Seminar Report: Google App Engine

By:
Parag Jain
Google App Engine

INDEX

1. Introduction to cloud Computing
2. Need for cloud computing
3. Types
4. What is Google App Engine?
5. The Application Run-Time Environment
6. The Java Run-Time Environment
7. The Python Run-Time Environment
8. The Data Store
9. Quotas and limit
10. Restrictions
11. Advantage and disadvantage
1. INTRODUCTION

Cloud computing is the next natural step in the evolution of on-demand information technology services and products. To a large extent cloud computing will be based on virtualized resources. The idea of cloud computing is based on a very fundamental principal of `reusability of IT capabilities`. The difference that cloud computing brings compared to traditional concepts of —grid computing||, —distributed computing, —utility computing, or —autonomic computing is to broaden horizons across organizational boundaries.

According to the IEEE Computer Society Cloud Computing is:

"A paradigm in which information is permanently stored in servers on the Internet and cached temporarily on clients that include desktops, Entertainment centers, table computers, notebooks, wall computers, handhelds, etc."

Though many cloud computing architectures and deployments are powered by grids, based on autonomic characteristics and consumed on the basis of utilities billing, the concept of a cloud is fairly distinct and complementary to the concepts of grid, SaaS , Utility Computing etc. In theory, cloud computing promises availability of all required hardware, software, platform, applications, infrastructure and storage with an ownership of just an internet connection. people can access the information that they need from any device with an Internet connection—including mobile and handheld phones—rather than being chained to the desktop. It also means lower costs, since there is no need to install software or hardware.

Cloud computing used to posting and sharing photos on orkut, instant messaging with friends maintaining and upgrading business technology.
Need for Cloud Computing

What could we do with 1000 times more data and CPU power? One simple question. That’s all it took the interviewers to bewilder the confident job applicants at Google. This is a question of relevance because the amount of data that an application handles is increasing day by day and so is the CPU power that one can harness.

There are many answers to this question. With this much CPU power, we could scale our businesses to 1000 times more users. Right now we are gathering statistics about every user using an application. With such CPU power at hand, we could monitor every single user click and every user interaction such that we can gather all the statistics about the user. We could improve the recommendation systems of users. We could model better price plan choices. With this CPU power we could simulate the case where we have say 1,00,000 users in the system without any glitches.

There are lots of other things we could do with so much CPU power and data capabilities. But what is keeping us back. One of the reasons is the large scale architecture which comes with these are difficult to manage. There may be many different problems with the architecture we have to support. The machines may start failing, the hard drives may crash, the network may go down and many other such hardware problems. The hardware has to be designed such that the architecture is reliable and scalable. This large scale architecture has a very expensive upfront and has high maintenance costs. It requires different resources like machines, power, cooling, etc. The system also cannot scale as and when needed and so is not easily reconfigurable.

The resources are also constrained by the resources. As the applications become large, they become I/O bound. The hard drive access speed becomes a limiting factor. Though the raw CPU power available may not be a factor, the amount of RAM available clearly becomes a factor. This is also limited in this context. If at all the hardware problems are managed very well, there arises the software problems. There may be bugs in the software using this much of data. The workload also demands two important tasks for two completely different people. The software has to be such that it is bug free and has good data processing algorithms to manage all the data.
The cloud computing works on the cloud - so there are large groups of often low-cost servers with specialized connections to spread the data-processing chores among them. Since there are a lot of low-cost servers connected together, there are large pools of resources available. So these offer almost unlimited computing resources. This makes the availability of resources a lesser issue. The data of the application can also be stored in the cloud. Storage of data in the cloud has many distinct advantages over other storages. One thing is that data is spread evenly through the cloud in such a way that there are multiple copies of the data and there are ways by which failure can be detected and the data can be rebalanced on the fly. The I/O operations become simpler in the cloud such that browsing and searching for something in 25GB or more of data becomes simpler in the cloud, which is nearly impossible to do on a desktop.

