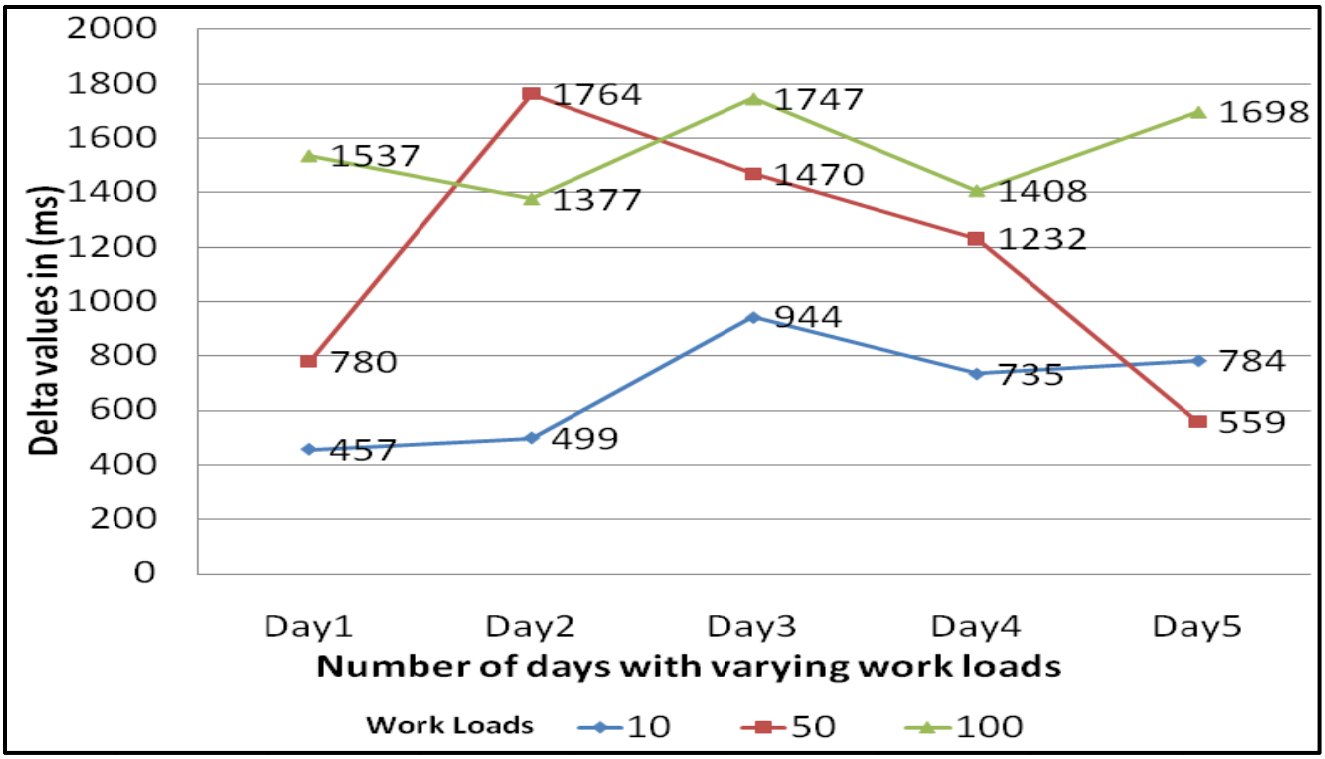


Figure 6-24. Difference of the times taken (ms) for comments service in Python for workloads

Results of the Java services. The evaluation for Java services is similar to the Python services. The Figures 6.25 till Figure 6.44 shows the graphs related to performance of the services and the difference between the maximum and minimum values of the time taken for 5 different services in Java experiment indicated as Delta values.

Table 6-3. Table showing the list of services evaluated using Java

Service name	Request Type
Login	POST
Posting	POST
Rating	POST
Main	GET
Viewing	GET

The Figures 6.25 through 6.27 shows the performance of the login service for workloads (10, 50, and 100) in Java. The Figure 6.25 shows huge difference between the worst case performance and the best performance. The Figure 6.25 shows patterns similar to the graphs in Python experiments but takes slightly more time than the time taken to process the requests in Python language.

The login service in Java is a POST operation. For example, the maximum time taken for the Python login service for 10 requests is 320 milliseconds as shown in Figure 6.5 where as for the Java login it takes 8010 milliseconds. It is very high when compared to the times taken for the Python services. From the Figures 6-25 till Figure 6-27 (for login service in Java) the maximum time taken is above 8000 ms. In comparison with the Python services the Java services take longer time to process the requests. The delta values of the Java login service with different workloads is shown in Figure 6-28.

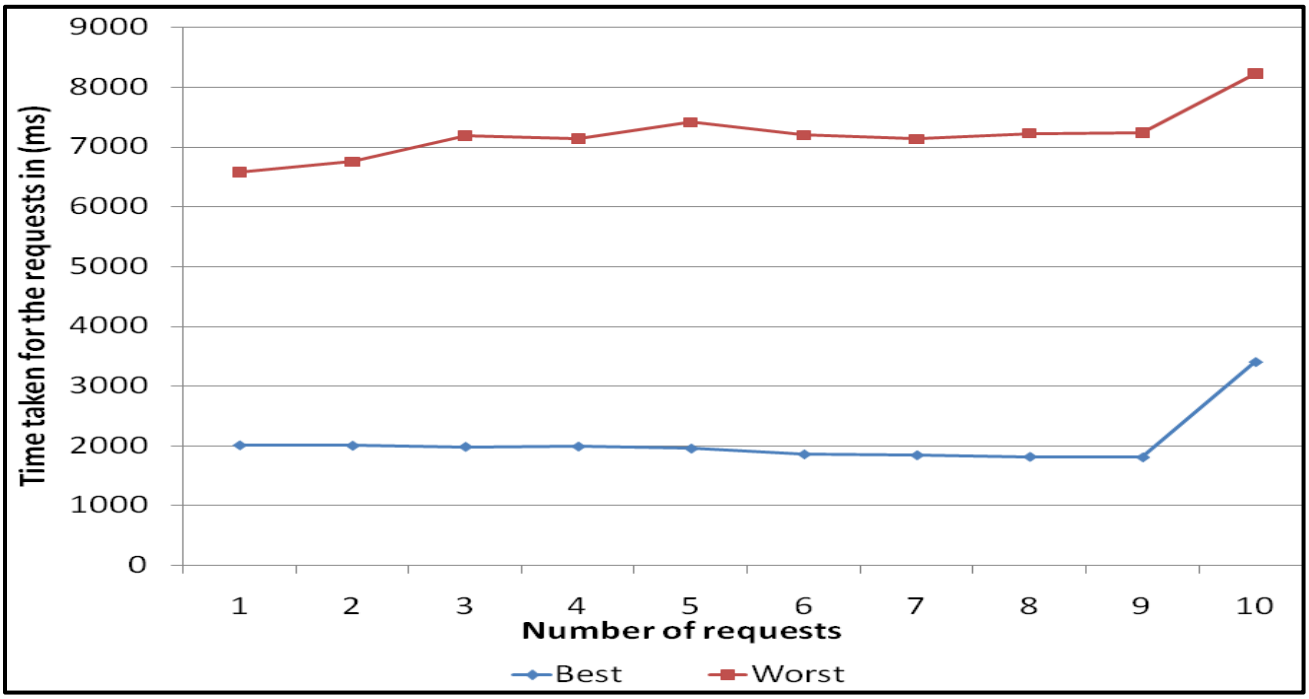


Figure 6-25. The performance of the Java login service for workload (10)

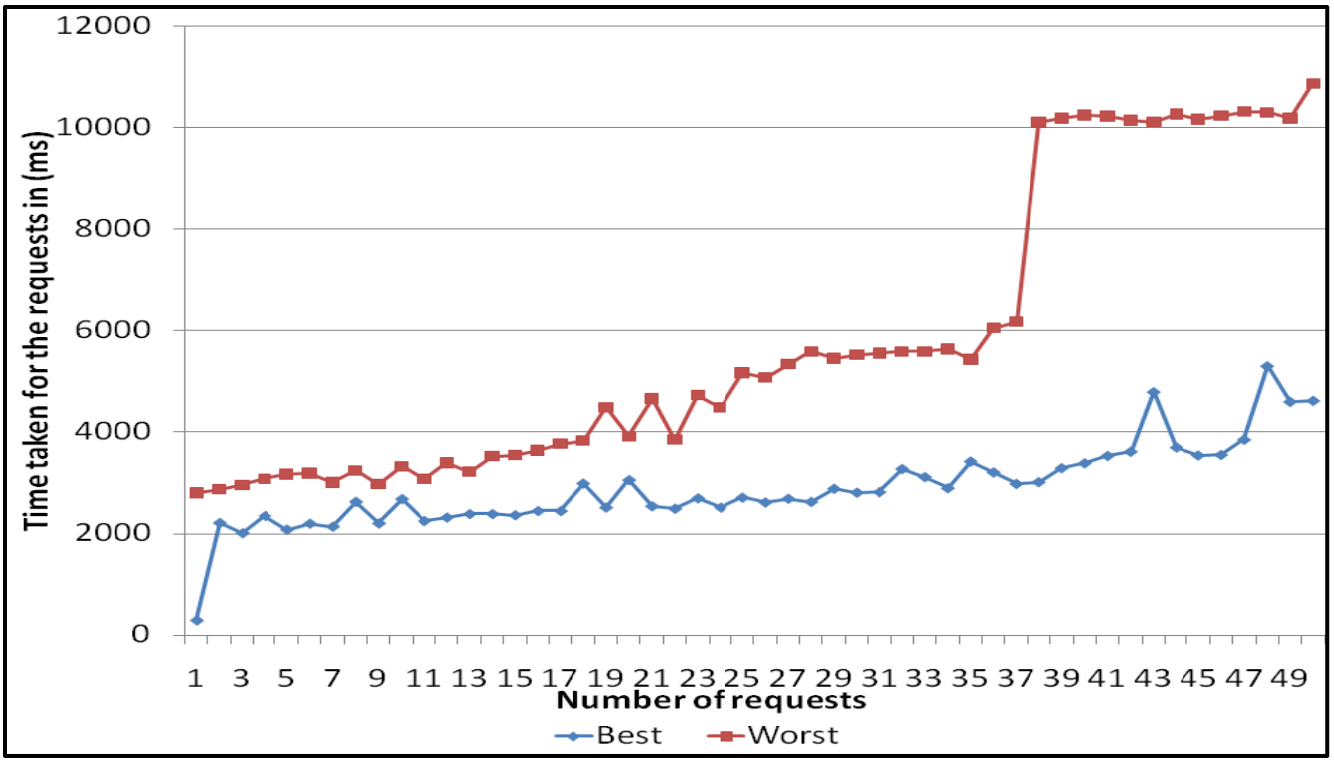


Figure 6-26. The performance of Java login service for workload (50)

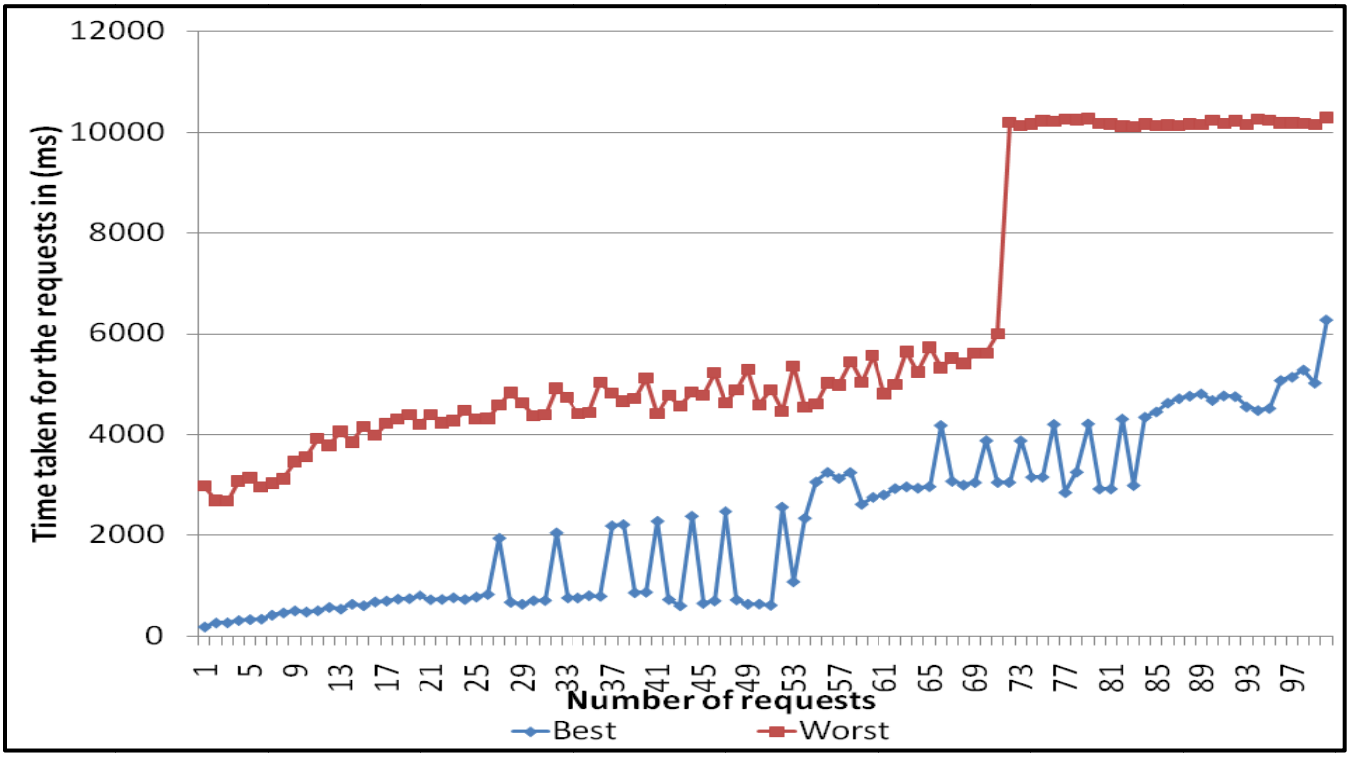


Figure 6-27. The performance of Java login service for workload (100)

As discussed above, almost all the services in Java take longer time to process the requests when compared to the Python services. The Figure 6-28 shows that delta value of the time taken for workloads 50 and 100 increases on Thursday and Friday (day4 and day5) because of the increase in the load. This is because that memory gets occupied with the Java objects that are created. These objects occupy all the GAEs horizontal memory storage. So for a new JDO request it takes some time to clear up the existing objects with the help of Java language garbage collector.

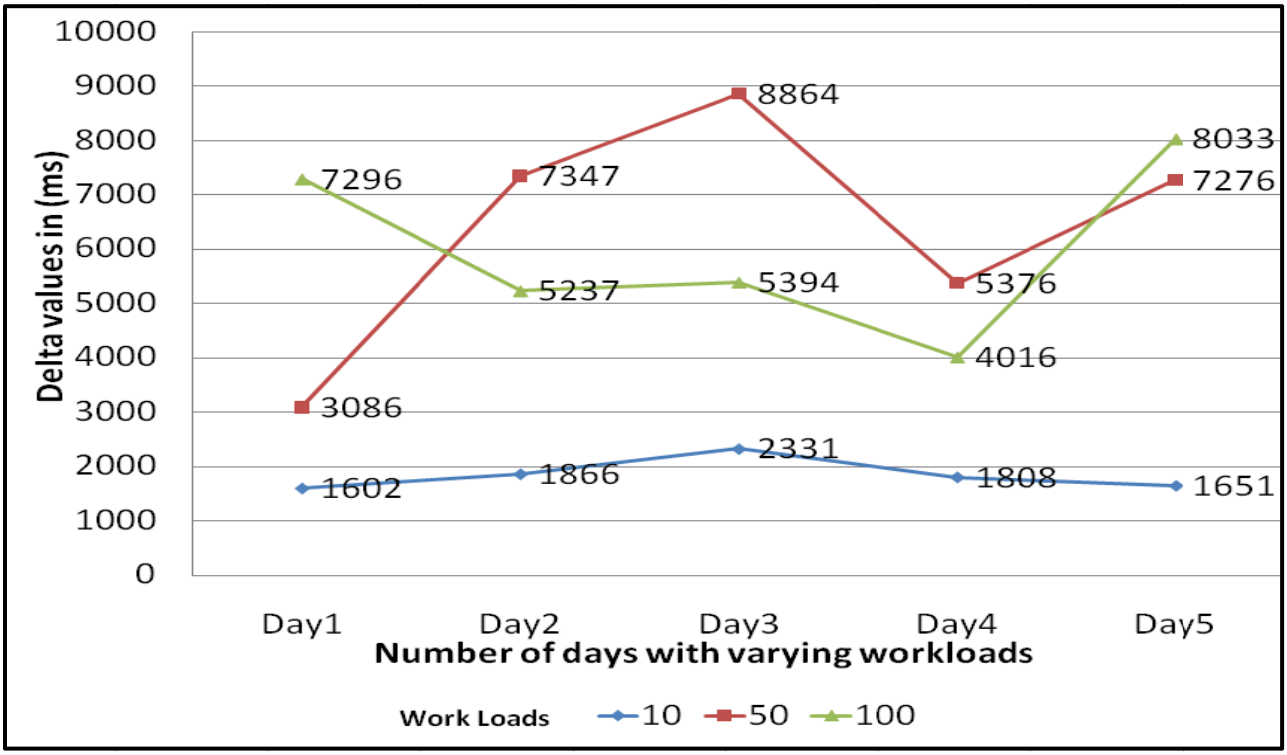


Figure 6-28. Difference of the times taken (m-s) for login service in Java for workloads

The Figures 6-29 till Figure 6-31 show the performance of the posting service for workloads (10, 50, and 100). The posting service is a POST operation and takes longer time for processing the requests when compared to the Python based poststories service which performs the same operation. Oelhman [57] states that the performance of the GAE is decreased when

processing the Java services when compared with the Python services. The Figure 6-32 shows the delta values of the workloads from day1 to day5 for workloads.

