Journal of Mathematics and System Science 5 (2015) 156-164
doi: 10.17265/2159-5291/2015.04.004

The Benefits of Using Google Cloud Computing for Developing Distributed Applications

Isak Shabani, Amir Kovaçi and Agni Dika

Dep. of Computer Engineering, Faculty of Electrical and Computer Eng., University of Prishtina, 1000, Prishtine, Kosovo

Received: January 24, 2015 / Accepted: February 25, 2015 / Published: April 25, 2015.

Abstract: IT as a dynamic filed changes very rapidly; efficient management of such systems for the most of the companies requires handling tremendous complex situations in terms of hardware and software setup. Hardware and software itself changes quickly with the time and keeping them updated is a difficult problem for the most of the companies; the problem is more emphasized for the companies having large infrastructure of IT facilities such as data centers which are expensive to be maintained. Many applications run on the company premises which require well prepared staff for successfully maintaining them. With the inception of Cloud Computing many companies have transferred their applications and data into cloud computing based platforms in order to have reduced maintaining cost, easier maintenance in terms of hardware and software, reliable and securely accessible services. The benefits of building distributed applications using Google infrastructure are conferred in this paper.

Key words: Datastore, BigTable, Distributed application, Cloud Computing

1. Introduction

Distributed Systems are those in which the communication is performed through the message exchange; the systems can be remotely located but for them to efficiently communicate they should meet the conditions such as concurrent communication, action synchronization between the nodes in the system, independent handling of the failures etc [1]. Google File System (GFS) is at the core of data storage and processing through Google App engine (GAE). GFS files are composed of chunks similar to clusters in traditional file systems; further down GFS contains many nodes of which each instance has a Master Node and multiple Chunk Servers. The Datastore operates on cloud and is spread across multiple servers. Today distributed systems are installed on different hardware and software platforms, a key requirement such systems should meet is the ability to scale and fault tolerant data management [2]. Google through its advanced technical features manages to retrieve data and to represent them in user friendly format. Google has moved to Cloud computing as which represent a new paradigm toward data access and storage. Cloud computing offers to the user services to access: hardware, software and data resources thus, an integrated computing platform as a service [3]. Google App Engine is a typical example of platform as service; it enables the developers to use its platform to build distributed application, using programming languages such as Java and Python to build powerful, reliable and secure application as Google itself.

Rapid growth of computer systems together with the LAN and WAN networks has increase the level of data exchange which altogether have enabled connection and information exchange between different type of machines (PC, server, mobile devices). This advancement of computer networks known as distributed systems.

Corresponding author: Isak Shabani, Professor Assistant / Ph.D., Research field: Distributed systems, Web services, Data communication and synchronization. E-mail: isak.shabani@uni-pr.edu.

2. State of the Art

Distributed systems are defined as those systems in which hardware and software components communicate through message exchange. Google today plays a central in the branch of Cloud Computing which is defined as a technique of providing the services through the Internet with the capabilities to accommodate user requirements [4]. Those systems can physically be located in different locations but for those to communicate efficiently they should meet conditions such as concurrent communication, synchronization of actions with the other computer nodes, independent handle of failures within the system so that the other parts to be unaffected from the failure.

The main services that Google offers are Gmail, Google Docs, Google Talk, and Calendar which aim to replace the traditional software applications. Through its API service platform, possibilities are offered to the developer's community and the companies to develop and host their web based applications. With the introduction of Google App Engine, Google [5] has managed to go beyond the software services and now it offers the entire system infrastructure as services where different organizations run their applications through the Google platform.

Google File System (GFS) is at the core of data processing and storage of the Google as search engine. GFS files are separated in chunks similar to clusters or sectors in the traditional file system. BigTable enable data manipulation on the applications with massive data. Through specific search strategies and data operation features Google manages handling of huge information, processing of many queries and producing great results for the clients. Critical requirements that Google is able to fulfill are those related to reliability and continuous data provide [3]. Chubby is the service which synchronizes the activities of the clients when accessing the distributed resources.