The cloud computing applications also provide automatic reconfiguration of the resources based on the service level agreements. When we are using applications out of the cloud, to scale the application with respect to the load is a mundane task because the resources have to be gathered and then provided to the users. If the load on the application is such that it is present only for a small amount of time as compared to the time its working out of the load, but occurs frequently, then scaling of the resources becomes tedious. But when the application is in the cloud, the load can be managed by spreading it to other available nodes by making a copy of the application on to them. This can be reverted once the load goes down. It can be done as and when needed. All these are done automatically such that the resources maintain and manage themselves.

**Cloud –Types**

**Public cloud:**
Public cloud or external cloud describes cloud computing in the traditional mainstream. Public clouds are run by third parties, and applications from different customers are likely to be mixed together on the cloud’s servers, storage systems, and networks. A public cloud provides services to multiple customers.

**Hybrid cloud:**
Hybrid clouds combine both public and private cloud models. This is most often seen with the use of storage clouds to support Web 2.0 applications.

**Private cloud:**
Private clouds are built for the exclusive use of one client, providing the utmost
control over data, security, and quality of service (Figure 4). The company owns the infrastructure and has control over how applications are deployed on it. Private clouds can be built and managed by a company’s own IT organization or by a cloud provider.

Cloud computing products and services can be classified into 4 major categories:

They are
1. Application as service (AaaS)
2. Platform as a Service (PaaS)
3. Infrastructure as a service (IaaS)
4. Software as a Service (SaaS)

1. Application as service (AaaS): These are the first kind of cloud computing services that came into being. Under this, a service is made available to an end-user. The end-user is asked to create an account with the service provider and start using the application. One of first famous application was web-based email service by hotmail started in 1996. Scores of such services are available now on the web.

2. Platform as a Service (PaaS): Cloud vendors are companies that offer cloud computing services and products. One of the services that they provide is called PaaS. Under this a computing platform such as operating system is provided to a customer or end user on a monthly rental basis. Some of the major cloud computing vendor are Amazon, Microsoft, Google etc

3. Infrastructure as a service: The cloud computing vendors offer infrastructure as a service. One may avail hardware services such as processors, memory, networks etc on agreed basis for specific duration and price.

4. Software as a Service (SaaS): Software package such as CRM or CAD/CAM can be accessed under cloud computing scheme. Here a customer upon registration is allowed to use software accessible through net and use it for his or his business
process. The related data and work may be stored on local machines or with the service providers. SaaS services may be available on rental basis or on per use basis.

What Is Google App Engine?

Google App Engine lets you run your web applications on Google's infrastructure. App Engine applications are easy to build, easy to maintain, and easy to scale as your traffic and data storage needs grow. With App Engine, there are no servers to maintain: You just upload your application, and it's ready to serve your users.

You can serve your app from your own domain name (such as http://www.example.com/) using Google Apps. Or, you can serve your app using a free name on the appspot.com domain. You can share your application with the world, or limit access to members of your organization.

Google App Engine supports apps written in several programming languages. With App Engine's Java runtime environment, you can build your app using standard Java technologies, including the JVM, Java servlets, and the Java programming language—or any other language using a JVM-based interpreter or compiler, such as JavaScript or Ruby. 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 your application runs quickly, securely, and without interference from other apps on the system.

With App Engine, you only pay for what you use. There are no set-up costs and no recurring fees. The resources your application uses, such as storage and bandwidth, are measured by the gigabyte, and billed at competitive rates. You control the maximum amounts of resources your app can consume, so it always stays within your budget. App Engine costs nothing to get started. All applications can use up to 500 MB of storage and enough CPU and bandwidth to support an efficient app serving
around 5 million page views a month, absolutely free. When you enable billing for your application, your free limits are raised, and you only pay for resources you use above the free levels.

**The Application Runtime Environment**

Google App Engine makes it easy to build an application that runs reliably, even under heavy load and with large amounts of data. App Engine includes the following features:

- dynamic web serving, with full support for common web technologies
- persistent storage with queries, sorting and transactions
- automatic scaling and load balancing
- APIs for authenticating users and sending email using Google Accounts
- a fully featured local development environment that simulates Google App Engine on your computer
- task queues for performing work outside of the scope of a web request
- scheduled tasks for triggering events at specified times and regular intervals

Your application can run in one of two runtime environments: the **JAVA** environment, and the **PYTHON** environment. Each environment provides standard protocols and common technologies for web application development.