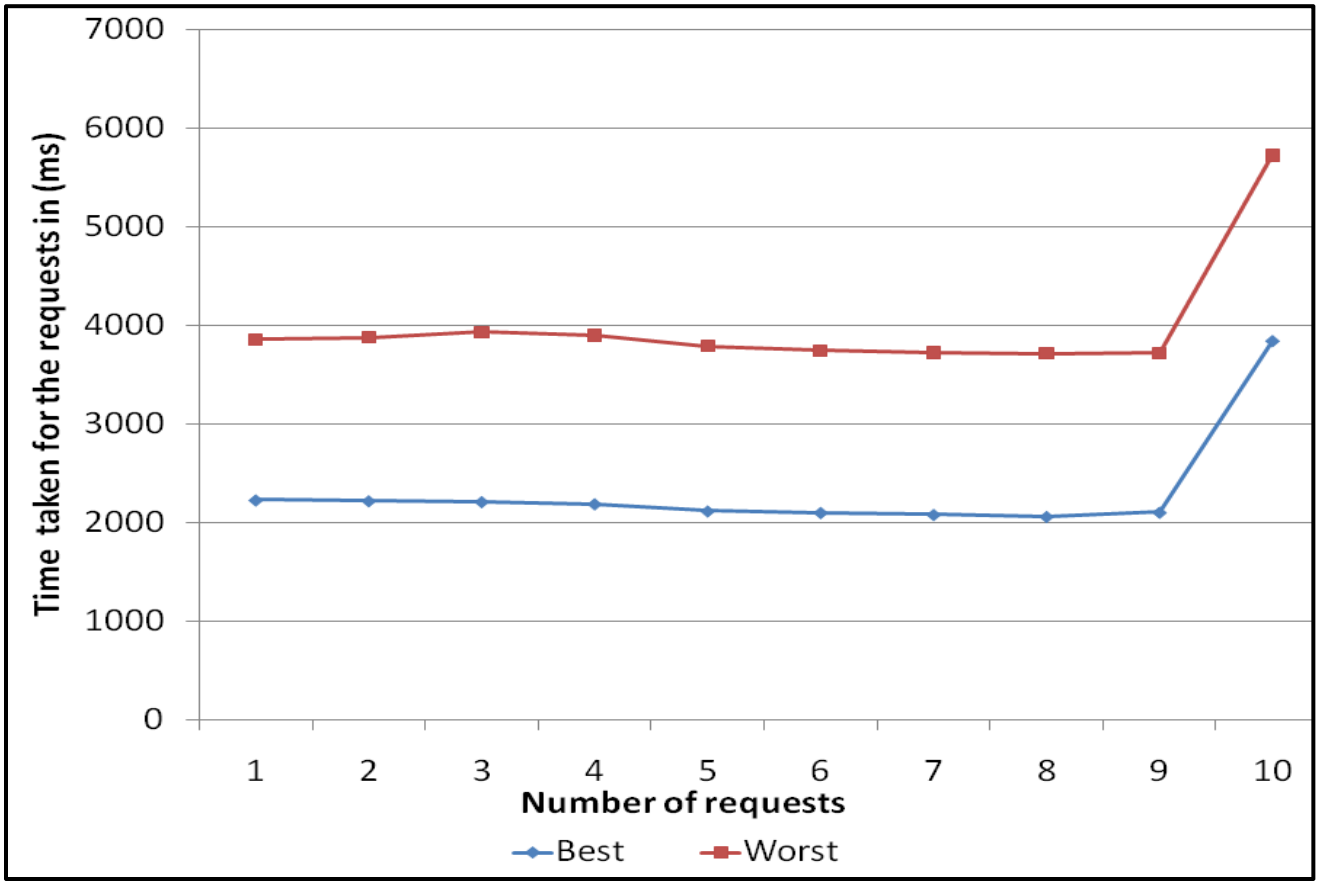


Figure 6-29. The performance of the Java posting service with workload (10)

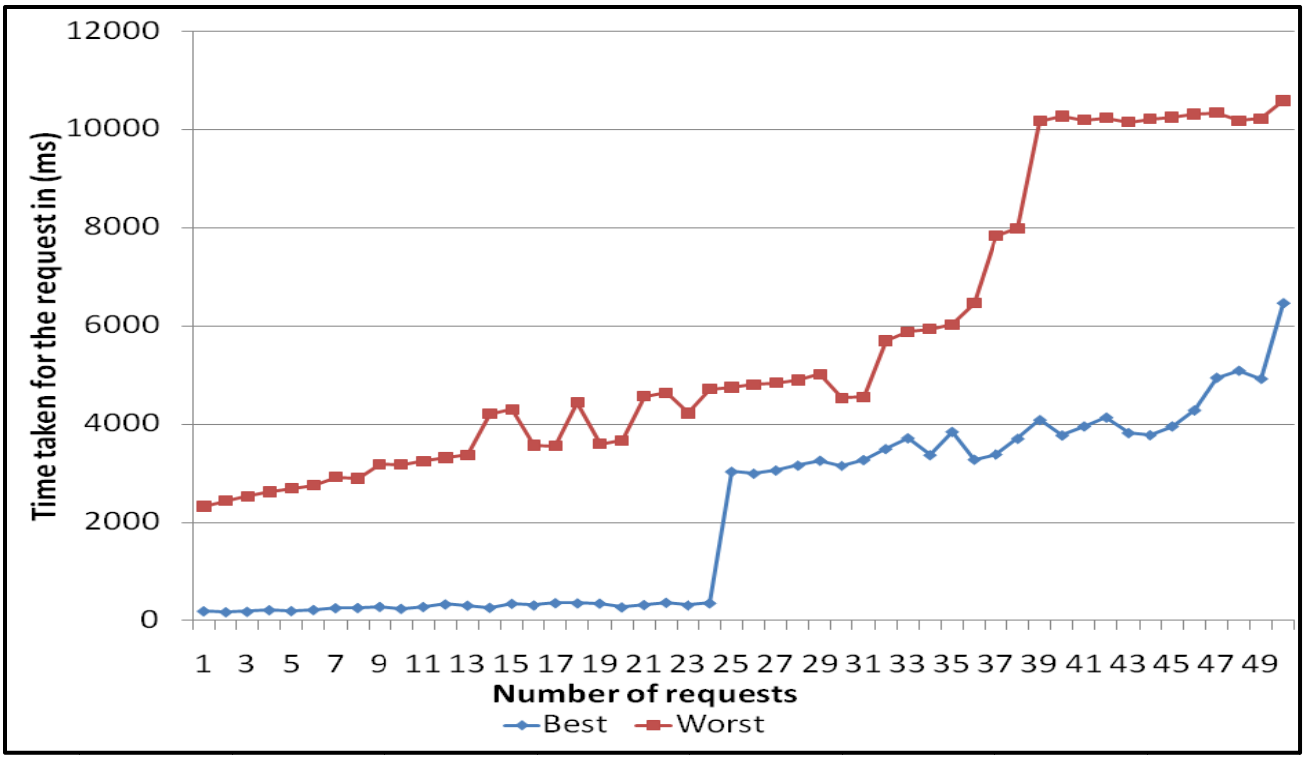


Figure 6-30. The performance of the Java posting service for workload (50)

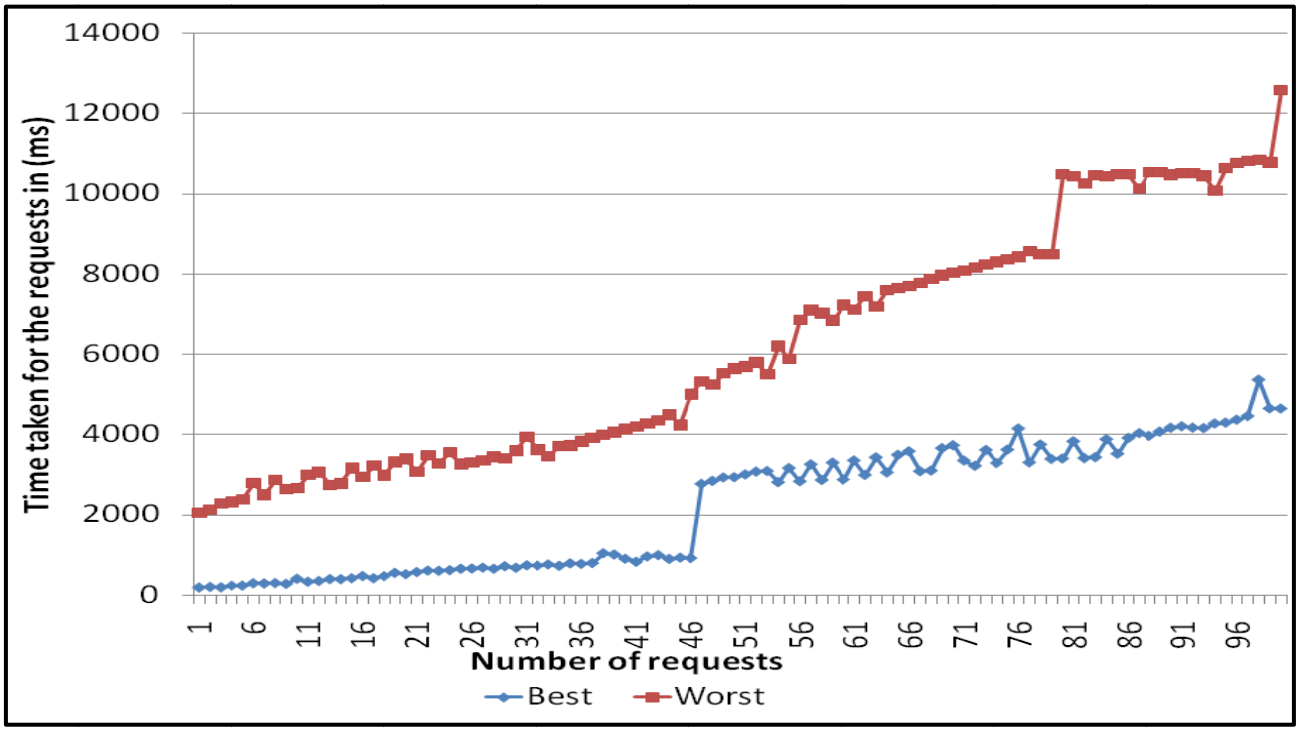


Figure 6-31. The performance of Java posting service for workload (100)

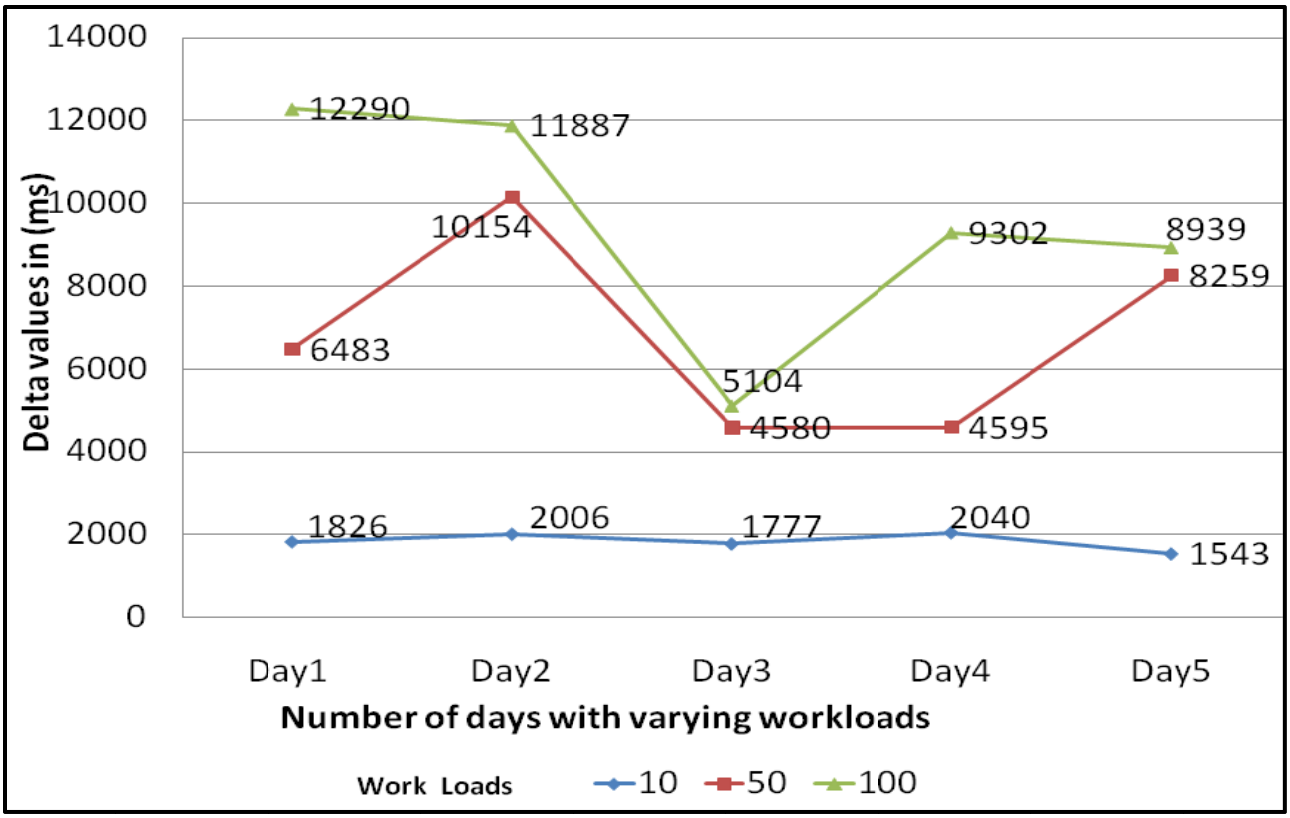


Figure 6-32. Difference of the times taken (ms) for the posting service in Java for workloads

Similarly, the Figure 6.33 till figure 6.44 shows the graphs for the time taken for workloads with ratings, main, and viewing services respectively. The ratings service is a POST operation and main, viewing services perform GET operations on the server. These services approximately take twice the time when compared with the services in Python. The graphs show that the overall time taken for all the services in Python is less when compared with the time taken by Java services. The Figure 6-36 shows the delta values for the workloads for the ratings service.

The Figures 6.33 till Figure 6.35 show the performance of ratings service for workloads (10, 50, and 100). The Figure 6.36 shows the delta values for workloads 10, 50 and 100 as shown below for the ratings service.