3. Google Technological Infrastructure

From the distributed perspective, Google is built from a number of distributed services which provide the basic functionality; in general these features can be grouped as following [1]:

- Communication layer model including the service for remote procedure call and indirect communication through request serialization of remote calls.
- Coordination services and data provide access by means of: GFS, Chubby and BigTable.
- Services for distributed processing performing parallel operations in the physical layer infrastructure.

Google today plays a significant role in the technology of cloud computing which is defined as a set of applications based on the internet, with the capability of storing and processing most of the user requirements.

With the inception of Google App Engine, Google has managed to go beyond software services and it provides the infrastructure for distributed services as cloud services where the businesses and organizations can run their web applications through Google framework [2].

Similarly with the file systems found on operating systems which are used for general purpose operations on files and directories on different applications, GFS is also a distributed file system with a variety of abstractions and provides more advanced capabilities. The main aim is to process the growing requirements of Google as a search engine and other request which come from other web applications. There are a set of requirements which GFS must fulfill:

- The first requirement is GFS to be executed in a reliable way in the hardware and software architecture.
- GFS is optimized for the patterns of usage within Google, both in terms of the types of files stored and the patterns to access the files. The number of files stored in GFS is not huge in comparison with other systems, but the files tend to be massive. The

patterns of access are also atypical of file systems in general.

• GFS must meet all the requirements for the Google infrastructure as a whole; that is, it must scale (particularly in terms of volume of data and number of clients), it must be reliable in spite of the assumption about failures noted above, it must perform well and it must be open in that it should support the development of new web applications.

BigTable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers [6]. The App Engine Datastore is based on Google's Bitable, which is Google's storage system used for Gmail, Google Maps, YouTube, and many other services. This system is designed to scale across thousands of machines and perform well under critical circumstances [7]. Applications such as Google Earth, Google Finance, Orkut are typical cases of applications that use BigTable. These applications have very strict requirements to be fulfilled where through asynchronous communications is required that processes of millions of operations to be performed at high speed and availability. BigTables store their data in multidimensional arrays sorted in rows, columns and the timestamp. The rows on the table are represented as strings with atomic read and write operations taking place. The columns which are accessed more frequently are arranged to be as family of columns. With the help of timestamp mechanism, the arrangement is reached so that multiple versions of the content can be saved in the same cell and the latest version to be accessible. The rows are sorted alphabetically and in the groups with similarities are stored in the same engine for easy access. Different from the database systems, Datastore uses distributed architecture with the aim of properly managing big dataset and it differs a lot from the traditional databases how it describes the relationships between data objects.

4. Google App Engine

Google App Engine is a web application hosting service [4], it is a runtime platform based in Python and it offers capacities to host applications, to store the data and networking capabilities. The primary role of the engine is to serve the applications which are simultaneously accessed by many users, apart from the traditional web content which is served; Google App Engine serves the services such as mobile applications, social networking web sites and multiplayer games. Applications built with Google App Engine have the same power as Google applications which we use in our daily life. The application engine can serve traditional website content too, such as documents and images, but the environment is especially designed for real-time dynamic applications. Applications developed with Google App Engine run on distributed resources, the programmer only is in charge of developing the solutions. The runtime environment operates in abstracted manner separated from the underlying layer of the operating system in order to avoid platform dependencies. The Google infrastructure provides automatic, on-demand traffic shaping and load balancing for the developed application by distributing it across multiple servers. Each application also runs in its own secure sandbox, independent of other applications and potential resource conflicts [5]. Using Google App Engine as platform to build and deploy the applications has some advantages over traditional techniques of developing software solutions, such as offering of free resources to a certain level; easily build and maintenance of the applications. Though the App Engine is a technology based in cloud computing it does not need any virtual machine for the applications to run and scale.

Google App Engine supports developing applications in several programming languages. With the App Engine's Java runtime environment, applications can be built by using standard Java technologies including the JVM, Java servlets and the

Java programming language. Google App Engine also features a dedicated Python runtime environment, which includes a fast Python interpreter and the Python standard library. The Java and Python runtime environments are built to ensure that the applications run quickly, securely, and without interference with other apps on the system [6].

