Unit 3 Cloud computing platforms

INFRASTRUCTURE AS A SERVICE: AMAZON EC2

The Amazon cloud provides infrastructure as a service (IaaS), It provides computing infrastructure such as for servers, storage or network end points of a desired capacity are virtually provided in minutes through an automated web-based management console.

This core IaaS service, called Elastic Compute Cloud, or EC2, EC2 is also often used to describe the entire cloud offering. Figure 5.1 illustrates the services provided by the Amazon infrastructure cloud from a user perspective. These services are implemented on a very large network of servers, shown as dashed boxes in the figure. The Elastic Compute Cloud service provides users access to dedicated virtual machines of a desired capacity that are provisioned on these physical servers, with details of the actual physical server, such as its location, capacity, etc. being transparent to the end-user.

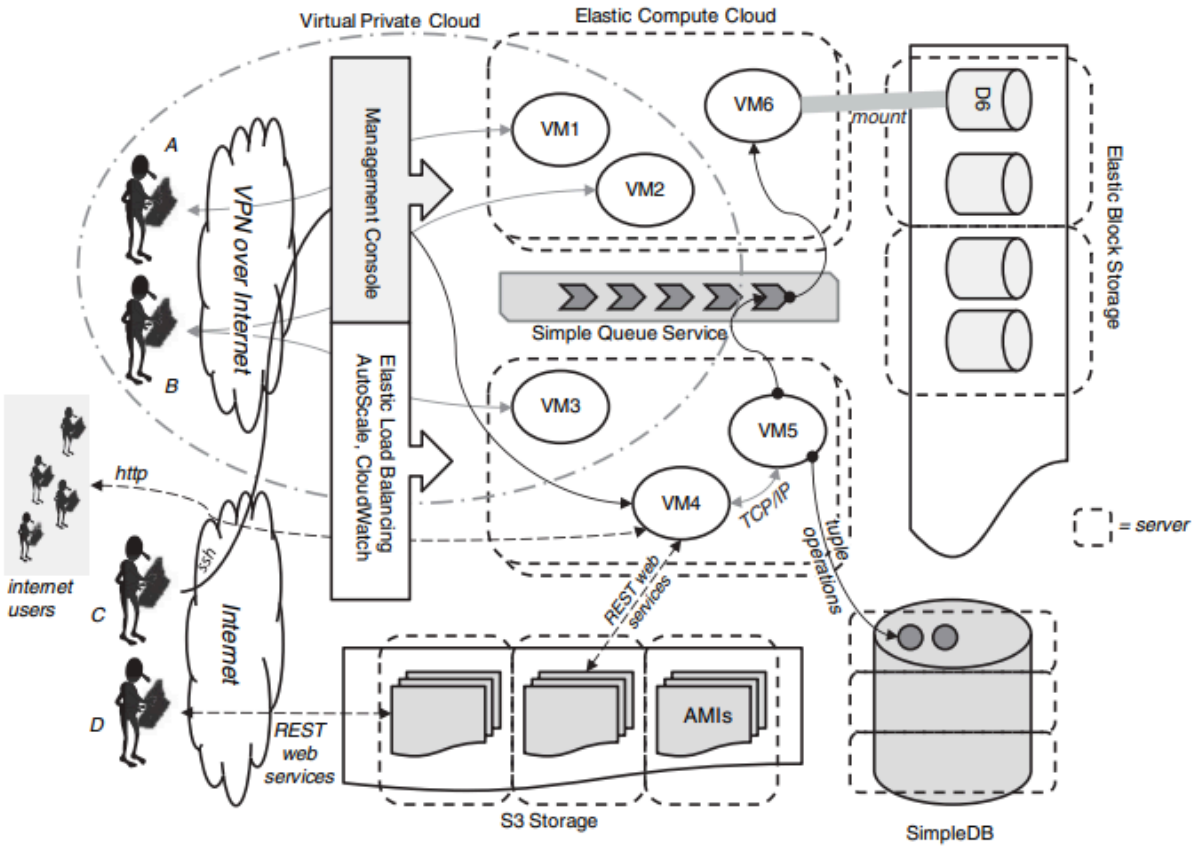

FIGURE 5.1. Amazon infrastructure cloud

Through the management console users generate PKI1 key-pairs using which they can securely login to these virtual servers over the internet.