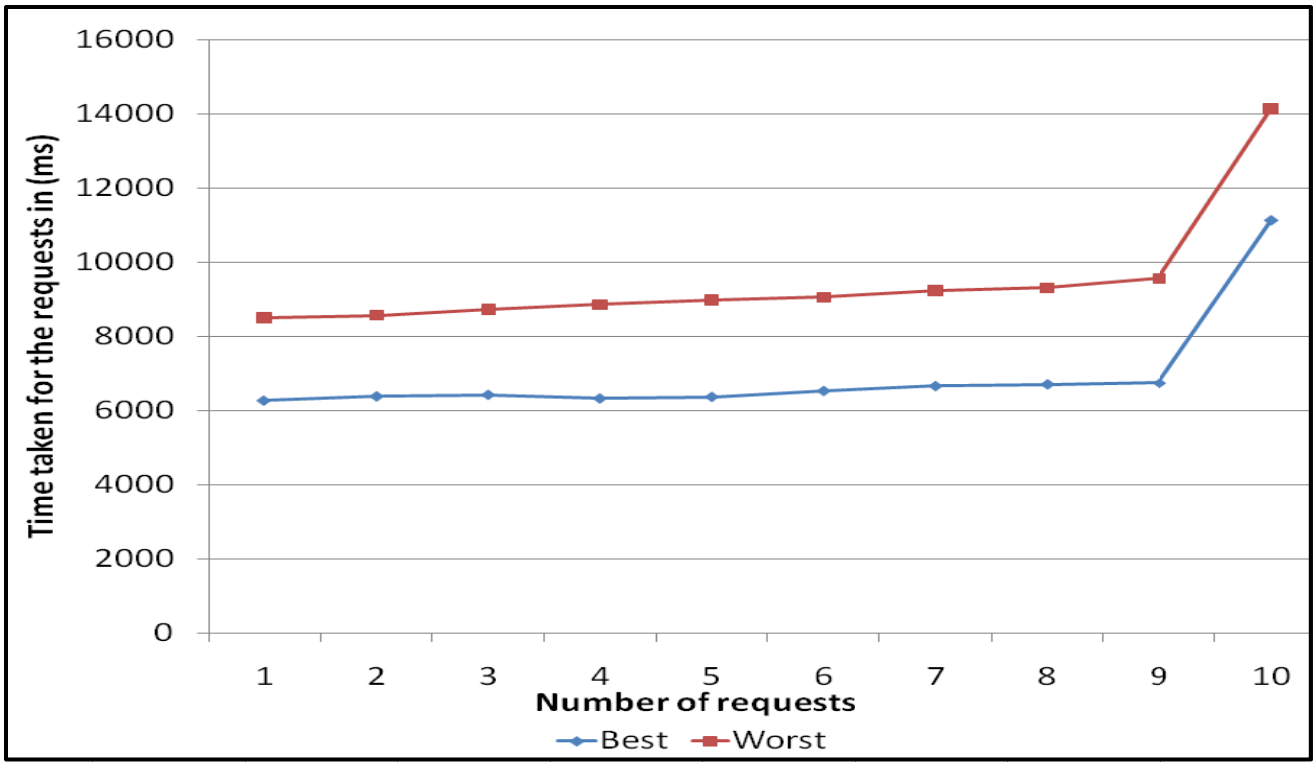


Figure 6-33. The performance of the Java ratings service for workload (10)

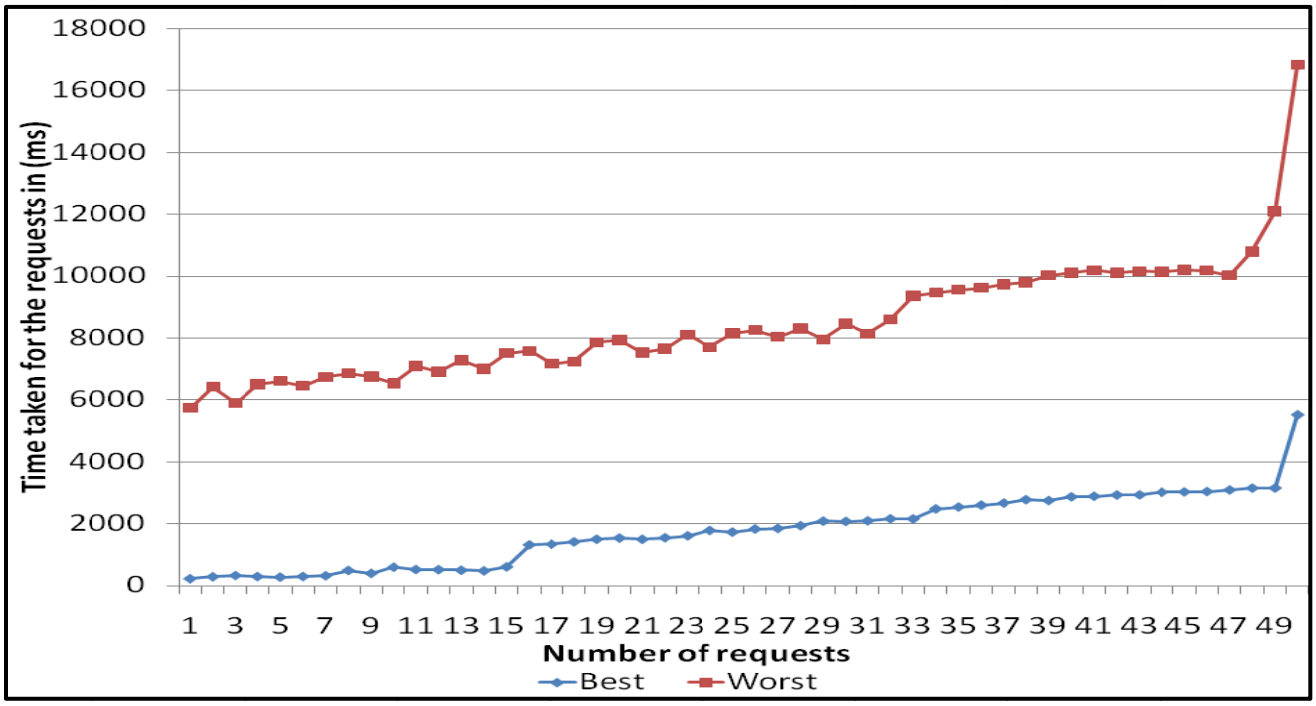


Figure 6-34. The performance of the Java ratings service for workload (50)

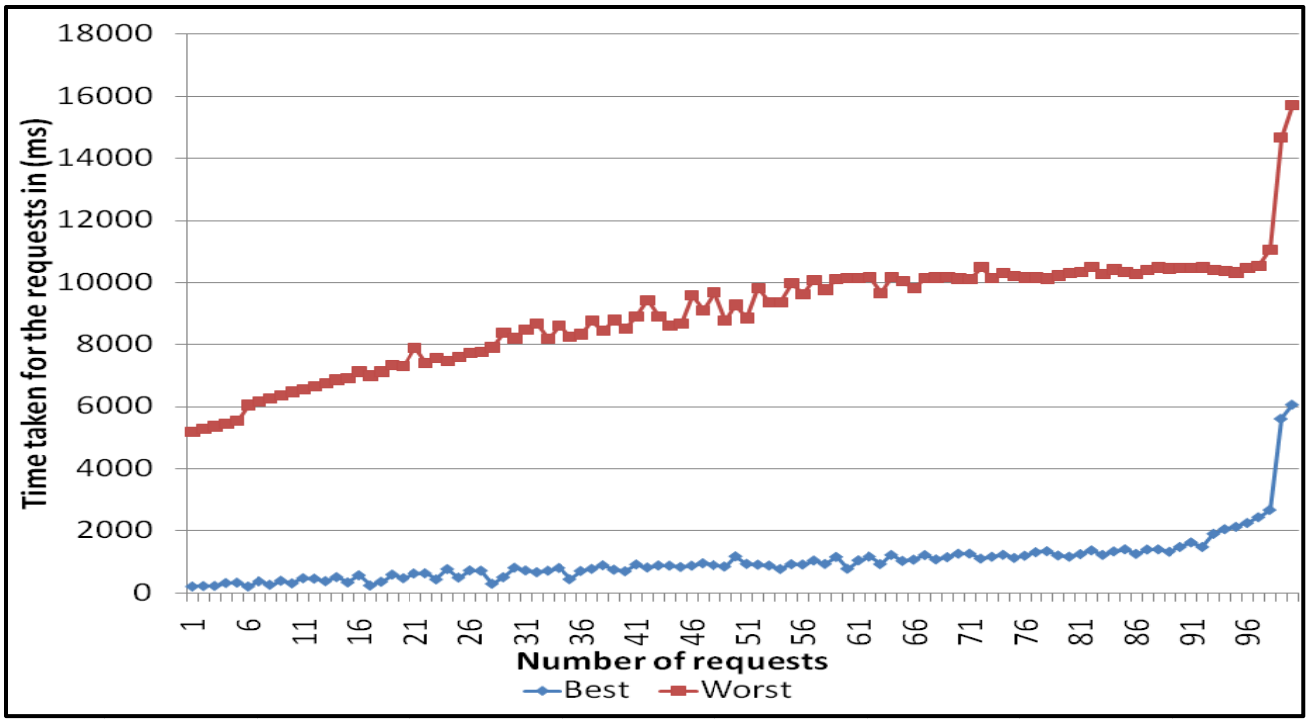


Figure 6-35. The performance of the Java ratings service for workload (100)

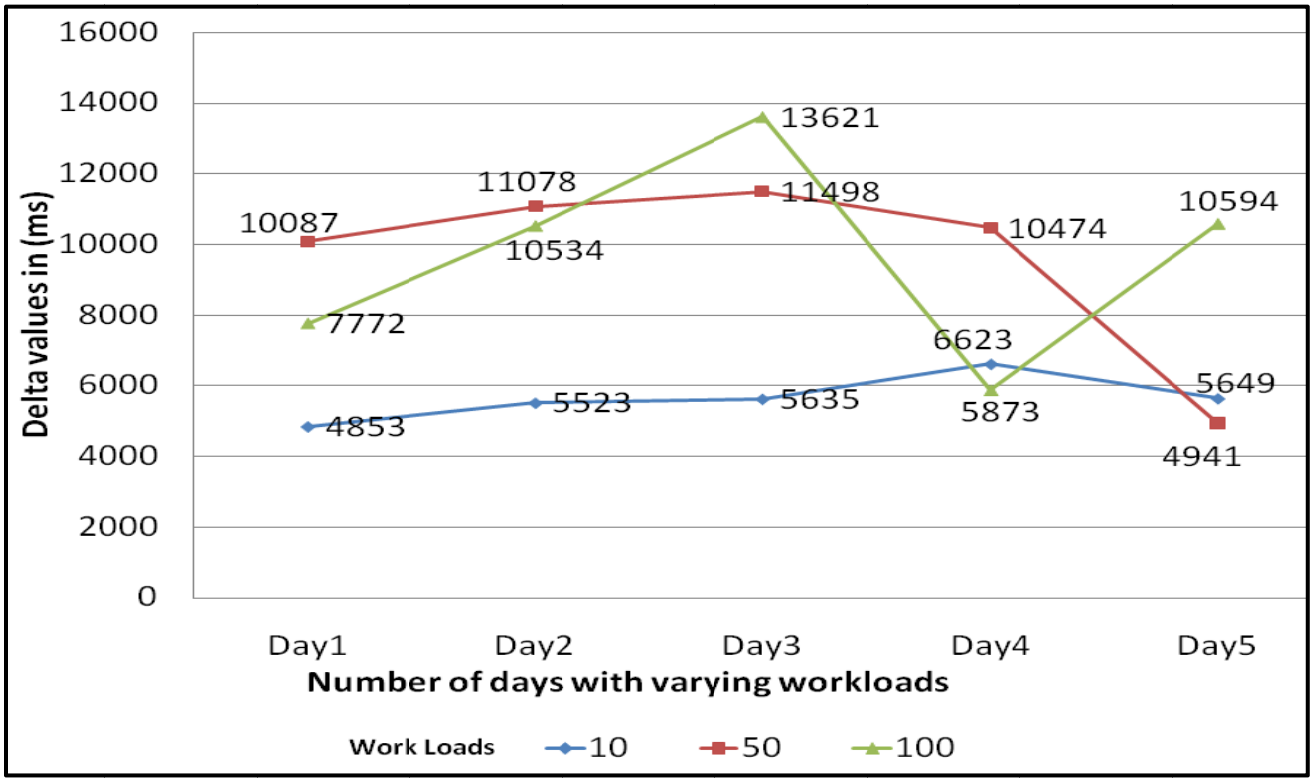


Figure 6-36. Difference of the times taken (ms) for the ratings service in Java for workloads

The Figure 6-37 to Figure 6-40 are the graphs for main service. The main service is a GET service.

Figure 6-37. The performance of the Java main service for workload (10)

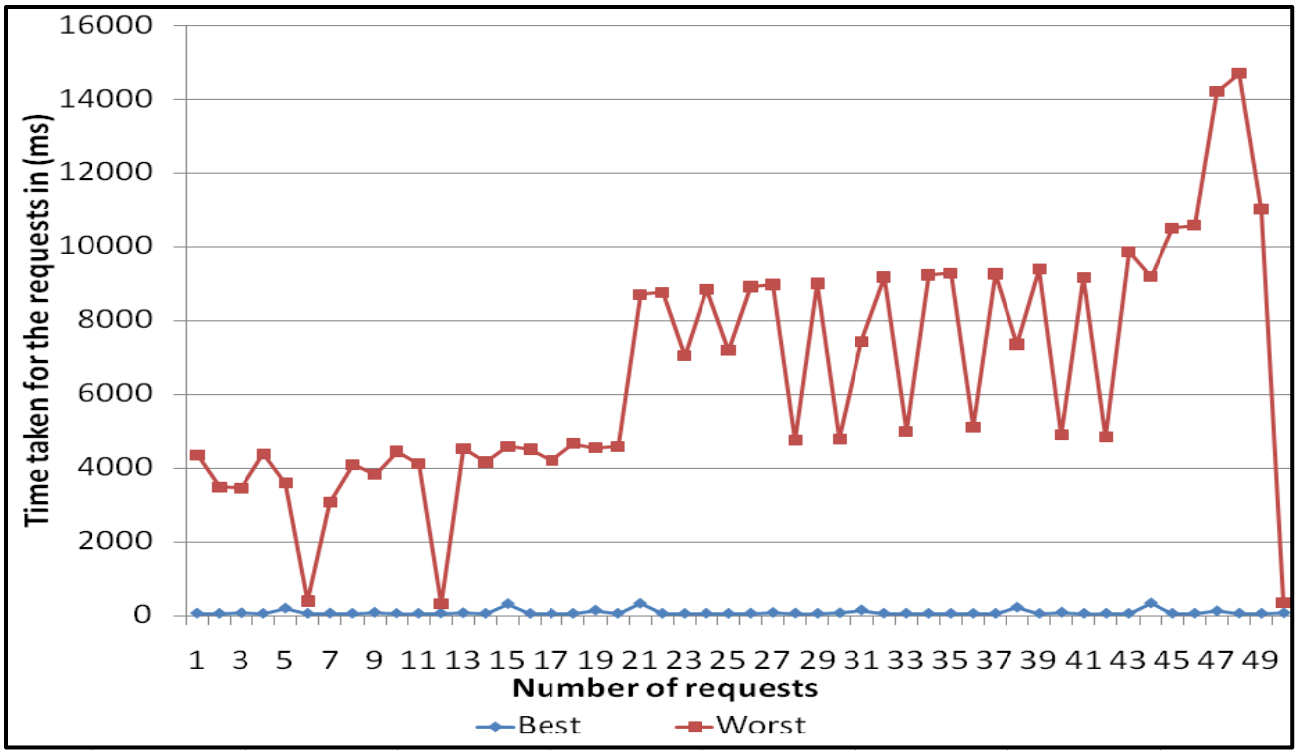


Figure 6-38. The performance of the Java main service for workload (50)