5. Google Datastore

Datastore is emerging as a promising way of storing and organizing data, it supports many similar features that the relational databases use to operate on data. The most used approach for data storage of web application is that based on Relational Databases model. Other types of organizing are those based on the hierarchical organization such as file systems, XML based databases and object databases. The database system supported by Google App Engine shares many features found on the object databases without supporting join operations on queries. Datastore uses tables, rows, columns and queries similar to relational databases for storing and performing standard create, read, update and delete operations on data. The queries performed on the Datastore return one or more entities of a given type and they can also return only the keys of the entities. Queries also can filter the data on the different conditions based on the values of properties of the entities; data ordering also is possible to be made. Although it offers execution of queries which syntactically look similar to SQL implemented on the relational database where the queries are planned and executed in the real time against the tables which are stored in the way they are designed by the developer, in the Google App Engine queries are run differently, each of the queries is managed by an index which itself is further managed by the Datastore. The Datastore also has more complex structure than many other NoSQL Datastores have. The first level, much like traditional relational databases, is a table holding data. Each table is split into multiple rows, with each row addressed with a unique key string. The values inside the row are arranged into cells, with each cell identified by a column family identifier, a column name, and a timestamp. The row keys are stored in ascending order within file chunks called shards. This ensures that operations accessing continuous ranges of keys are efficient [7]. When the application runs a query, the Datastore finds the corresponding index, it proceeds by scanning the first row which matches the index and returns the entity for each of the rows linked to the index till the first row which does not match query. Google App Engine provides some indexes for simple queries, while for the complex queries the application itself must have additional information for indexes during the configuration phase. The engine itself offers the possibility of creating a configuration file in which the indexes used during the test phase can be used. Through transaction features processes are performed in the way that the changes are done in full or in case of any failure they are rolled back to the previous state so that during the multiple simultaneous accesses the data are in coherent state. When the commands are called though the Datastore API, the result is returned to the caller only after the transaction is successfully performed. The App Engine Datastore is primarily designed and optimized for completely different usage scenarios and performance characteristics than relational storages.

The code listed below, in Java programing language, depicts the case when an entity of Student kind is created together with some features and the types of data used to store the student data on the datastore.

```
// datastore initiation
DatastoreService datastore = new DatastoreServiceFactory().getDatastoreService();
Entity stu = new Entity("Student");
stu.setProperty("firstName", req.getParameter("firstName_input"));
stu.setProperty("lastName", req.getParameter("lastName_input"));
stu.setProperty("Email",
```

```
req.getParameter("email_input"));
  //Save the data on the datastore
  datastore.put(stu);
```

6. Google App Engine and Cloud Computing

Google App Engine represent the service for hosting web applications which usually are accessed through the web browsers which can be in form of social networks, games, mobile applications, publications etc. The engine also can serve to standard applications such as documents, images, but its primary usage is for real time dynamic applications [8]. Especially it is designed for storing applications with multiple users which are simultaneously accessed, the engine manages to do this by scaling. With the increase of the number of users, the engine allocates more resources with the applications not having to worry about the processes that happen during the adaption process of the resources. The engine is composed of three main components [6]: applications instances, data store features and services. The engine handles the requests by identifying the application from the name of the domain of the address; it proceeds by offering one the servers which in most of the cases is the server which offers the fastest response. In order for the resources to be used as much as possible and avoiding repeating initializations, the engine allows longer execution environment than the handling of the requests. Each of the instances has local memories for storing the imported source code and data structures. The engine handles instances creation and destruction based on the traffic needs. Different from the traditional way of accessing the applications, the application code does not access the server in that way, the application can read the files form the system files but it cannot write on them and it cannot read other applications files. Google app engine offers three execution environments: Java environment, Python and the environment based on the language called go. Static files which do not change on the web sites, such as html files, CSS, images and other non-dynamic files is not necessary to be taken to much care when programming in Google App Engine, these files are managed by the dedicated server for the static files.