In Figure 5.1, user C provisions the virtual server VM4 through the management console, and accesses it using ssh, for a Linux server, or via 'remote desktop' for a Windows server.

All AMIs are stored in common storage in the Amazon S3 storage service and used to boot the desired configuration of virtual server.

The user's account is charged on an hourly basis based on actual consumption, i.e. time that the server is up.

Charges vary depending on the AMI used and capacity of server chosen while provisioning.

For example, a 'small' Linux server costs a few cents per cpu-hour, whereas a larger server preloaded with licensed software, such as Windows, as well as other database or middleware products, could end up costing close to a dollar per hour.

Cloud users have root/administrator access to these servers, and therefore control them completely.

For example, they can deploy applications and make them publicly accessible over the internet.

Static network addresses required for such purposes can also be provisioned through the management console.

Thus, VM4 is also accessible by internet users at large over HTTP. Such publicly available static IP addresses are charged on a fixed monthly basis

Network data transfer to any server, with or without a static IP address, is charged on usage basis. Users can provision and access many servers that can communicate with each other over the fast internal network within the Amazon cloud.

If VM4 is a web server, VM5 may be a database server, and these two communicate over the internal cloud network using TCP/IP.

Since VM5 is a database server, it needs to store and retrieve data.

The Amazon SimpleDB service provides an object store where key-value pairs can be efficiently stored and retrieved.

It is important to note that SimpleDB is not a relational database, and we shall describe the features provided by such key-value pair cloud databases in a later section.

Instead of using SimpleDB, virtual servers could instead use a relational database system, which may come either pre-installed as part of the AMI, or separately by users in the normal manner. However, it is important to understand that virtual servers do not have any persistent storage; so any user data on file system is lost when the server shuts down. In order to store data persistently, such in a relational database,

Elastic Block Storage needs to be mounted on a virtual server.

The Elastic Block Storage service maintains persistent data across all users on a large set of physical servers.

After a virtual server boots, it must attach user data from the EBS as a logical storage volume mounted as a raw device (disk).

VM6 might be an archival server where VM5 sends logs of whatever updates it makes to the SimbleDB datastore. VM5 sends data to VM6 not over TCP/IP, but using the Amazon Simple Queue Service (SQS).

The SQS is a reliable persistent message queue that is useful for temporarily storing data that needs to eventually get to a processing server such as VM6,

The Amazon S3 Storage Service provides a different storage model. Data in S3 can be files of any type, and in general any blob (binary large object).

Users access and modify S3 objects via URIs, using REST web services (which we shall cover in Chapter 7). S3 objects are accessible over the internet as well as from virtual servers within the Amazon cloud.

Storage in S3 is also used for storing machine images (AMIs) that users define themselves.

Runtime performance parameters, such as CPU and I/O utilization, of a user's virtual servers can be monitored in real-time by Amazon Cloud Watch; this data can be used by Amazon Auto Scale to add (or remove) virtual servers from an application cluster and automatically provision them with predefined machine images. Finally, Elastic Load Balancing allows a group of servers to be configured into a set across which incoming requests are load balanced.

The performance statistics of the load-balanced requests can also be monitored by Cloud Watch and used by Auto Scale to add or remove servers from the load balanced cluster.

PLATFORM AS A SERVICE: GOOGLE APP ENGINE

A scalable runtime environment, Google App Engine is mostly used to run Web applications. These dynamic scales as demand change over time because of Google's vast computing infrastructure. Because it offers a secure execution environment in addition to a number of services, App Engine makes it easier to develop scalable and high-performance Web apps.

The App Engine SDK facilitates the testing and professionalization of applications by emulating the production runtime environment and allowing developers to design and test applications on their own PCs.