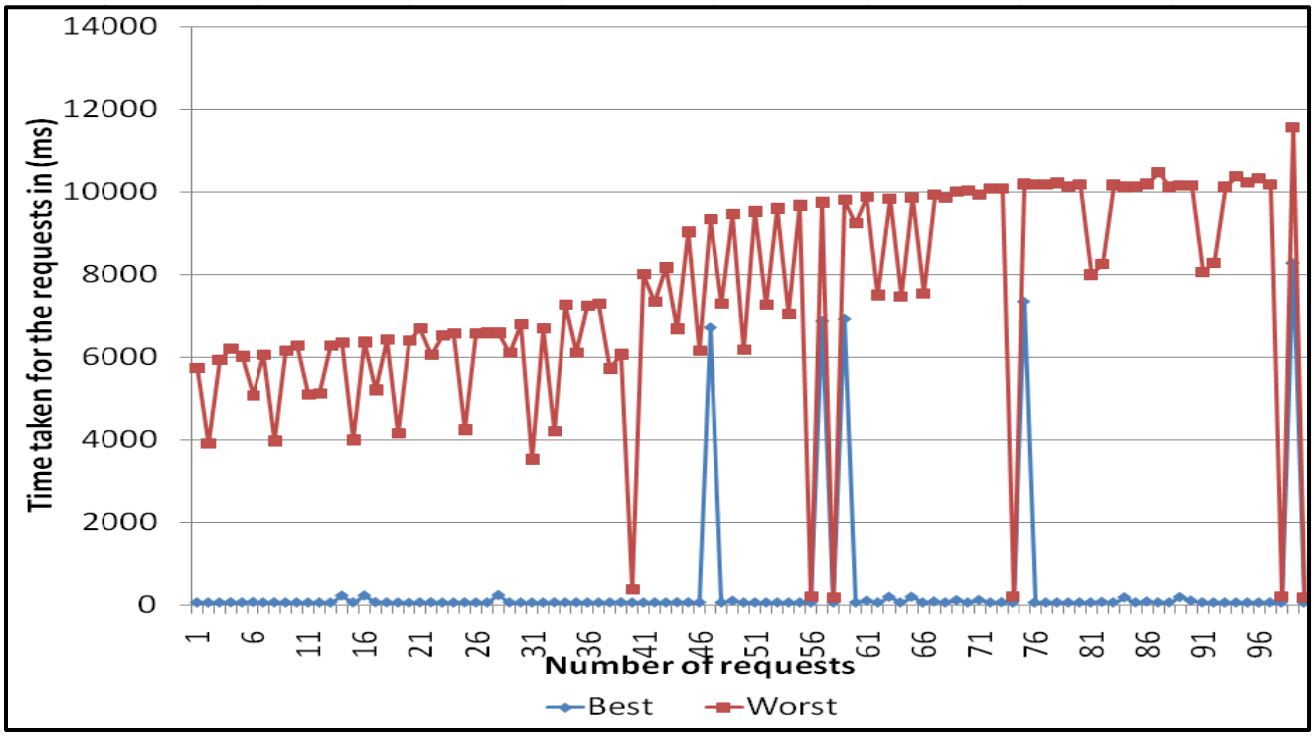


Figure 6-39. The performance of the Java main service for workload (100)

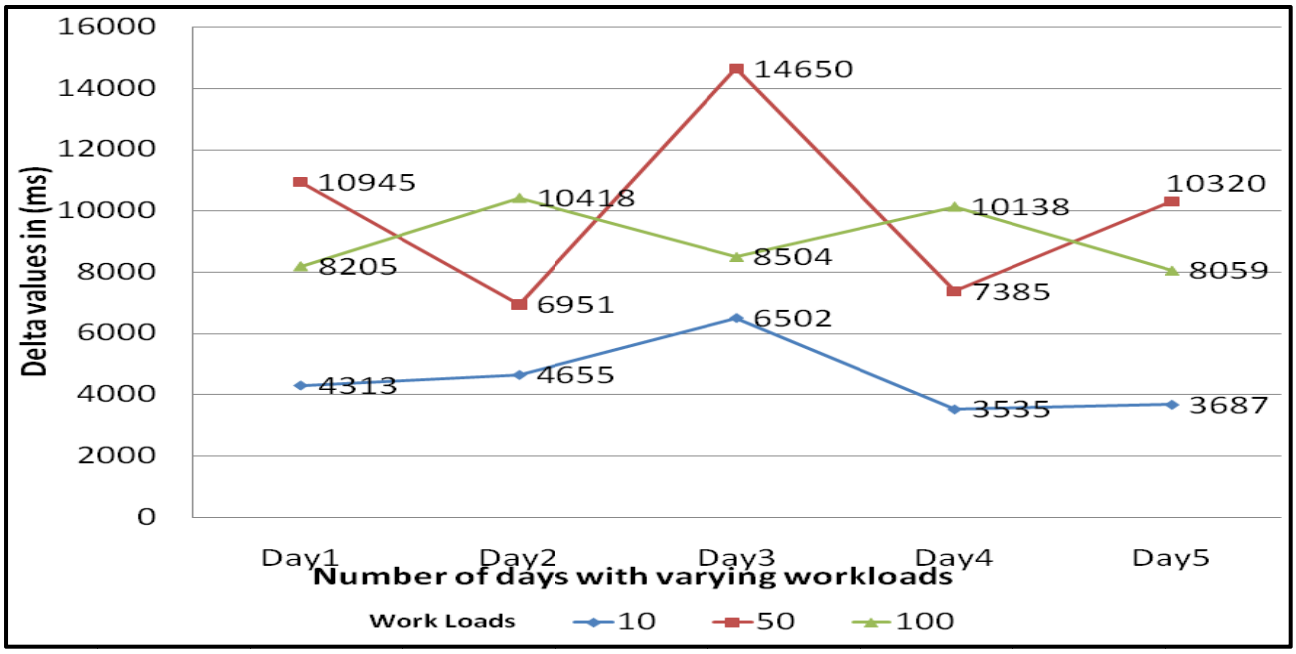


Figure 6-40. Difference of the times taken (ms) for main service in Java for workloads

Similarly graphs are recorded for the viewing service. This is service is used to view the stories posted in the application. It is a GET operation based service.

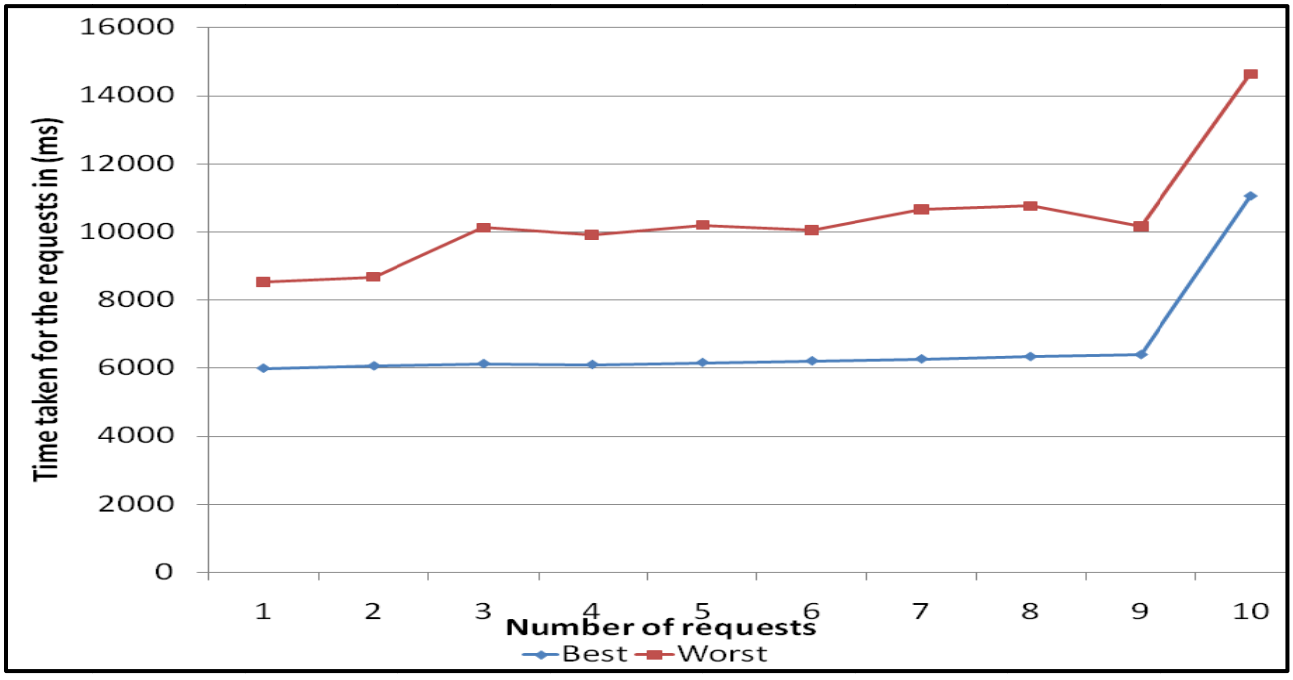


Figure 6-41. The performance of the Java viewing service for workload (10)

The figures show the time taken for processing the workloads and the delta values for workloads.

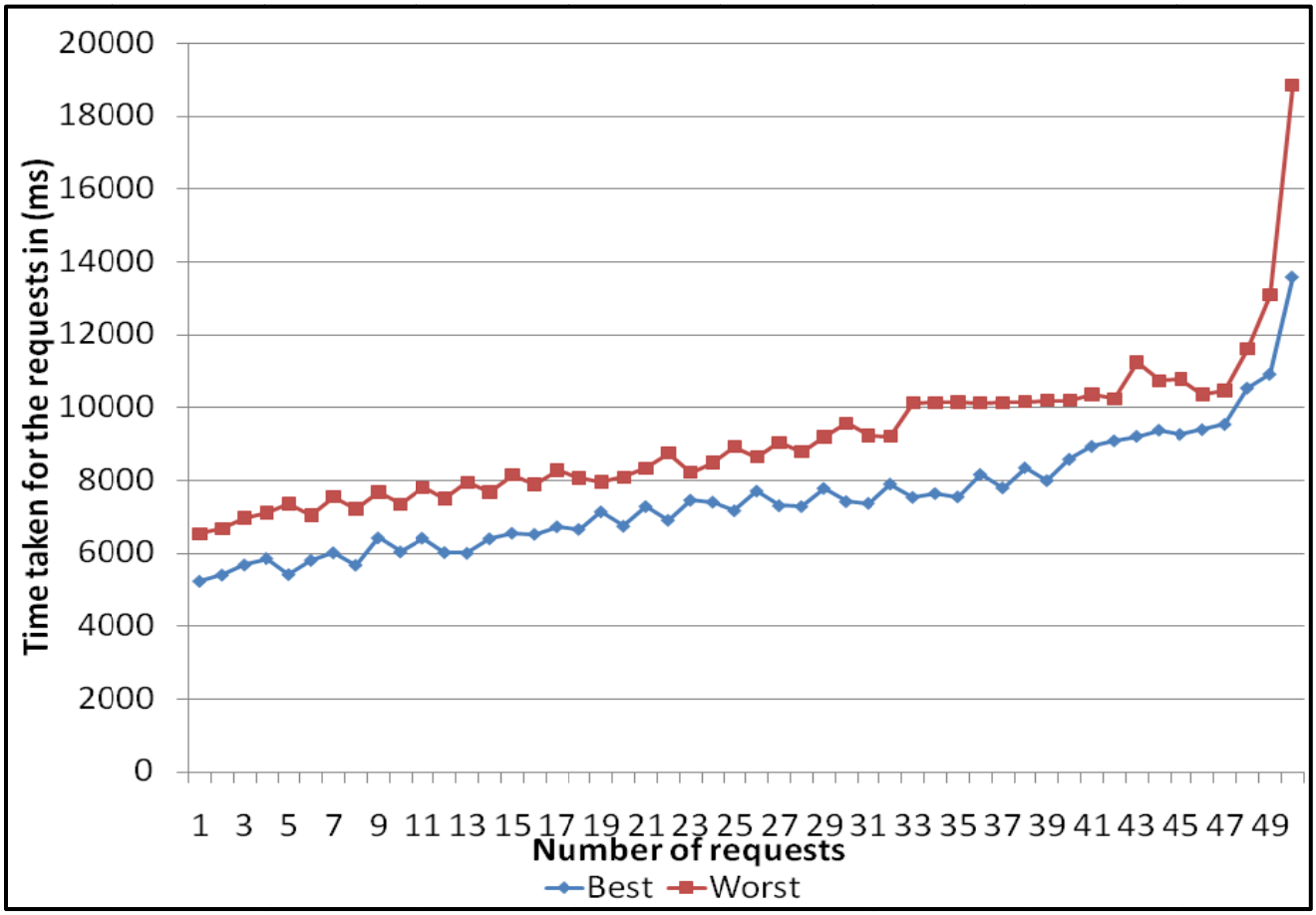


Figure 6-42. The performance of the Java viewing service for workload (50)

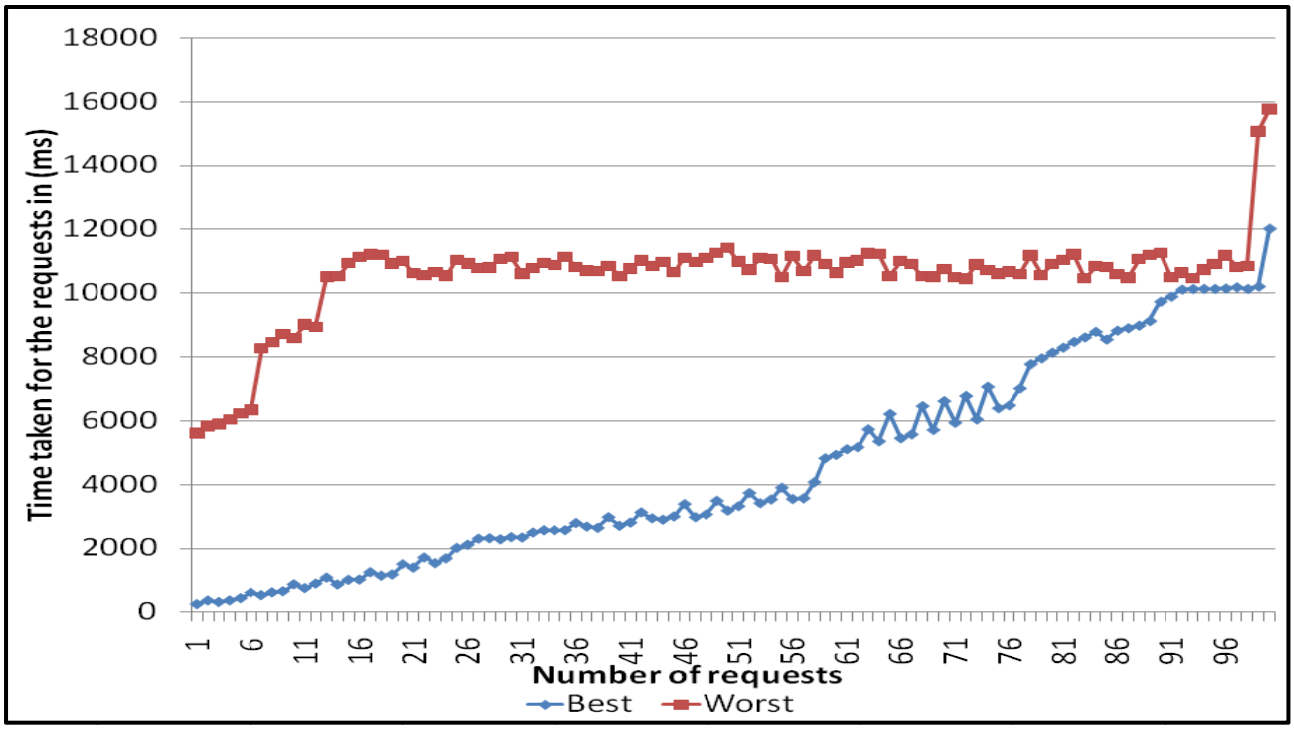


Figure 6-43. The performance of the Java viewing service for workloads (100)

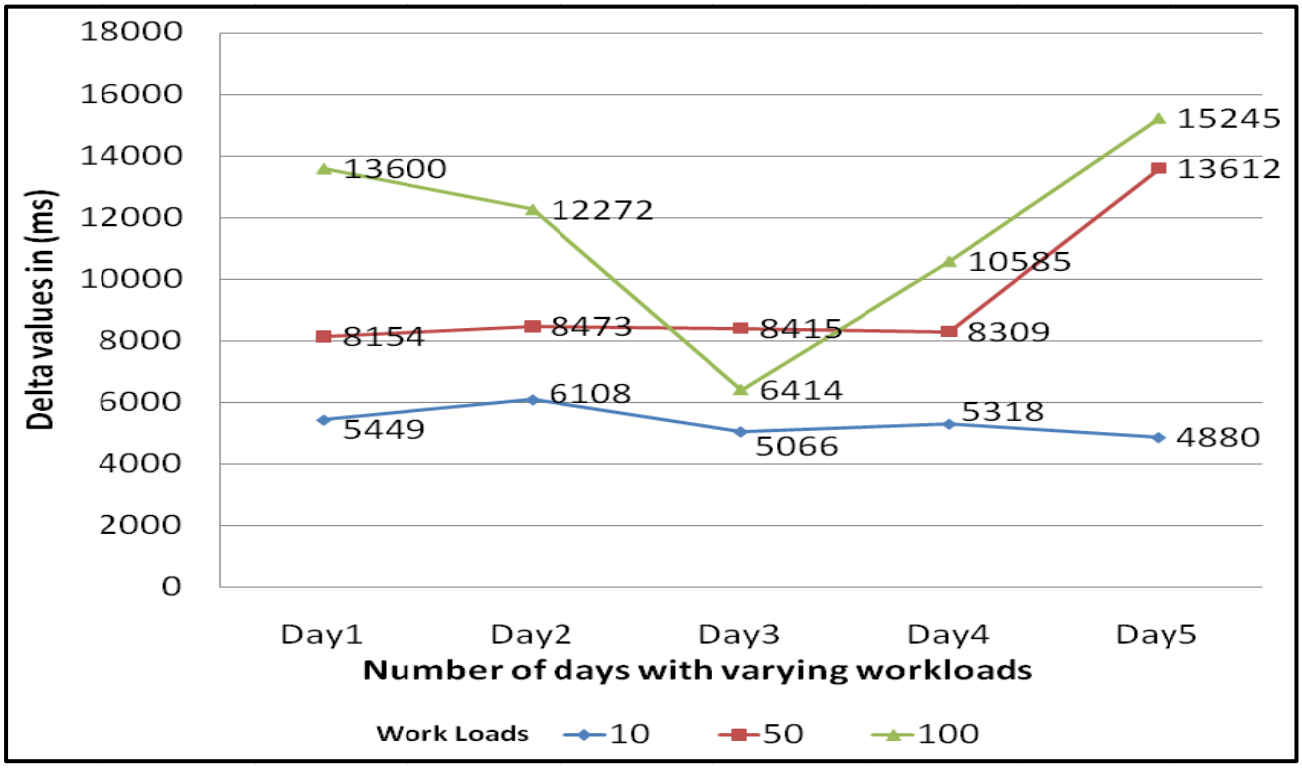


Figure 6-44. Difference of the times taken (ms) for viewing service in Java for workloads

From the Phase1 experiment results it is evident that Java services show the maximum time for 50 and 100 requests with workloads(10, 50, and 100) for the last request.