Today Cloud Computing techniques are being used massively also thanks to the advancements in the information technology infrastructures. This increase leads to the need of advancement of the human and hardware resources in order to maintain the systems which are built using such techniques. The decision to use the external resources is proven to be a good choice for the companies who use those services because of the efficiency which is provided by the service providers. The companies which offer services based on cloud computing technologies usually provide their services in one of the following ways [9]:

- Hardware as a Service (HaaS): As the result of rapid advances in hardware virtualization, IT automation and usage metering & pricing, users could buy IT hardware, or even an entire data center, as a pay-as-you-go subscription service.
- Software as a Service (SaaS): software or an application is hosted as a service and provided to customers across the Internet. This mode eliminates the need to install and run the application on the customer's local computers. SaaS therefore alleviates the customer's burden of software maintenance, and reduces the expense of software purchases by on-demand pricing.
- Data in various formats and from multiple sources could be accessed via services by users on the network. Users could, for example, manipulate the remote data just like they operate on a local disk or access the data in a semantic way in the Internet.

Cloud computing can be used to write versatile applications, although cloud computing can efficiently be used to develop almost all kinds of the applications, including standalone ones, it is advisable to write the applications by using cloud computing techniques should they fall in on one of the categories [10]:

- Collaborative applications: if the application are developed to be used by a group of people working together, share data, communicate or collaborate the application should be built using cloud computing technologies.
- In cases when the application is of nature of services, then cloud based techniques is ideal choice. Even though the difference between the applications and services is quite narrow, there are sometime some applications which look like services in fact they are centralized applications.

If the applications intend to perform massive computation which with user's computer cannot be carried out, then the cloud computing techniques are ideal to perform such computation by offering timely access on the resources. Typical example of this kind of application is research teams working on big data projects.

Cloud Computing uses technology, services and the applications similar to those based in web using them as tools to build services, when using the word cloud two main concepts are denoted:

Abstractions: Many details of the implementation are not made known to the developers and the users of the applications. Applications are run in the physical layer and the data are stored in unknown locations; administration is done by the service provide and the access in enabled to all those involved.

Virtualization: The resources are shared were the systems are managed by the needs arising from the main infrastructure; the cost of the service is dependent from the volume of the usage of the service.

Cloud computing usually is of two main models:

Deployment models: represents the location and management of the cloud infrastructure

Service models: represents the specific service which can be accessed through the cloud computing platform.

7. GQL – Google Query Language

Google Query Language provides an efficient way of accessing data. The version of GQL implemented here is based on the Google Visualization API Query Language [11]. Even though the syntax is similar to SQL used in much database software it has some limitations. Query formulation in the execution environment of the engine such as Java and Python is performed in the way that the Datastore is required to return the entities or the keys that meet the different criteria.

Every GQL query always begins with SELECT

*, SELECT key or SELECT <property list>, where property is a comma delimited list of one or more entity properties to be returned from the query. The optional DISTINCT clause specifies that only completely unique results will be returned in a result set. This will only return the first result for entities which have the same values for the properties that are being projected.

The optional FROM clause limits the result set to those entities of the given kind. A query without a FROM clause is called a kind less query and cannot include filters on properties.

The optional WHERE clause filters the result, set to those entities that meet one or more conditions. Each condition compares a property of the entity with a value using a comparison operator. If multiple conditions are given with the AND keyword, then an entity must meet all of the conditions to be returned by the query. GQL does not have an OR operator. However, it does have an IN operator, which provides a limited form of OR. The IN operator compares value of a property to each item in a list, an entity whose value for the given property equals any of the values in the list can be returned for the query.

The code below shows how the queries are written by using the tools which offers Google. In this case we have used Python programming language together with the necessary Google App Engine tools.

from google.appengine.ext import db

import webapp2from google.appengine.api import memcache

class People (db.Model): name=db.StringProperty() email = db.StringProperty() age = db.IntegerProperty()

p = People (key_name='K1',name = 'Isak Shabani', age = 37, email = 'isak.shabani@uni,pr.edu').put()

People (key_name = 'K2',name = 'Amir Kovaci',

age = 30, email = 'amir.kovaci@uni-pr.edu').put() People (key_name = 'K3',name = 'John Smith',

age = 29, email = 'johny@smith.com').put() People (key_name = 'K4',name = 'Arsim Besimi', age = 16, email = 'arsim@yahoo.com,

parent = p).put()