**The Java Runtime Environment**

You can develop your application for the Java runtime environment using common Java web development tools and API standards. Your app interacts with the environment using the Java Servlet standard, and can use common web application technologies such as JavaServer Pages (JSPs).

The Java runtime environment uses Java 6. The App Engine Java SDK supports developing apps using either Java 5 or 6.

The environment includes the Java SE Runtime Environment (JRE) 6 platform and libraries. The restrictions of the sandbox environment are
implemented in the JVM. An app can use any JVM bytecode or library feature, as long as it does not exceed the sandbox restrictions. For instance, bytecode that attempts to open a socket or write to a file will throw a runtime exception.

Your app accesses most App Engine services using Java standard APIs. For the App Engine datastore, the Java SDK includes implementations of the Java Data Objects (JDO) and Java Persistence API (JPA) interfaces. Your app can use the JavaMail API to send email messages with the App Engine Mail service.

The java.net HTTP APIs access the App Engine URL fetch service. App Engine also includes low-level APIs for its services to implement additional adapters, or to use directly from the application. See the documentation for the datastore, URL fetch, mail, images and Google Accounts APIs.

Typically, Java developers use the Java programming language and APIs to implement web applications for the JVM. With the use of JVM-compatible compilers or interpreters, you can also use other languages to develop web applications, such as JavaScript, Ruby, or Scala.

For more information about the Java runtime environment, see The Java Runtime Environment.

The Python Runtime Environment

With App Engine's Python runtime environment, you can implement your app using the Python programming language, and run it on an optimized Python interpreter. App Engine includes rich APIs and tools for Python web application development, including a feature rich data modeling API, an easy-to-use web application framework, and tools for managing and accessing your app's data. You can also take advantage of a wide variety of mature libraries and frameworks for Python web application development, such as Django.

The Python runtime environment uses Python version 2.5.2. Additional support for Python 3 is being considered for a future release.

The Python environment includes the Python standard library. Of course, not all of the library's features can run in the sandbox environment. For instance, a call to a method that attempts to open a socket or write to a file will raise an exception. For convenience, several modules in the standard library whose core features are not supported by the runtime environment have been disabled, and code that imports them will raise an error.

Application code written for the Python environment must be written exclusively in Python. Extensions written in the C language are not supported.
The Python environment provides rich Python APIs for the datastore, Google Accounts, URL fetch, and email services. App Engine also provides a simple Python web application framework called webapp to make it easy to start building applications. You can upload other third-party libraries with your application, as long as they are implemented in pure Python and do not require any unsupported standard library modules.

**The Datastore**

App Engine provides a distributed data storage service that features a query engine and transactions. Just as the distributed web server grows with your traffic, the distributed datastore grows with your data. You have the choice between two different data storage options differentiated by their availability and consistency guarantees.

The App Engine datastore is not like a traditional relational database. Data objects, or "entities," have a kind and a set of properties. Queries can retrieve entities of a given kind filtered and sorted by the values of the properties. Property values can be of any of the supported property value types. Datastore entities are "schemaless." The structure of data entities is provided by and enforced by your application code. The Java JDO/JPA interfaces and the Python datastore interface include features for applying and enforcing structure within your app. Your app can also access the datastore directly to apply as much or as little structure as it needs.

The datastore is strongly consistent and uses optimistic concurrency control. An update of an entity occurs in a transaction that is retried a fixed number of times if other processes are trying to update the same entity simultaneously. Your application can execute multiple datastore operations in a single transaction which either all succeed or all fail, ensuring the integrity of your data. The datastore implements transactions across its distributed network using "entity groups." A transaction manipulates entities within a single group. Entities of the same group are stored together for efficient execution of transactions. Your application can assign entities to groups when the entities are created.