The performance graphs for each of the services show the best, and worst performances. These graphs indicate the reliability of the GAE services performance over a period of 5 days. As I mentioned in the evalution part the tests are conducted in the morning for 5 days during 9am to 12 noon Monday to Friday. The second part of the experiment shows the differences between the maximum and minimum times taken for each service evaluated in Python and Java for different days with workloads. It shows the variation of services performance with workloads (10, 50, and 100) on each day.

The Python services perform best by taking minimum time for processing the requests. The Java services are scalable for all the workloads but show increase in time in comparison with the Python services. But in total, the services in both the languages Python and Java are scalable to the client requests with a variation in time taken.

But there is an interesting question left unanswered, why do Java services take more time than Python?

It is observed that for all the Java graphs starting Figures 6-25 till Figure 6-44 they show a chaotic behavior under loads. Java is known as of the best and most widely used mature platforms for application development showing the worst times taken is surprising. But the GAE platform it is still very young and it still in its early development stages. Since its launch in 2008 with Python as a platform, there were many development changes incorporated. Most importantly, Java is introduced with the GAE framework after the Python in 2009. GAE uses its own compilers for the Java languages due to which the performance of the applications is affected. The length of the Java code also reduces the performance. The Chapter 5 states that

Java code is lengthier than Python. However when comparing Java with Python in the GAE framework, Java code is verbose. It takes more input and an output statement to process a job when compared to Python a simple scripting language with few lines of code. The DS interactions for Python are only a few lines compared with Java JDO class in Java used for DS. It is also observed that the Java development environment for GAE framework uses Google's compilers and interpreters that are affecting the performance of Java applications by filling up the memory with JDO objects. It creates many objects in memory, and thus filling up the space on the horizontal Google storage servers. Once the memory is filled, the garbage collector of the Java starts automatically to clear the unused space of the Java objects thus creating an overhead in time. Thus the time taken is longer in Java when compared to Python.

In conclusion, it is evident that GAE Java framework has to be improved to reduce the time taken for processing the requests.

Phase2: Workflows for the experiments in Java and Python

The Java and Python experiments aim at the performance of the services as a workflow. In these experiments a set of services are arranged as a workflow in such a way that users execute the requests as the path mentioned in the workflow. Each time specified number of requests are sent to the workflow for N days where N=5. The workflows are run every day in the morning for N days and graphs are recorded accordingly.

Test Bed

The test bed for the experiments using Apache JMeter is explained in this section for the workflows. The workflows are designed separately for the experiments in Java and Python.

The workflow is a combination of services in Python and Java. For the test bed we are discussing the test bed designed in Java. The test bed has a sequence of 5 important steps. The test bed is same as the test bed for individual services (described for Phase1 experiments) but with additional simple controllers for session information.

The test bed consists of the thread group which is used to define number of users, ramp-up time in seconds and the number of loops. The workflows are tested with workloads 10, 50, and 100. These workloads are tested on everyday for N=5 days. Usually a client logs into the experiment using “login” service which leads into the main service and the client may post a story using the “poststories”. Later the client may post a comment and change the account settings. These operations are defined in the workflow with the requests to different pages. Each page request is defined by the HTTP Request Defaults and is associated with a HTTP Requests shown in figure 6.2 and figure 6.3. The HTTP Requests Defaults is having the default values based on which HTTP Requests is processed.

The simple logic controllers are used in the workflow to organize the samplers and other logical controllers. The simple logic controllers can be added by right clicking on the thread group. In the simple logic controllers HTTP URL Re-writing Modifier is added where the session information is managed using the variable mentioned. The session id is cached if the option for the cache is checked and can be used for the other services. The results are recorded using the View Results in a table which is added by right click option on the thread group. The results panel is shown in Figure 6.4. The figure 6.15 shows the HTTP URL Re-writing Modifier.

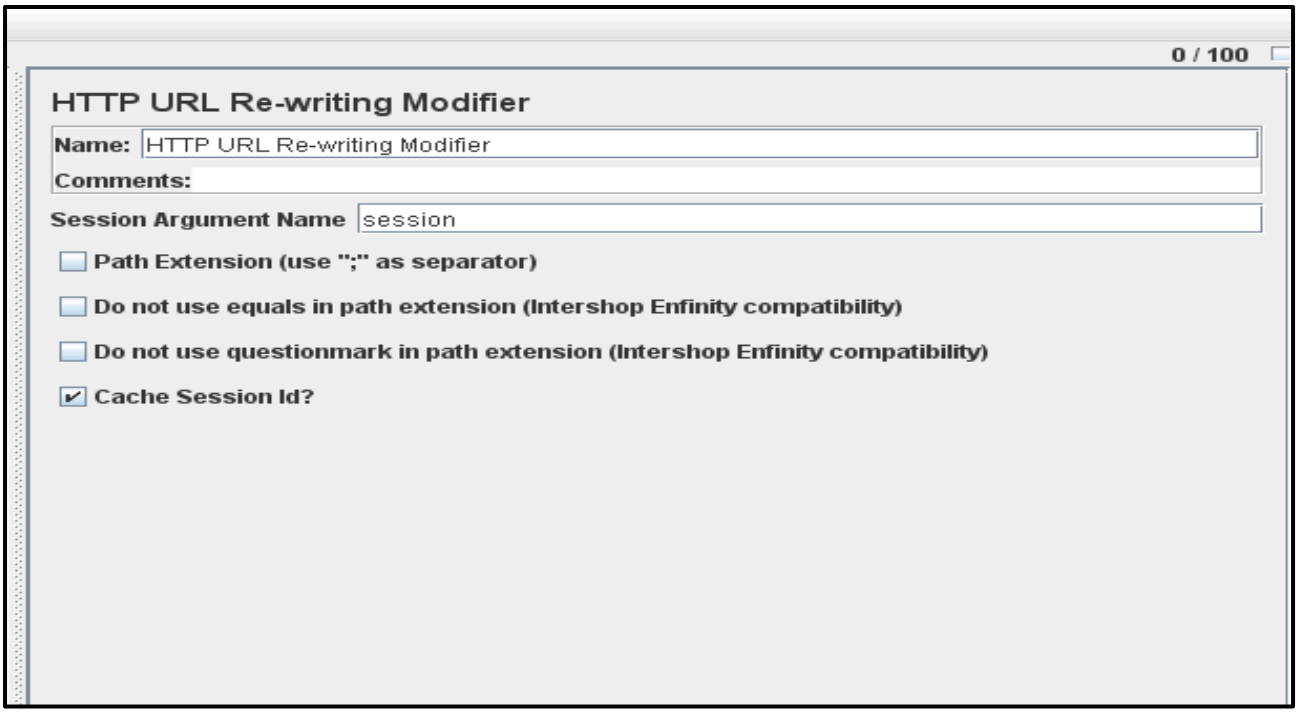


Figure 6-45. The HTTP URL Re-writing Modifier and its properties

Results of the Phase2 Experiments

This section discusses the results of the workflows using Python and Java that are evaluated using JMeter. The workflows are evaluated for workloads 10, 50, and 100. The workloads are recorded for N=5 days as to measure the performance of the GAE services during the days of a week. The workflow is defined as a sequence of service navigation through an application. For the Python experiment the workflow is in the order of the services mentioned in table 6.2. The first set of graphs shown in Figure 6-46 till Figure 6-48 show the best, and worst performances of the workflow in Python for the workloads (10, 50, and 100).

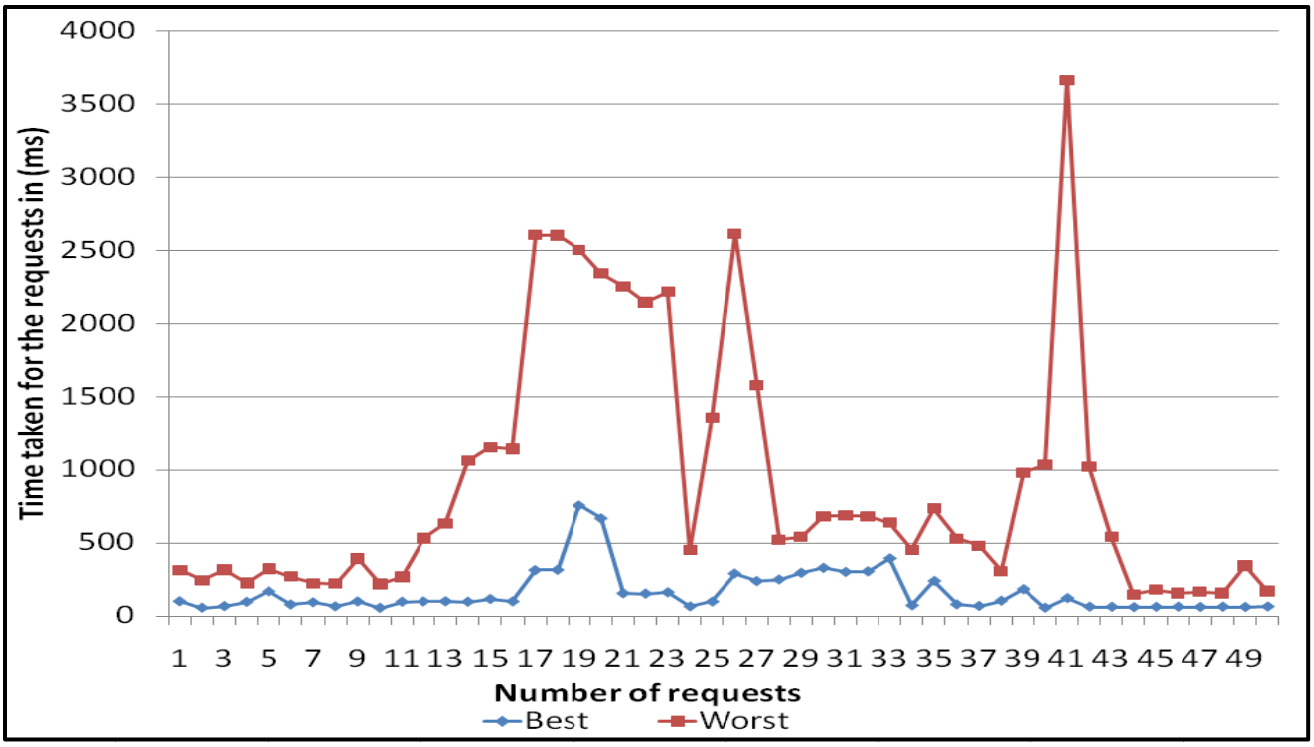


Figure 6-46. The performance of the Python workflow for workload (10)

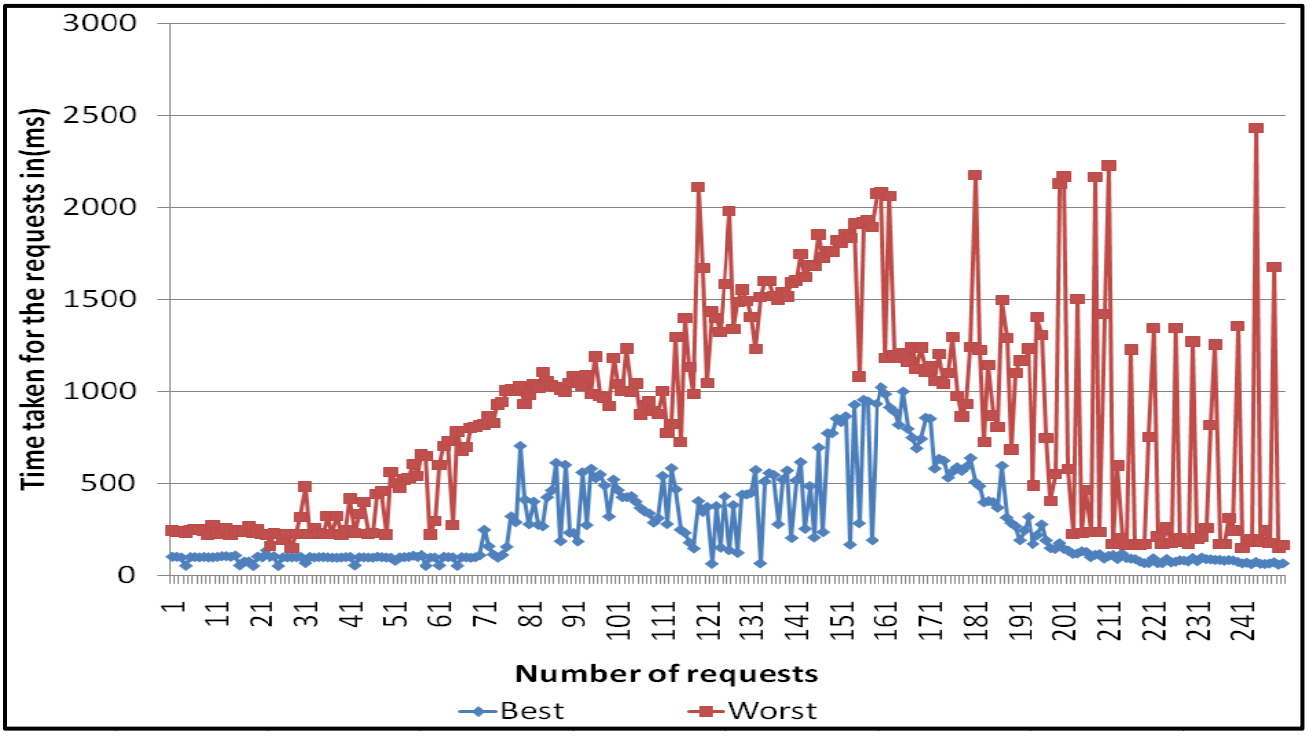


Figure 6-47. The performance of the Python workflow for workload (50)