People (key_name = 'K5',name = 'Artan Kosova', age = 89, email = 'artan@kosova.com').put()

After executing the code above, people data (name, email, and age) will be saved on the given entity. The keys K1, K2, …., K5 are used to uniquely identify each of the records of the People entity. Google App Engine offers tools to view the entities together with their data that they contain, Datastore viewer shows the data of the entity which stores people data is shown here:

The following shows how the data are retrieved by running some queries for representing the data:

*self.response.write('<h3>People</h3>
')*

*pyetesori = "SELECT * FROM People where age >= 18 AND age <= 35" varpeople = db.GqlQuery(pyetesori) man in varpeople: self.response.write(man.email +' ' + str(man.age) +'</br>')*

self.response.write('Executed query: ' + pyetesori+'')

Two entities of the same kind can have different properties, whereas different entities can have properties with same name but with different type [12]. Although there are quite similarities with relational databases, it has quite some differences with them, Datastore is designed to process big data. Google App Engine supports also the Java programming language. In the code below we have shown the case where a Datastore of Student kind is created; it contains some properties with different types of values and saves them in a new entity.

//Datastore initialization

Datastore ServiceDatastore = new DatastoreServiceFactory().getDatastoreService(); Entitystu = new Entity("Student");

stu.setProperty ("Name", req.getParameter ("name_input"));

stu.setProperty ("Surname", req.getParameter ("surname_input"));

stu.setProperty ("Email", req.getParameter ("email_input"));

//Save the data into the data store

Datastore.put(stu);

Save, display and delete of the entities is done by using the corresponding commands of the Datastore. Data can be retrieved from the store by using getDatastoreService method of DatastoreServiceFactory class. Programming in Google App Engine by using both of the programming languages Java and Python yields great applications. Python is the lightweight dynamic language; Java offers the most enriched features, its Google Web Toolkit (GWT) offers to the programmer many tools to easily develop the applications. The main aspects to be considered when choosing a programming language to use are:

- Strong typing: can catch many kinds of programming errors when the program is compiled. This is particularly valuable in an environment like the cloud, where it's harder to debug your program. It is difficult to just fire up a debugger and probe it.
- Style: As it is presented above, developing a cloud application in Java has a very different Style and structure from Python. For some developers, the style of Java development in App Engine can be much more comfortable than Python.
- Tools: Google released a set of plugins for the free Eclipse IDE for building Java/GWT App Engine services and applications.

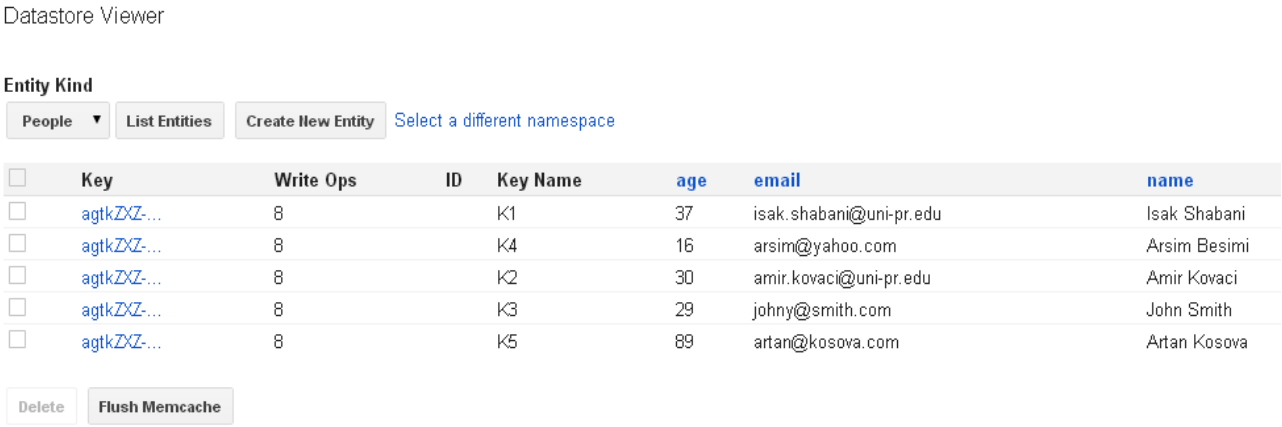

Fig. 1 Datastoreviewer shows the data of the entity which stores people data.