**Quotas and limits**

Not only is creating an App Engine application easy, it's free! You can create an account and publish an application that people can use right away at no charge, and with no obligation. An application on a free account can use up to 500MB of storage.
storage and up to 5 million page views a month. When you are ready for more, you can enable billing, set a maximum daily budget, and allocate your budget for each resource according to your needs.

**Free quotas**

Application creators who enable billing pay only for CPU, bandwidth, storage, and e-mails used in excess of the free quotas. Limits marked with * are increased for application authors who enable billing, even if their application never uses enough resources to incur charges. Free quotas were reduced on May 25, 2009 and were reduced again on June 22, 2009.

<table>
<thead>
<tr>
<th>Quota</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emails per day</td>
<td>2,000</td>
</tr>
<tr>
<td>Bandwidth in per day</td>
<td>1 GB</td>
</tr>
<tr>
<td>Bandwidth out per day</td>
<td>1 GB</td>
</tr>
<tr>
<td>CPU time per day</td>
<td>6.5 hours per day</td>
</tr>
<tr>
<td>HTTP Requests per Day</td>
<td>1,300,000*</td>
</tr>
<tr>
<td>Datastore API calls per day</td>
<td>10,000,000*</td>
</tr>
<tr>
<td>Data stored</td>
<td>1 GB</td>
</tr>
<tr>
<td>URLFetch API calls per day..</td>
<td>657,084*</td>
</tr>
</tbody>
</table>
App Engine defines usage quotas for free applications. Extensions to these quotas can be requested, and application authors can pay for additional resources.

**Hard limits**

<table>
<thead>
<tr>
<th>Quota</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apps per developer</td>
<td>10</td>
</tr>
<tr>
<td>Time per request</td>
<td>30 sec</td>
</tr>
<tr>
<td>Blobstore size (total file size per app)</td>
<td>2 GB</td>
</tr>
<tr>
<td>HTTP response size</td>
<td>10 MB</td>
</tr>
<tr>
<td>Datastore item size</td>
<td>1 MB</td>
</tr>
<tr>
<td>Application code size</td>
<td>150MB</td>
</tr>
</tbody>
</table>
**Restrictions**

- Developers have read-only access to the file system on App Engine. Your application can use only virtual file systems, like gae-filestore.
- App Engine can only execute code called from an HTTP request (scheduled background tasks allow for self calling HTTP requests).
- Users may upload arbitrary Python modules, but only if they are pure-Python; C and Pyrex modules are not supported.
- Java applications may only use a subset (The JRE Class White List) of the classes from the JRE standard edition.
- Java applications cannot create new threads.
- Does not support 'naked' domains (without www) like http://example.com. The required alias to ghs.google.com is implemented with a DNS CNAME record in order for changes in Google server IP addresses not to impact the service. This record cannot be used with other DNS records (RFC 1034 section 3.6.2, RFC 1912 section 2.4), including the required Start of Authority for the example.com DNS zone. Suggested workaround is to use the domain registrar HTTP redirection to a subdomain, e.g. "www.example.com".
- SSL/HTTPS is only available via *.appspot.com domains and not via Google Apps Domains.
- Datastore cannot use inequality filters on more than one entity property per query.
- A process started on the server to answer a request can't last more than 30 seconds. (with the 1.4.0 release, this restriction does not apply to background jobs anymore)
- Does not support sticky sessions (a.k.a. session affinity), only replicated sessions are supported including limitation of the amount of data being serialized and time for session serialization
Advantage and Disadvantage

➢ **Advantage**

• Easy to get Started
• The reliability, performance and security of Google's infrastructure
• Cost efficient hosting
• Risk free trial period

➢ **Disadvantage**

• Developers have read-only access to the filesystem on App Engine.
• App Engine can only execute code called from an HTTP request
• Java applications may only use a subset (The JRE Class White List) of the classes from the JRE standard edition.
• Java applications cannot create new threads.
• Does not support 'naked' domains (without www) like http://example.com.
• A process started on the server to answer a request can't last more than 30 seconds.