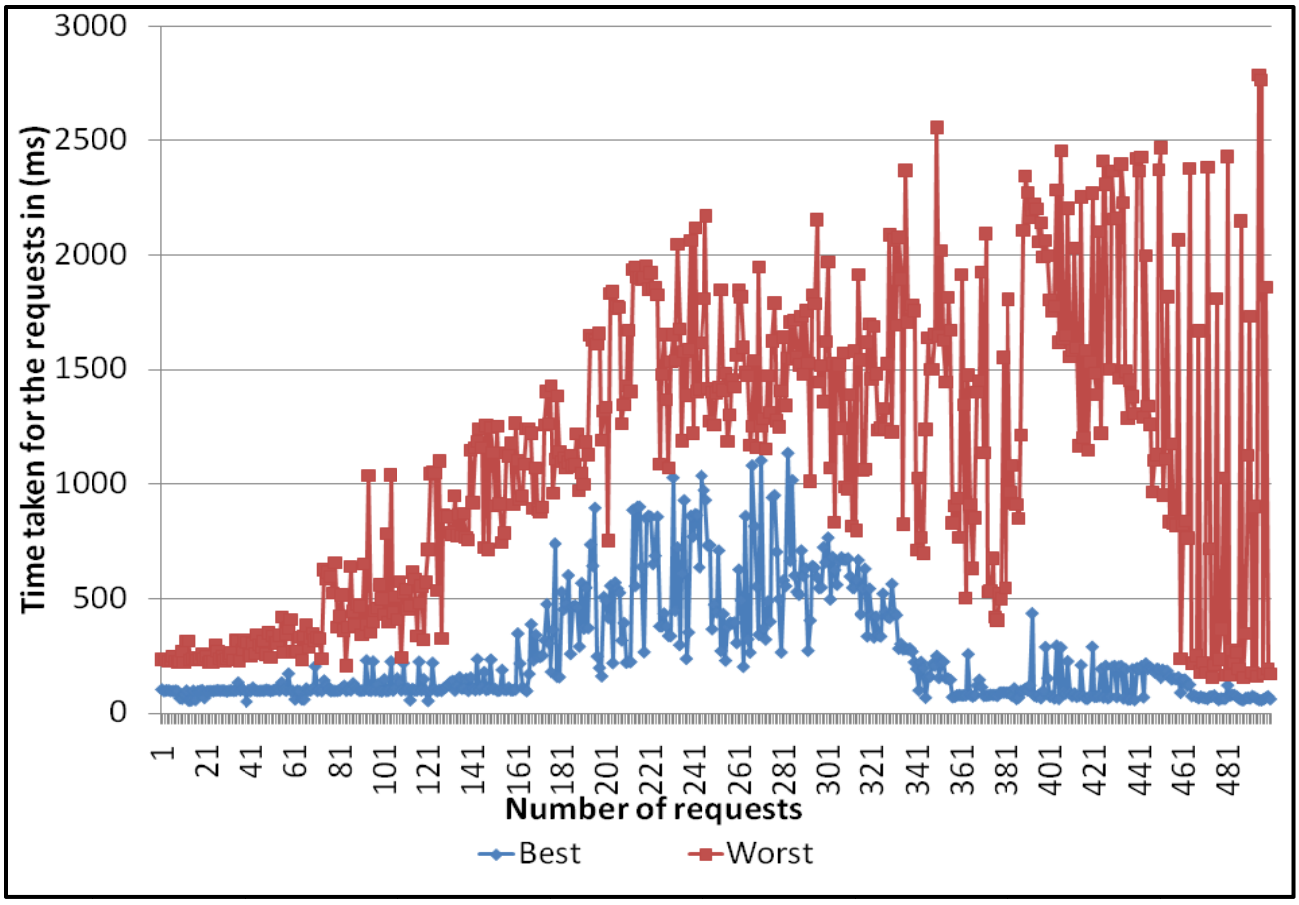


Figure 6-48. The performance of the Python workflow for workload (100)

For the workflows everyday maximum and minimum values are calculated. The workflows delta values calculated as the difference between maximum and minimum (discussed in results of the phase1). The services that are used for the workflows for the Python and Java experiments are mentioned in the tables 6.2.

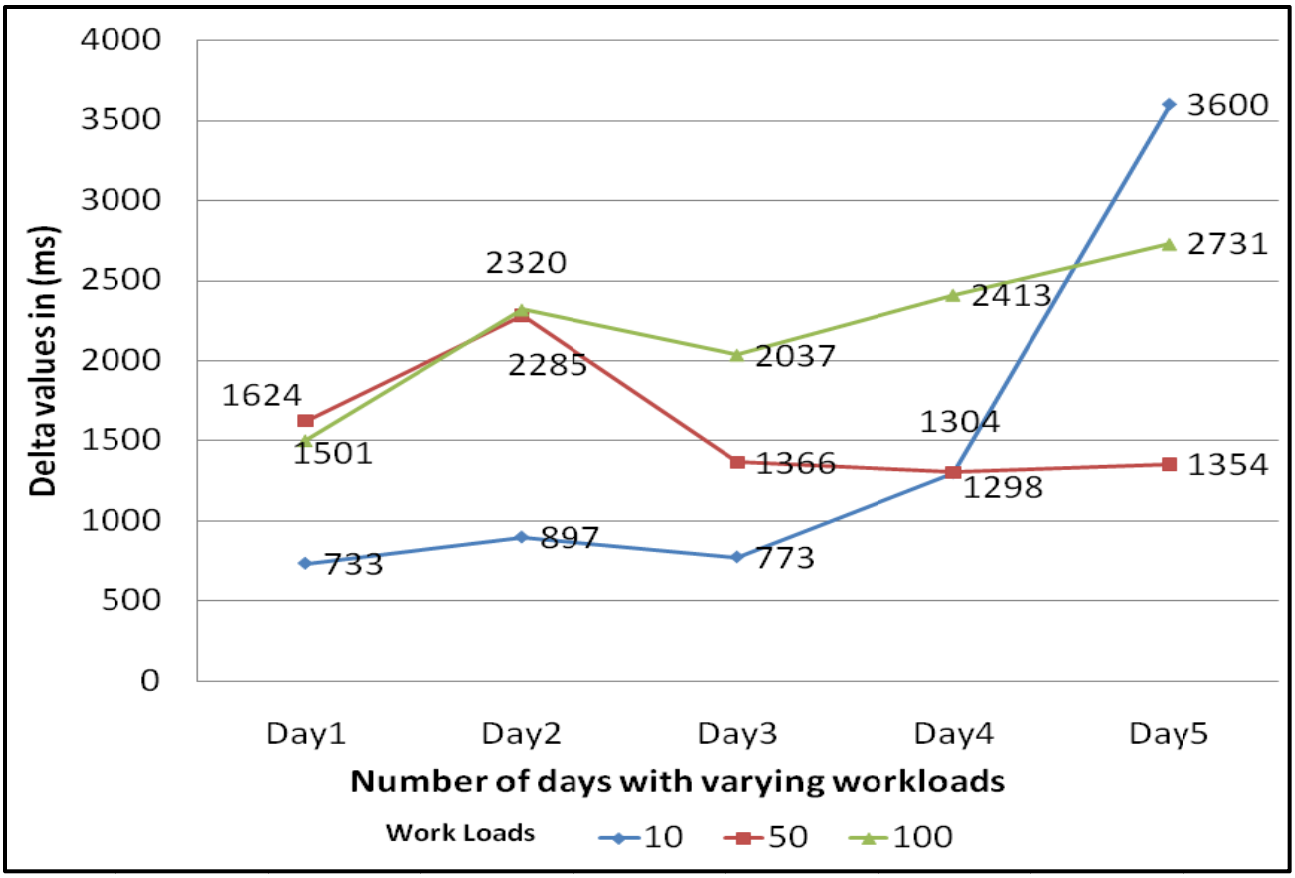


Figure 6-49. Difference of the times taken (ms) for the workflow in Python for workloads

Similarly the Java services are also evaluated as a workflow. The workflow for the Java services is in the same order as mentioned in the table 6.3. The graphs for this set of services are in the following order. First the performance of the workflow is calculated with the best and worst values. Then the delta values calculated for the workflows every day and the graph is recorded with the difference values recorded for the workloads. Thus the pattern indicated in Figure 6-49 shows average performances of the graphs for each day for workloads. The Figures 6-50 till Figure 6-52 show the performance for the workloads and the delta values for the services in Java.

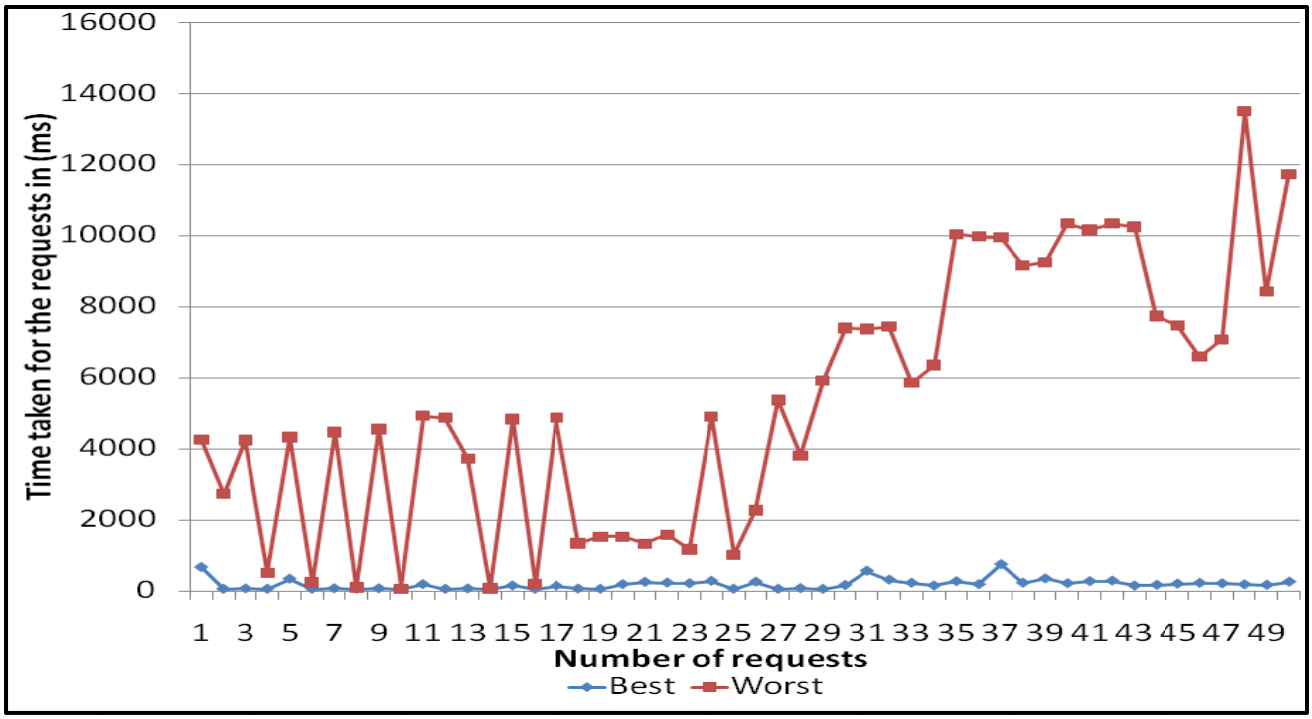


Figure 6-50. The performance of the Java workflow for the workload (10)

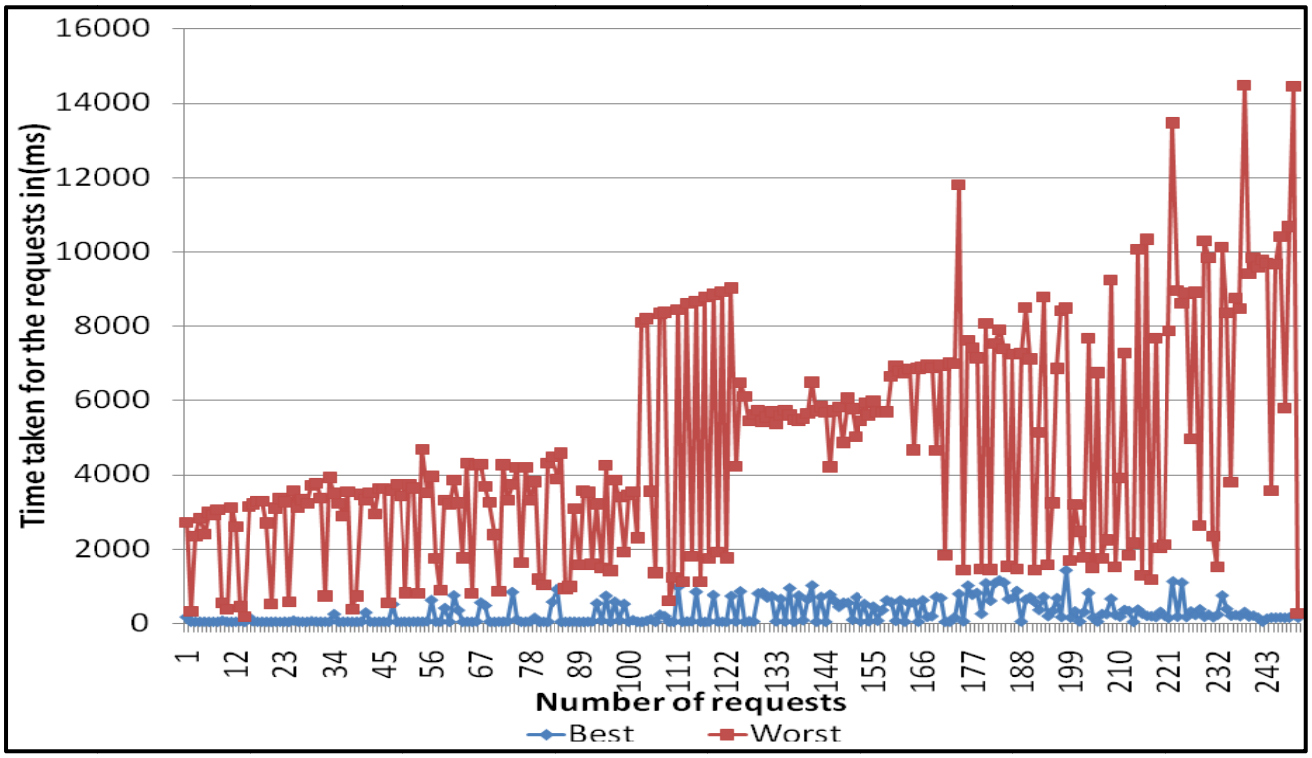


Figure 6-51. The performance of the Java workflow for workloads (50)

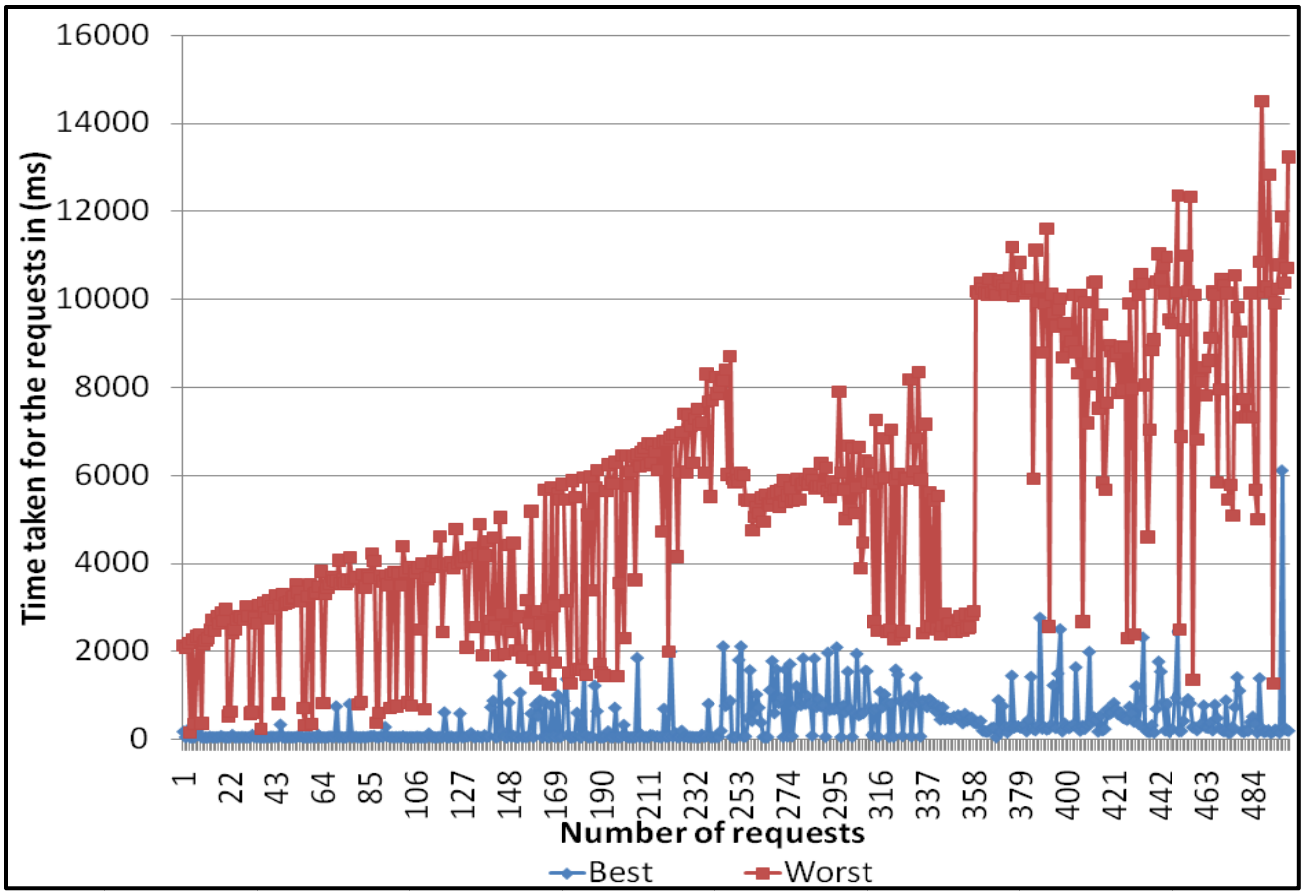


Figure 6-52. The performance of the Java workflow for the workload (100)

The workflows in Python and Java are designed with same number of services and similar GET and POST requests. But the Figures 6-46 till Figure 6-53 show a tremendous difference in time taken each day for different workloads. These results indicate that Java services are worse than Python services because of the JDO objects that take time to be cleared from the memory.