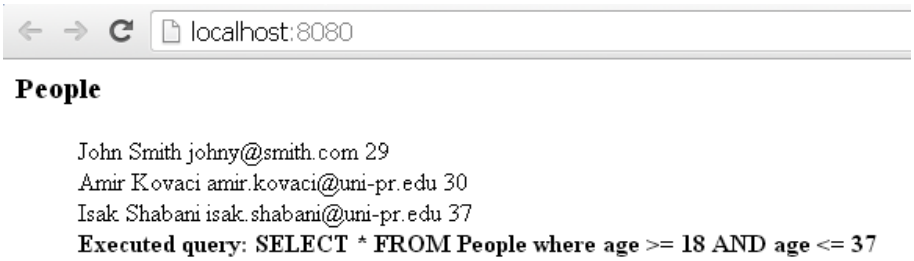

Fig. 2 Retrieved data from the Datastore.

8. Conclusions

In this paper we have aimed to depict some of benefits which Google through its App Engine offers to build cloud based applications by using the Google infrastructure. Thus we have discussed about the GFS, Datastore and its foundation core BigTable as powerful tool to store and serve highly reliable data to the end users. The programming features of Google App Engine were also introduced by emphasizing the two main programming languages which its supports: Java and Python. The data storage approach which Google offers to the companies and the developers to design distributed applications enables them to get into development by using many features which many Relational Databases Software similarly share; SQL syntax found in most of the database software tools is similar to GQL though some substantial changes exist.

References

[1] Birman, Kenneth P., and Thomas A. Joseph. "Reliable communication in the presence of failures." ACM Transactions on Computer Systems (TOCS) 5.1 (1987): 47-76., accessed on [30/09/2014]

[2] Bunch, Chris, et al. "An evaluation of distributed Datastores using the AppScale cloud platform." Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on. IEEE, 2010.

[3] Wang, Lizhe, et al. "Cloud computing: a perspective study." New Generation Computing 28.2 (2010): 137-146.

[4] Armbrust, Michael, et al. "A view of cloud computing." Communications of the ACM 53.4 (2010): 50-58.

[5] Buyya, Rajkumar, Chee Shin Yeo, and Srikumar Venugopal. "Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities." High Performance Computing and Communications, 2008. HPCC'08. 10th IEEE International Conference on. Ieee, 2008

[6] Chang, Fay, et al. "BigTable: A distributed storage system for structured data." ACM Transactions on Computer Systems (TOCS)

[7] (2008): 4.

[8] De Jonge, Adrian. Essential app engine: building high-performance java apps with google app engine. Addison-Wesley Professional, 2011

[9] Wolfgang Emerich: Distributed Systems Principles: http://www0.cs.ucl.ac.uk/staff/w.emmerich/lectures/ds98-99/dsee3.pdf, accessed on [11/10/2014]

[10] Scientific Cloud Computing: Early Definition and Experience: http://cyberaide.googlecode.com/svn/trunk/papers/08-cloud/vonLaszewski-08-cloud.pdf, 2008, accessed on [05/10/2014]

[11] Chu-Carroll, Mark C. Code in the Cloud. Pragmatic Bookshelf, 2010.

[12] Querly Language Reference: https://developers.google. com/chart/interactive/docs/querylanguag e, accessed on [20/09/2014]

[13] Chang, Fay, et al. "BigTable: A distributed storage system for structured data."ACM Transactions on Computer Systems (TOCS) 26.2 (2008): 4.

[14] Sosinsky, Barrie. Cloud computing bible. Vol. 762. John Wiley & Sons, 2010.

[15] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." National Institute of Standards and Technology 53.6 (2009): 50.