

US
N

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--



Internal Assessment Test 3 – Set 2 – July 2023 – Answer Key

Sub:	Cloud Computing and its Applications	Sub Code:	18CS643	Branch	ISE
Date:	05/07/2023	Duration:	90 min's	Max Marks:	50
		Sem/Sec:	VI / A and B		OBE

Answer any FIVE questions

						MARKS	CO	RBT
1	a. What is a Map Reduce model?					3		
	<p>MapReduce expresses the computational logic of an application in two simple functions: map and reduce. Data transfer and management are completely handled by the distributed storage infrastructure (i.e., the Google File System), which is in charge of providing access to data, replicating files, and eventually moving them where needed. the MapReduce model is expressed in the form of the two functions, which are defined as follows:</p> <p>The map function reads a key-value pair and produces a list of key-value pairs of different types. The reduce function reads a pair composed of a key and a list of values and produces a list of values of the same type. The types (k1,v1,k2,kv2) used in the expression of the two functions provide hints as to how these two functions are connected and are executed to carry out the computation of a MapReduce job: The output of map tasks is aggregated together by grouping the values according to their corresponding keys and constitutes the input of reduce tasks that, for each of the keys found, reduces the list of attached values to a single value. Therefore, the input of a MapReduce computation is expressed as a collection of key-value pairs < k1,v1 >, and the final output is represented by a list of values: list(v2).</p>					7		
	b. Explain the Map Reduce computation workflow with a neat diagram.							
	<p>Figure 8.5 depicts a reference workflow characterizing MapReduce computations. As shown, the user submits a collection of files that are expressed in the form of a list of < k1,v1 > pairs and specifies the map and reduce functions. These files are entered into the distributed file system that supports MapReduce and, if necessary, partitioned in order to be the input of map tasks. Map tasks generate intermediate files that store collections of < k2, list(v2) > pairs, and these files are saved into the distributed file system. These files constitute the input of reduce tasks, which finally produce output files in the form of list(v2).</p>						CO2	L2

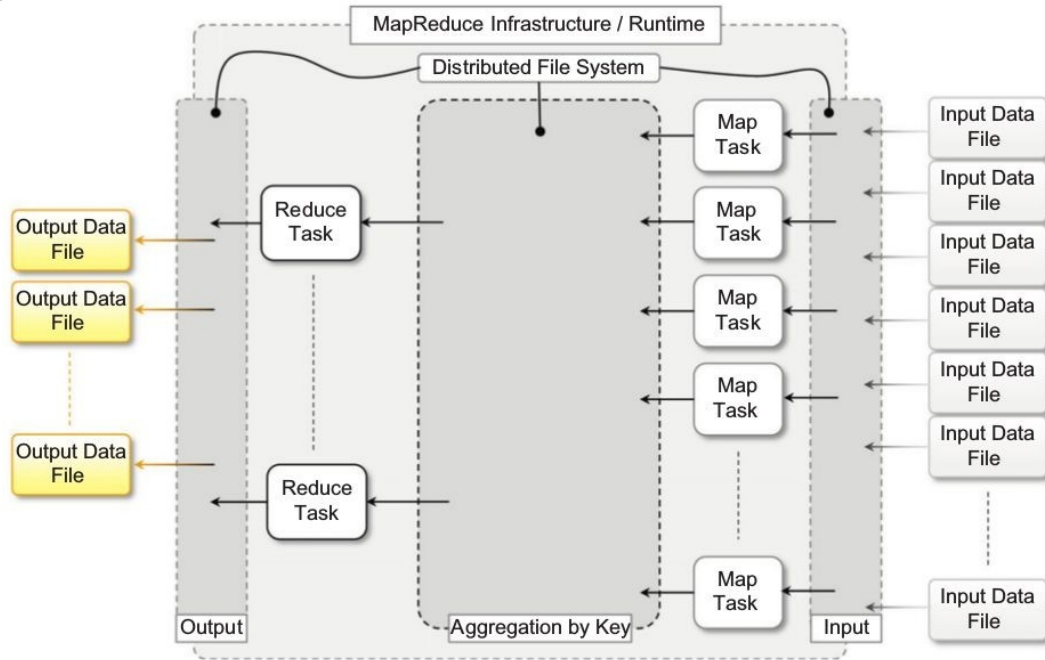


FIGURE 8.5

MapReduce computation workflow.

The computation model expressed by MapReduce is very straightforward and allows greater productivity for people who have to code the algorithms for processing huge quantities of data. In general, any computation that can be expressed in the form of two major stages can be represented in terms of MapReduce computation.