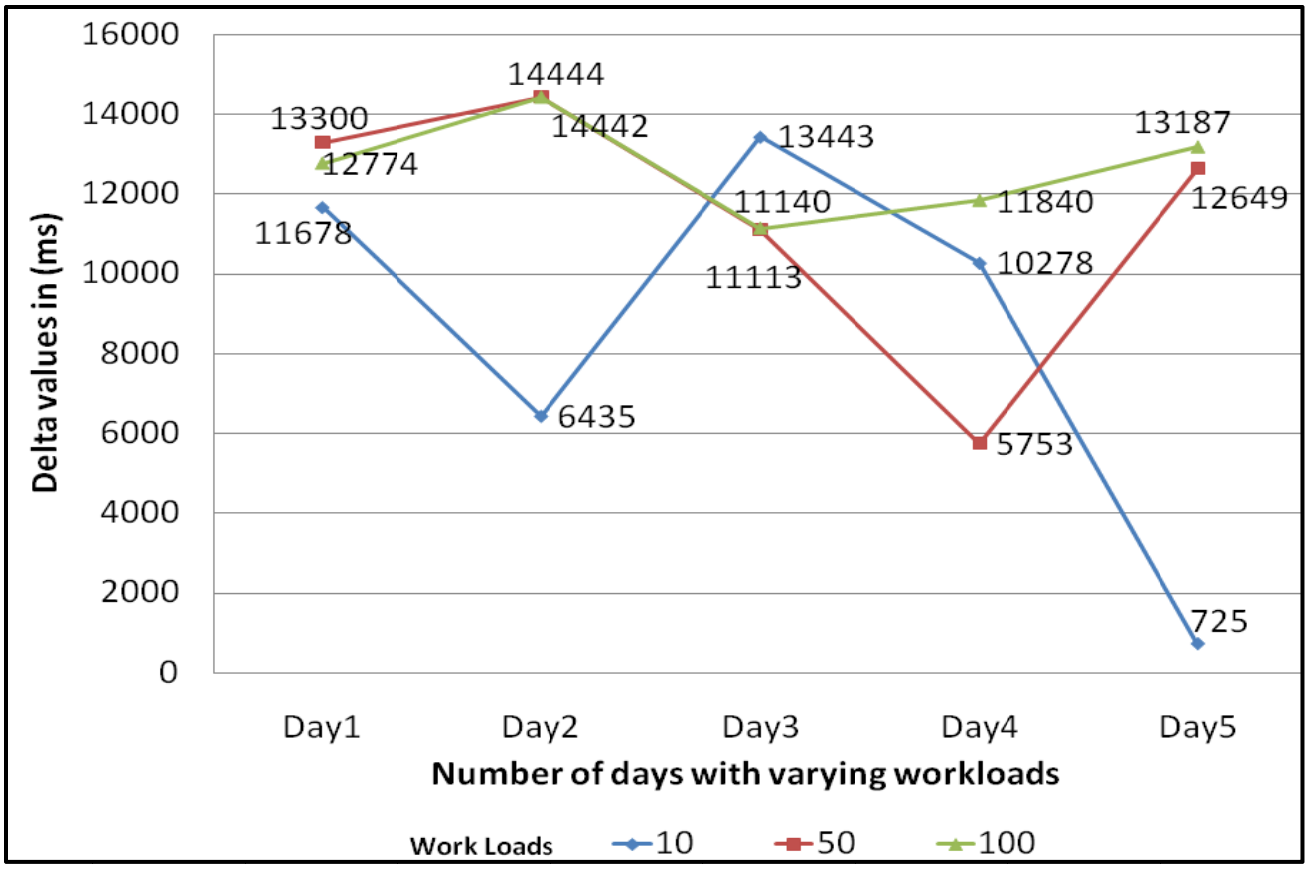


Figure 6-53. Difference of the times taken (ms) for the Java service for workloads

Summary

In summary, as mentioned in the goals for evaluation, the applications were tested with varying workloads for Python and Java applications. From the figures it is evident that Python based applications take less time than Java based applications and the attributes that define the performance of the web application like scalability and reliability are fulfilled for normal and bench mark users. Thus the performance experiments based on loads prove that the GAE is able to withstand the concurrent requests for normal and benchmark users and is scalable and reliable with the workloads.

Although the argument “Apache JMeter is not a best tool for testing the scalability of the applications for concurrent client requests”, it is one of the most popular one. The repeated experiments prove that the time taken for the services does not change rapidly. As a result, the Python and Java results indicate the performance is dependent on the length of the code in an application. As Python applications have a minimum code they take less time than the Java JDO object libraries that are built on top of the GAE. However, the extensive tests are to be conducted in an controlled environment without any traffic shaping.

The important conclusion is that, Java is an expensive programming language in regards to the time taken with the GAE requests. Thus it raises a question about the suitability of object oriented programming languages for the GAE. Thus there is need for slimmer programming languages that do not occupy all the memory in horizontal scalable servers and does not use object oriented programming. The emergence of a new trend to develop the programming languages that are thin and take less memory is suitable with cloud platforms like PaaS.

CHAPTER 7
CONCLUSIONS AND FUTURE WORK

Conclusions

Traditional computing patterns do not support and maintain the growing demand for rapid application development. In the past introduction of Internet has increased the need for the applications to interact over the web for business transactions. The WS has played a major role to enhancing the speed of the communicating parties. The business applications were highly developed using the RPC SOAP interactions which later had complications due to its interconnected behavior, complex code in applications. All these overheads lead to the development of a light weight protocol to establish the communication between the parties. It is called as the REST architectural style which develops the application based on the rules that govern the web. Due to all the advantages it can be easily embedded to design very complex applications with simple code and fewer interactions between the parties without affecting the behavior of the other parties.

Now the web is governed by the rich Internet applications that use REST based interactions. Based on these principles a model was proposed to develop applications easily using the REST principles and also to cut down the development costs incurred. These purposes are served by the recent developments in computation with which the services are available to people as models. The CC platforms cut down the costs of the applications and are categorized into different platforms based on the resources they provide [9]. Among all the platforms we are mostly concentrating on the platforms that allow development of the applications. Some of these platforms are the GAE, the Microsoft Azure. The GAE is popular since Google provides the service for free.

This research is based on using the GAE framework.

The contribution of this research includes the following

- Re-design a social networking application in the cloud

  The goal of this research is to re-implement an existing social networking application in the cloud. The MVC architecture enables thin client interaction with a clear separation of concerns. RESTful WS are suitable for developing thin resource oriented service that can develop web applications suitable for the cloud. Applications are designed by choosing the functionality of an existing social networking application that has scalability and maintenance issues. The experiments designed in Python and Java are based on the MVC architecture and multiple clients are used to present the design of the application (Flex clients with Python and Java, HTML, JSP, and IPOD client). In addition, Python code is approximately 660 which is limited compared to the Java code that takes 1500 including design, database interactions, and business logic.

- Design patterns for the cloud

  The first chapter introduces the problems with traditional computing and the need for scalable applications. It was evident from the literature that PaaS provides a platform where applications can be developed. Among the PaaS model, GAE provides a free and an attractive environment. GAE allows application development with 2 languages. Java and Python allows creating simple, fast, and attractive resources that creates scalable.

- Scalability

The CC enables easy applications that are easily developed with less complexity and are scalable. The social networking application that is re-implemented is evaluated for performance in Python and Java languages. Though the GAE Java application's scale better than traditional software's, it stills needs to improve its performance when compared to Python applications. Java is an object oriented programming language and is not suitable for the cloud platforms. The PaaS cloud applications need applications to scale better than the traditional applications for which there is a need for choosing the languages that are slim and does not scale horizontally over the Google's servers. However the GAE is still in its early phases of development and will require more time and upgrades for the platform to mature and develop scalable applications in Java language.

Thus, the GAE uses the RESTful architectural style to design services using the Python and Java languages that are scalable and provide and make them available with multiple clients with resource oriented approach based on MVC design principles.

Future work

- To re-implement the applications using the HTTP1.1 protocol.

  The HTTP1.0 protocol is used as the most successful protocol. In spite of its wide usage, it has numerous flaws. HTTP1.1 reuses the socket connections. It does not break the socket connection once the request and response is completed so that the next request could be processed on the

same instead of additional delay in establishing a new one. HTTP1.0 has a serious impact on Java's performance with network connection break up for every interaction. A system which uses same socket is significantly better. However even if the network connection improves, the problem with Java creating too many objects remains the same.

- To test the applications using testing software and do repeated testing on different operating systems. Also to test the applications within a network where there is no traffic shaping.

  To test the applications on the multiple operating systems like Linux, Mac OS, to check the performance of the applications as the web applications performance changes with operating systems. Further, to move the applications into an uncontrolled environment independent of the institutional or organizational environment to check the best and worst case performances on each of them.

- The GAE applications can be cached to see the performance of the applications. Caching is an important feature of the GAE that provides for high performance memory objects primarily used for faster access to the results of cached DS queries.

  The existing applications are designed without caching to test the worst case performance of the GAE services. In future, the best performance of the applications has to be evaluated in the GAE with the help of caching features.

LIST OF REFERENCES

[1] Papazoglou, M.P., Traverso, P., Dustdat, S., and Leymann, F. Service-Oriented Computing Research Roadmap, Service Oriented Computing (SOC), number 0432 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum fuer Informatik, Schloss Dagstuhl, Germany 2006.

[2] Liu, Y., Wang, Q., Zhuang, M., and Zhu, Y., Reengineering Legacy Systems with RESTful Web Services, Published in Computer Software and Applications, 2008. COMPSAC '08, 32nd Annual IEEE International.

[3] Menasce, A.D., QoS Issues in Web Services, IEEE Internet Computing 2002, vol. 6, Issue 6, Nov./Dec 2002.

[4] Litoiu, M., Migrating to Web services: a performance engineering approach, Published in Journal of Software Maintenance and Evolution: Research and Practice, vol. 16, Issue 1-2, Jan./Apr. 2004.

[5] The hidden battle between web services: REST versus SOAP [Online], 1st July 2010, Available: http://hinchcliffe.org/archive/2005/02/12/171.aspx.

[6] Palankar, M., Ripeanu, M., Garfinkel, S., Amazon S3 for Science Grids: a Viable Solution?, InProceedings of the 2008 international Workshop on Data-Aware Distributed Computing, Boston, MA, USA, June 24 - 24, 2008, DADC '08. ACM, New York, NY, 55-64.

[7] Pautasso, C., Zimmermann, O., Leymann, F., RESTful Web Services vs."Big" Web Services: Making the Right Architectural Decision, Published in WWW 2008, Web Engineering-Web Service Deployment, April 21-25, 2008, Beijing, China.

[8] Calcote, J., Open Sourcery, Technology, Open source and Identity [Online], 13th July 2010, Available: http://jcalcote.wordpress.com/2008/10/16/put-or-post-the-rest-of-the-story/.

[9] Cloud Computing [Online], 12th March 2009, Available: http://en.wikipedia.org/wiki/Cloud_computing.

[10] Platform as a Service [Online], 23rd October 2009, Available: http://www.keeneview.com/2009/03/what-is-platform-as-service-paas.html.

[11] Amazon Elastic Compute Cloud [Online], 30th March 2009, Available: http://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud.

[12] Google App Engine [Online], 30th March 2009, Available: http://en.wikipedia.org/wiki/Google_App_Engine.

[13] Azure [Online], 30th March 2009, Available: http://en.wikipedia.org/wiki/Microsoft_Azure.

[14] Vinoski, S., Where is Middleware, IEEE Internet Computing 2002, 1089-7801/02, March- April, 2002.

[15] Vaughan-Nichols, S.J., Web Services: Beyond the Hype, Published in IEEE Computer Society, Computer, vol. 35, no. 2, pp. 18-21, Feb. 2002, doi:10.1109/2.982908.

[16] Why we need Web Services Networks [Online], Web Services & XML, 2009(21/09), Available: http://www.ebizq.net/topics/web_services/features/1542.html.

[17] Papazoglou, M. P., Web Services: Principles and Technology, Pearson Education Limited, London, 2008, pp.120.

[18] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S., Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI, IEEE Internet Computing, vol. 6, no.2, pp. 86-93, Mar./Apr. 2002.

[19] Muehlen, M.Z., Nickerson, J.V., Swenson, K.D., Developing Web Services Choreography Standards-The Case of REST vs. SOAP, Decision Support Systems 37(2004), Elsevier, North Holland (to appear).

[20] Fremantle, P., Weerawarana, S., Khalaf, R., ENTERPRISE SERVICES, Published in the COMMUNICATIONS OF THE ACM, vol.45,no.10, pp. 77-81, Oct. 2002.

[21] Shi, X., Sharing Service Semantics using SOAP-Based and REST Web Services, Published in IT Professional, vol. 8, no.2, pp.18-24, Mar./Apr. 2006.

[22] W3schools.com [Online], 18th February 2010, Available: http://www.w3schools.com/soap/soap_example.asp.

[23] Wilde, E., Open Location-Oriented Services for the Web [Online], Published in Springer 2004, UCB ISchool Report 2008-026, August 2008, http://dret.net/netdret/publications#wil08o.

[24] Brose, G., Securing Web Services with SOAP Security Proxies, Published in International Conference Web Services (ICWS'03), 2003.

[25] Gokhale, A., Kumar, B., Sahuguet, A., Reinventing the Wheel? CORBA vs. Web Services, Published in the Eleventh International World Wide Web (WWW2002), 2002.

[26] Fielding, R. T., Architectural Styles and the Design of Network-based Software Architectures (Ph.D. Thesis), University of California, Irvin, CA, Information and Computer Science, 2000.

[27] Fielding, R. T., Taylor, R. N., Principled Design of the Modern Web Architecture, Published in 2000, ACM, In the Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000), Irvine.

[28] Xu, X., Zhu, L., Liu, Y., Staples, M., Resource-Oriented Architecture for Business Processes, Published in Software Engineering Conference, 2008, APSEC'08, 15th Asia-Pacific.

[29] Costello, R. L., Building Web Services the REST Way [Online], 2007 Available: http://www.xfront.com/REST-Web-Services.html.

[30] Curbera, F., Duftler, M. J., Khalaf, R., Composing RESTful Services and Collaborative Workflows: A Lightweight Approach, Published in the IEEE Computer Society, IEEE Internet Computing 2008, 1089-7801/08, September-October 2008, pp 24-31.

[31] Heaton, R., Sanity Stack, Available: http://sanitystack.blogspot.com/ 2010(01/02)

[32] Maibücher, S., REST Web Services [Online], 14th October 2009, Available: http://www.predic8.com/rest-webservices.htm.

[33] Jucyte, K., Kevelaitis, K., Park, S. W., Web service implementation with SOAP and REST, RUC Datalogi, Module2 , Fall 2006.