Google App Engine (GAE) is a service for developing and hosting Web applications in Google's data centers, belonging to the platform as a service (PaaS) category of cloud computing. Web applications hosted on GAE are sandboxed and run across multiple servers for redundancy and allow for scaling of resources according to the traffic requirements of the moment. App Engine automatically allocates additional resources to the servers to accommodate the increased load.

Google App Engine is Google's platform as a service offering that allows developers and businesses to build and run applications using Google's advanced infrastructure. These applications are required to be written in one of a few supported languages, namely:

Java, Python, PHP, and Go. It also requires the use of Google query language and the database used is Google Big Table. Applications must abide by these standards, so applications either must be developed with GAE in mind or else modified to meet the requirements.

GAE is a platform, so it provides all of the required elements to run and host Web applications, be it on mobile or the Web. Without this all-in feature, developers would have to source their own servers, database software, and the APIs that would make all of them work properly together, not to mention the entire configuration that must be done. GAE takes this burden off the developers so they can concentrate on the app front end and functionality, driving better user experience.

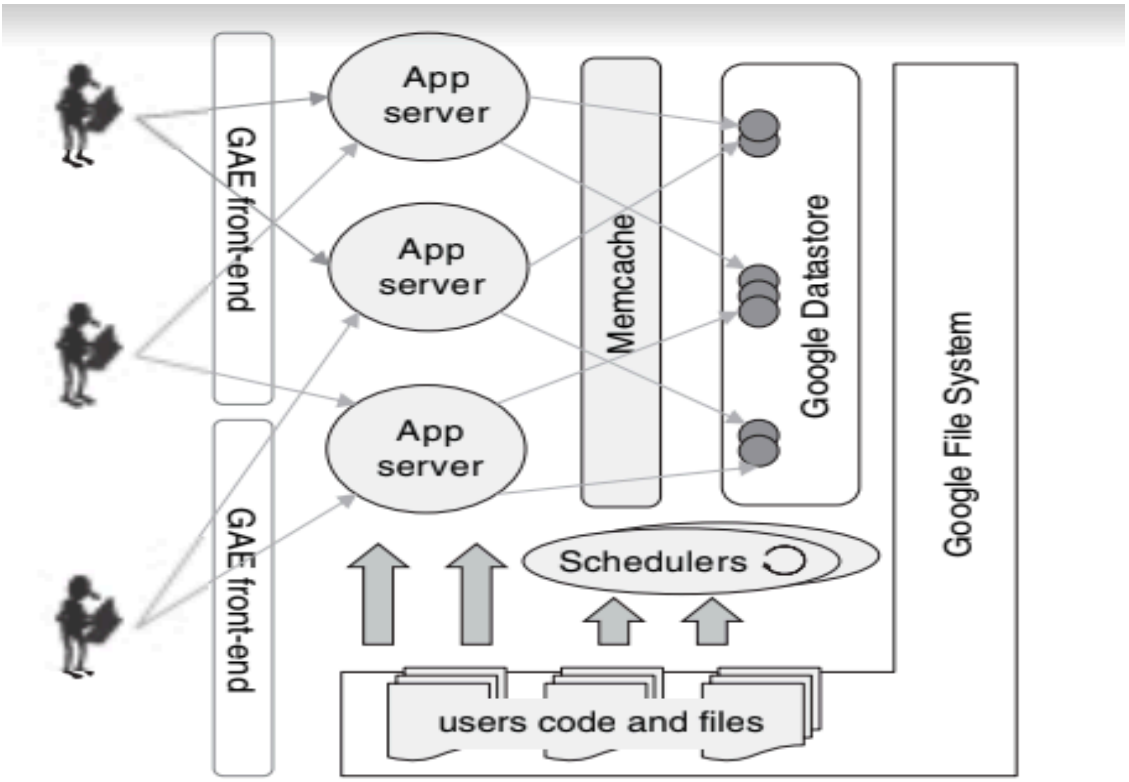

FIGURE 5.2. Google App Engine

Resource usage for an application is metered in terms of web requests served and CPU-hours actually spent executing requests or batch jobs.