These stages are:

1. Analysis. This phase operates directly on the data input file and corresponds to the operation performed by the map task. Moreover, the computation at this stage is expected to be embarrassingly parallel, since map tasks are executed without any sequencing or ordering.
 2. Aggregation. This phase operates on the intermediate results and is characterized by operations that are aimed at aggregating, summing, and/or elaborating the data obtained at the previous stage to present the data in their final form. This is the task performed by the reduce function. Figure given below shows a more complete overview of a MapReduce infrastructure, according to the implementation proposed by Google.
- As depicted, the user submits the execution of MapReduce jobs by using the client libraries that are in charge of submitting the input data files, registering the map and reduce functions, and returning control to the user once the job is completed. A generic distributed infrastructure (i.e., a cluster) equipped with job-scheduling capabilities and distributed storage can be used to run MapReduce applications.

Two different kinds of processes are run on the distributed infrastructure:
 a master process and
 a worker process.

The master process is in charge of controlling the execution of map and reduce tasks, partitioning, and reorganizing the intermediate output produced by the map task in order to feed the reduce tasks. The master process generates the map tasks and assigns input splits to each of them by balancing the load.

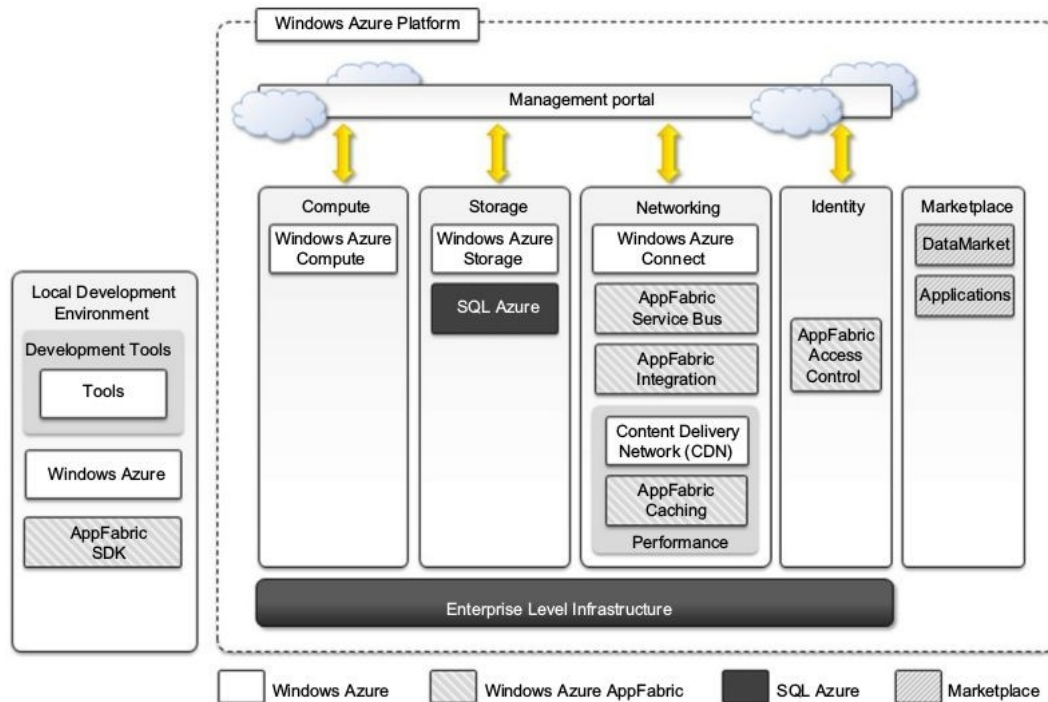
The worker processes are used to host the execution of map and reduce tasks and provide basic I/O facilities that are used to interface the map and reduce tasks with input and output files.

a. What is the Azure model?

3
7

Microsoft Windows Azure is a cloud operating system built on top of Microsoft datacenters' infrastructure and provides developers with a collection of services for building applications with cloud technology. Services range from compute, storage, and networking to application connectivity, access control, and business intelligence.

Figure 9.3 provides an overview of services provided by Azure. These services can be managed and controlled through the Windows Azure Management Portal, which acts as an administrative console for all the services.



2

CO3 L2

FIGURE 9.3

Microsoft Windows Azure Platform Architecture.

b. Describe the relationship between a process and a thread.