[34] Buyya, R., Yeo, C. S., Venugopal, S., Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities, In the Proceedings of the

10th IEEE International Conference on High Performance Computing and Communications (HPCC-08, IEEE CS Press, Alamitos, CA, USA), Sept. 25-27, 2008, Dalian, China.

[35] Hayes, B., Cloud Computing, Communications of the ACM, 2008, Volume 51, Issue 7 (July 2008).

[36] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R.,Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., Zaharia, M., Above the Clouds: A Berkeley View ofCloud Computing,Technical Report UCB/EECS-2009-28, UCB, Feb. 2009.

[37] Barnatt, C., Explaining Cloud Computing [Online], 10th May 2009, Available: http://www.explainingcomputers.com./cloud.html.

[38] Software as a Service [Online], 23rd October 2009, Available: http://searchcloudcomputing.techtarget.com/sDefinition/0,,sid201_gci1170781,00.html

[39] Software as a Service [Online], 23rd October 2009, Available: http://it.toolbox.com/wiki/index.php/SaaS.

[40] Platform as a Service [Online], 30th March 2009, Available: http://en.wikipedia.org/wiki/Platform_as_a_service.

[41] Infrastructure as a Service [Online], 23rd October 2009, Available: http://searchcloudcomputing.techtarget.com/sDefinition/0,,sid201_gci1358983,00.html.

[42] As a Service: The many faces of a cloud [Online], 23rd October 2009, Available: http://devcentral.f5.com/weblogs/macvittie/archive/2008/11/20/as-a-service-the-many-faces-of-the-cloud.aspx.

[43] Cheow, C., What is Cloud Computing and how it can help your Business [Online], 14th July 2010, Available: http://ezinearticles.com/?What-is-Cloud-Computing-and-How-it-Can-Help-Your-Business?&id=4533382.

[44] Top Cloud Service Providers; Amazon, Google, IBM [Online], 24th October 2009, Available: http://www.mrwebmarketing.com/web-news/top-cloud-service-providers-amazon-google-and-ibm.

[45] Weiss, A., COMPUTING IN THE CLOUDS, ACM Networker, Vol. 11, Issue 4, pp 16-25, December 2007.

[46] Gift, N., Orr, M., Google App Engine in Action,http://www.manning.com/gift/, Manning, July 2008.

[47] Sahib, Z. H., WISETales: Designing a New Niche Online Community for Women in Science and Engineering to Share Personal Stories, University of Saskatchewan, 2009.

[48] Google App Engine [Online], 26th October 2009, Available: http://code.google.com/appengine/docs/whatisgoogleappengine.html.

[49] Google App Engine limitations and how to get around them [Online], 26th October 2009, Available: http://www.digitalistic.com/2008/09/16/google-app-engine-limitations-and-how-to-get-around-them/.

[50] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D.A., Burrows, M., Chandra, T., Fikes, A., Gruber, R. E., "Bigtable: A Distributed Storage System for Structured Data", (To appear in OSDI 2006).

[51] Jazayeri, M., Some Trends in Web Application Development, Future of Software Engineering (FOSE'07),IEEE Computer Society, Washington, DC, USA (2007).

[52] Kalil, T., Harnessing the Mobile Revolution, Published in The New Policy Institute, 8th October, 2008.

[53] GigaOMPro, Quarterly Wrap-up: Second Quarter 2009 in Review: Mobile [Online], 20th July 2009, Available: http://pro.gigaom.com/2009/07/second-quarter-2009-in-review-2/.

[54] Serhani, M. A., Benharref, A., Dssouli, R., Mizouni, R., Toward an Efficient Framework for Designing, Developing, and Using Secure Mobile Applications, In the Proceedings of World Academy of Science, Engineering and Technology, Volume 40 April 2009, ISSN 2070-3740.

[55] Adobe Labs, Flex Builder 3 features [Online], 2nd November 2009, Available: http://www.adobe.com/products/flex/features/flex_builder/.

[56] Solanki, A., JMeter: A tool for performance testing your webapp [Online], July 3rd 2010, Available: http://www.slideshare.net/amitkssolanki/jmeter-performance-testing-your-webapp.

[57] Oehlman, D., Google App Engine—Java vs Python Performance Comparison [Online], July 02nd 2010, Available: http://distractable.net/coding/google-appengine-java-vs-python-performance-comparison/.

[58] The Apache Jakarta Project http://jakarta.apache.org [Online], 20th June 2010, Available: http://jakarta.apache.org/jmeter/usermanual/build-web-test-plan.html.

[59] The Amazon CloudFront [Online], 17th July 2010, Available: http://aws.amazon.com/cloudfront/.

[60] Miller, R., Where Amazon's Data Centers are Located [Online], 17th July 2010, Available: http://www.datacenterknowledge.com/archives/2008/11/18/where-amazons-data-centers-are-located/.

[61] Miller, R., Google Data Center FAQ [Online], 17th July 2010, Available: http://www.datacenterknowledge.com/archives/2008/03/27/google-data-center-faq/

[62] Royal Pingdom Blog, Map of all Google Data Center Locations [Online], 17th July 2010, Available: http://royal.pingdom.com/2008/04/11/map-of-all-google-data-center-locations/.

[63] Ciurana, E., Developing with Google App Engine, pp: 73-74 (book) Publisher: Apress – A, 1 edition Feb 2, 2009.

[64] Wilson, J., Understanding HBase and Big Table [Online], 17th July 2010, Available: http://jimbojw.com/wiki/index.php?title=Understanding_Hbase_and_BigTable.

[65] Batty, P., Google App Engine and the BigTable-VERY interesting! [Online], 17th July 2010, Available:http://geothought.blogspot.com/2009/04/google-app-engine-and-bigtable-very.html.

[66] Hitchcock, A., Google's BigTable [Online], 17th July 2010, Available: http://andrewhitchcock.org/?post=214.

[67] Model-View-Controller [Online], 17th July 2010, Available: http://en.wikipedia.org/wiki/Model–view–controller.

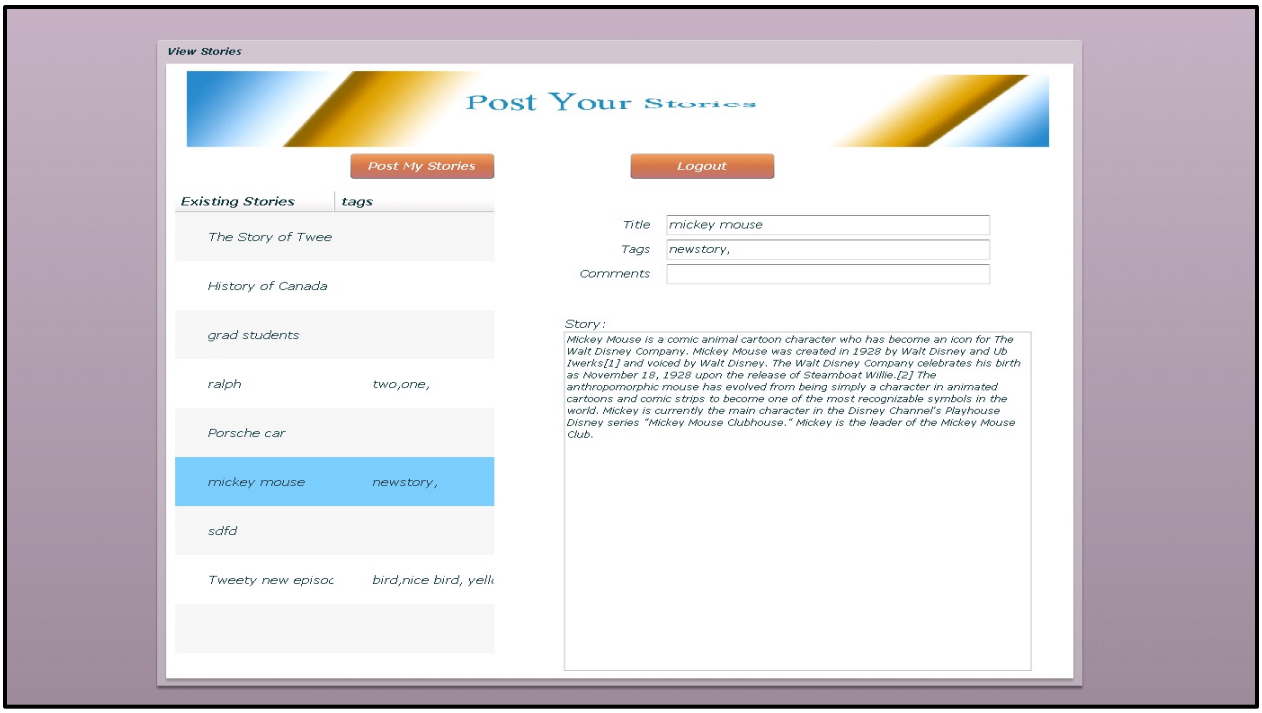
[68] Anderson, D. J., Using MVC Pattern in Web Interactions [Online], 17th July 2010, Available: http://www.uidesign.net/Articles/Papers/UsingMVCPatterninWebInter.html, Published in October 1999.

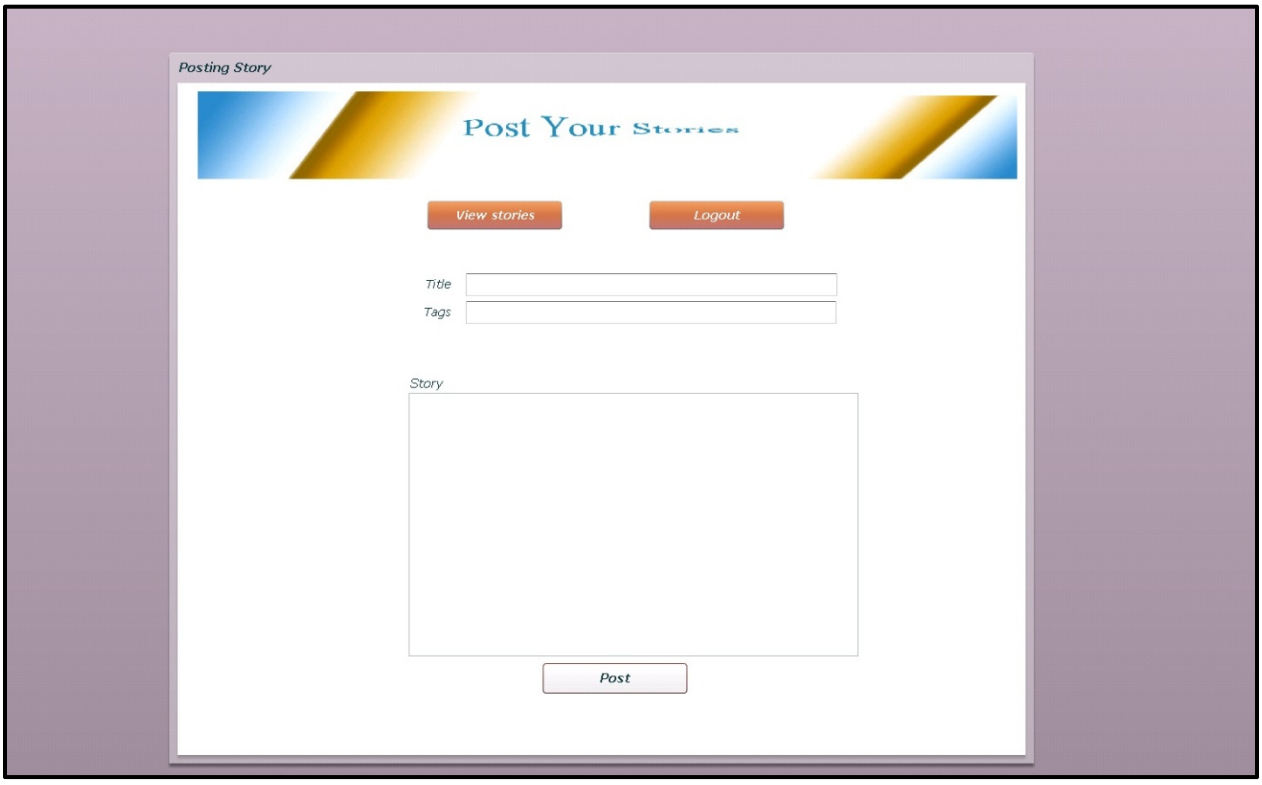
[69] Sarrel, M. D., NoSQL Databases: Providing Extreme Scale and Flexibility, Gigaompro Infrastructure [Online], 16thJuly 2010, Available: http://pro.gigaom.com/2010/07/report-nosql-databases-providing-extreme-scale-and-flexibility/.

[70] Bhatia, A., Big Table [Online], 17th July 2010, http://it.toolbox.com/wiki/index.php/BigTable.
[71] Dogan, B., BigTable Concept: Why do the World's Smartest People Ignore Relational DB's? [Online], 17th July 2010, http://blog.burcudogan.com/9/

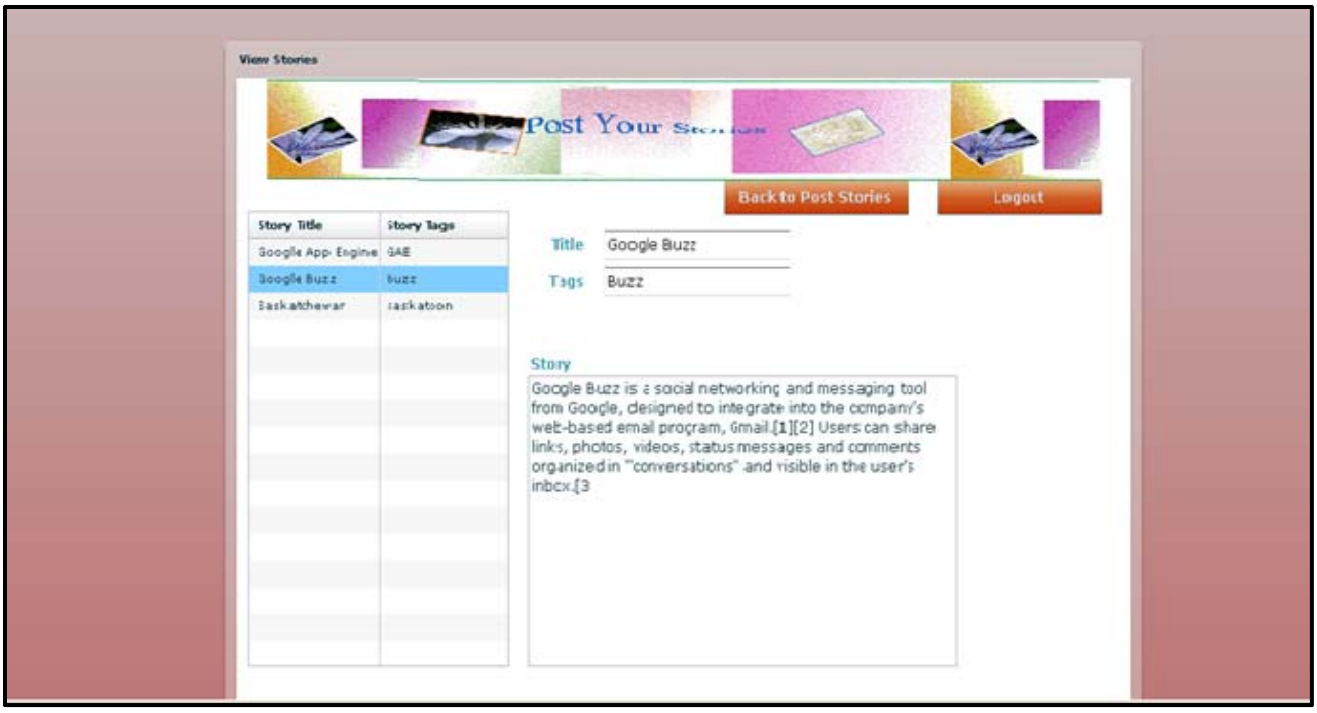
APPENDICES

Appendix A – Python Flex client screen





Appendix B – Java Flex client screen


View Stories
Post Your Stories
Back to Post Stories
Logout
Story Title
Story Tags
Google App Engine
GAE
Google Buzz
buzz
Saskatchewan
saskatoon
Title
Google Buzz
Tags
Buzz
Story
Google Buzz is a social networking and messaging tool from Google, designed to integrate into the company's web-based email program, Gmail.[1][2] Users can share links, photos, videos, status messages and comments organized in "conversations" and visible in the user's inbox.[3


Post Stories
Post Your Stories
View Stories
Logout
Title
Tags
Story
Post Story