Note that this is very different from the IaaS model:

A PaaS application can be deployed and made globally available 24×7, but charged only when accessed in contrast, in an IaaS model merely making an application continuously available incurs the full cost of keeping at least some of the servers running all the time. Further, deploying applications in Google App Engine is free, within usage limits; thus applications can be developed and tried out free and begin to incur cost only when actually accessed by a sufficient volume of requests.

The PaaS model enables Google to provide such a free service because applications do not run in dedicated virtual machines; a deployed application that is not accessed merely consumes storage for its code and data and expends no CPU cycles.

GAE applications are served by a large number of web servers in Google's data centers that execute requests from end-users across the globe.

The web servers load code from the GFS into memory and serve these requests. Each request to a particular application is served by any one of GAE's web servers; there is no guarantee that the same server will serve requests to any two requests, even from the same HTTP session. Applications can also specify some functions to be executed as batch jobs which are run by a scheduler.

While this architecture is able to ensure that applications scale naturally as load increases, it also means that application code cannot easily rely on in-memory data. A distributed in-memory cache called Memcache is made available to partially address this issue: In particular HTTP sessions are implemented using Memcache so that even if requests from the same session go to different servers they can retrieve their session data, most of the time (since Memcache is not guaranteed to always retain cached data).

MICROSOFT AZURE

Microsoft's cloud offering, called Azure, has been commercially released to the public only recently, though a community preview beta has been publicly available for longer. Thus, it is important to note that some elements of the Azure platform are likely to change in future commercial editions.

Like the Google App Engine, Azure is a PaaS offering.

Developers create applications using Microsoft development tools an Azure extension to the standard toolset allows such applications to be developed and deployed on Microsoft's cloud, in much the same manner as developing and deploying using Google App Engine's SDK. There are also similarities with aspects of Amazon's IaaS offering, such as the use of virtualization, user control over the number of virtual servers allocated to an application and user control on elasticity. However, unlike the non-relational Google Datastore and Amazon SimpleDB, the recently released commercial edition of Azure provides relational storage services, albeit with certain limitations as we cover below. Azure also allows storage of arbitrary files and objects like Amazon S3 as well as a queuing service similar to Amazon SQS.

Figure 5.4 illustrates a user's view of Microsoft Azure. Application code deployed on Azure executes in a number of virtual machines (called instances) with each virtual

machine running the Windows Server operating system. While users do not have any control on when instances boot and how long they stay up, they can specify the number of instances a particular application is likely to require.

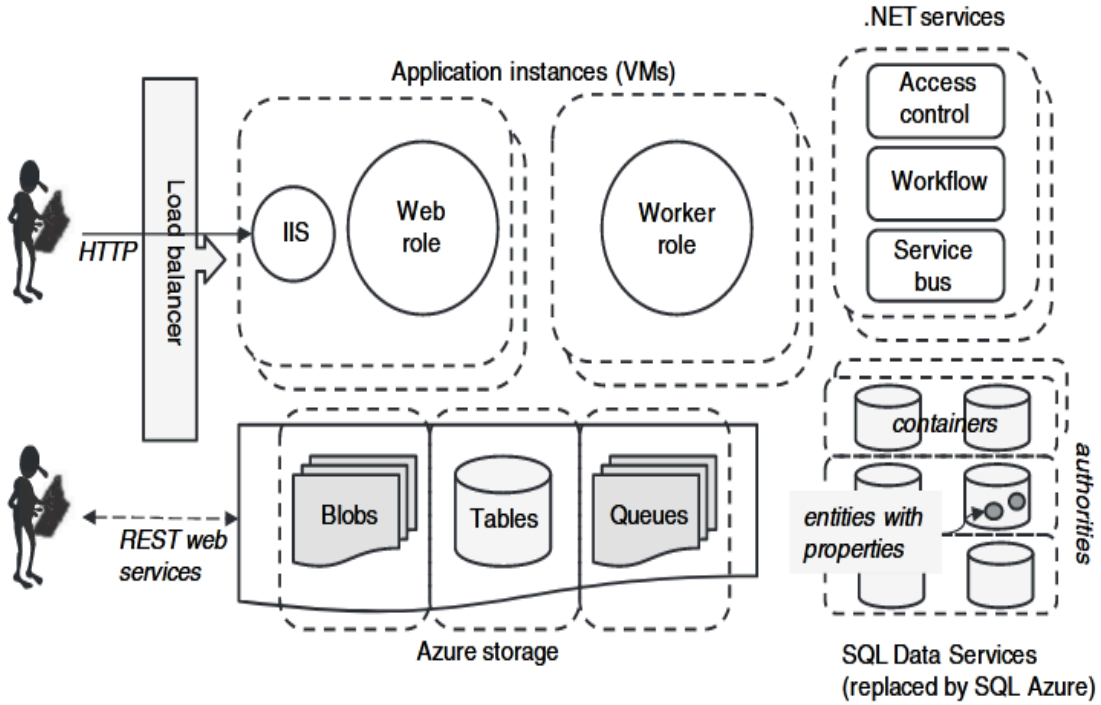

FIGURE 5.4. Microsoft Azure

Thus, Azure provides a finer level of control on the resources to be allocated to an application as compared to Google App Engine, but not to the extent of an IaaS offering such as Amazon EC2.

Application code can be deployed on Azure as a web role or a worker role. Code in web-role instances is run through a web server (Microsoft IIS) that is included in the instance. Web roles respond to HTTP requests that are automatically load balanced across application instances. Worker role code can run as a batch process that can communicate with a web role through Azure data storage, such as queues or tables. Worker role applications cannot be accessed from an external network, but can make external HTTP requests, either to worker roles or over the internet.

Azure data storage allows storage of blobs, non-relational tables (similar to SimpleDB) as well as queues. In addition, an alternative database-storage scheme was provided by SQL Data Services in the beta edition of Azure. SQL Data Services are implemented on a number of clusters running Microsoft's SQL Server database, deployed in various Microsoft-owned data centers called authorities. Each authority can host a number of containers, which are similar to database tables. Applications access SQL Data Services at the level of containers, through a global address scheme. Each container contains

entities which have properties, and like SimpleDB or Google Datastore, properties for each entity can differ in type and number. Microsoft SQL Data Services was rebranded as 'SQL Azure' in the commercial edition of Azure. The architecture of authorities, containers and tables has been replaced by a traditional relational model supported by Microsoft SQL Server. In particular this includes support for joins between tables, transactions and other features of relational database. However, as of this writing, each SQL Azure database is limited to under 10GB in size. In case larger volumes of data need to be stored, multiple virtual database instances need to be provisioned. Further, since cross-database queries are not permitted, queries on larger data sets, including joins, need to be implemented within application code. The SQL Azure model represents a practical compromise between the limitations of non-relational models versus a traditional relational database: Full relational queries are permitted, but only on small data sets, which are likely to suffice for many of the early adopters of cloud services.

In addition to compute and storage services, Microsoft Azure also provides what are called .NET services. These include access control services that provide globally configurable and accessible security tokens, a service bus that enables globally published service end points and a configurable web-service- based workflow-orchestration service. Like SQL Data services and SQL Azure,

these are all based on Microsoft's enterprise middleware products for identity management and web services, deployed in the cloud on a distributed and globally accessible platform.

As compared to Google's PaaS offering, Microsoft Azure enjoys the advantage of a large base of Microsoft applications already in use within enterprises.

From what was available in the community preview of Azure and its rapid evolution in the commercial edition (such as the addition of full relational database support, at least for small applications), it is likely to become even easier to migrate existing Microsoft applications to Azure in the future. Therefore it may well be that Microsoft Azure becomes a more popular PaaS platform

at least for large enterprises, as compared to Google's App Engine.