A thread identifies a single control flow, which is a logical sequence of instructions, within a process. By logical sequence of instructions, we mean a sequence of instructions that have been designed to be executed one after the other one. Operating systems that support multithreading identify threads as the minimal building blocks for expressing running code. Each process contains at least one thread but, in several cases, is composed of many threads having variable lifetimes. Threads within the same process share the memory space and the execution context. In a multitasking environment the operating system assigns different time slices to each process and interleaves their execution. The process of temporarily stopping the execution of one process, saving all the information in the registers, and replacing it with the information related to another process is known as a context switch.

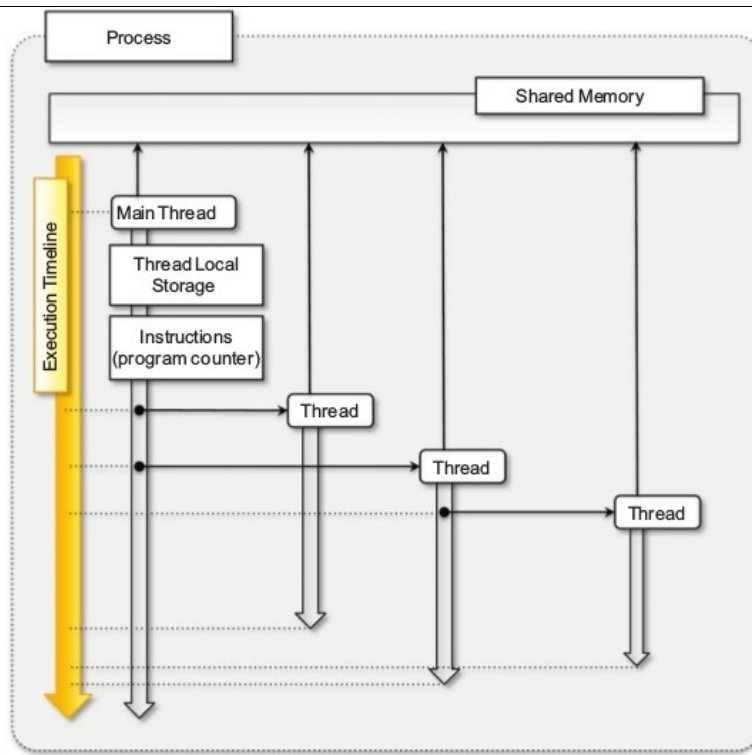


FIGURE 6.2

The relationship between processes and threads.

Figure 6.2 provides an overview of the relation between threads and processes and a simplified representation of the runtime execution of a multithreaded application. A running program is identified by a process, which contains at least one thread, also called the main thread. Such a thread is implicitly created by the compiler or the runtime environment executing the program. This thread is likely to last for the entire lifetime of the process and be the origin of other threads, which in general exhibit a shorter duration. As main threads, these threads can spawn other threads. There is no difference between the main thread and other threads created during the process lifetime. Each of them has its own local storage and a sequence of instructions to execute, and they all share the memory space allocated for the entire process. The execution of the process is considered terminated when all the threads are completed.

3 a. What are the domains for the application of Cloud Computing?

3
7

CO3 L2

Cloud computing has gained huge popularity in industry due to its ability to host applications for which the services can be delivered to consumers rapidly at minimal cost. This section discusses some application case studies, detailing their architecture and how they leveraged various cloud technologies.

Applications from a range of domains, from scientific to engineering, gaming, and social networking have cloud computing as their foundation.

Scientific applications

Healthcare: ECG analysis in the cloud

Biology: protein structure prediction

Biology: gene expression data analysis for cancer diagnosis

Geoscience: satellite image processing

Business and consumer applications

CRM and ERP

- 1 Salesforce.com
- 2 Microsoft dynamics CRM 3 NetSuite
- Productivity
- 1 Dropbox and iCloud 2 Google docs
- 3 Cloud desktops: EyeOS and XIOS/3
- Social networking 1 Facebook
- Media applications 1 Animoto
- 2 Maya rendering with Aneka
- 3 Video encoding on the cloud: Encoding.com
- Multiplayer online gaming

b. Describe how Cloud Computing can be applied to online gaming with a required diagram.

Online multiplayer gaming attracts millions of gamers around the world who share a common experience by playing together in a virtual environment that extends beyond the boundaries of a normal LAN. Online games support hundreds of players in the same session, made possible by the specific architecture used to forward interactions, which is based on game log processing.

Players update the game server hosting the game session, and the server integrates all the updates into a log that is made available to all the players through a TCP port. The client software used for the game connects to the log port and, by reading the log, updates the local user interface with the actions of other players.

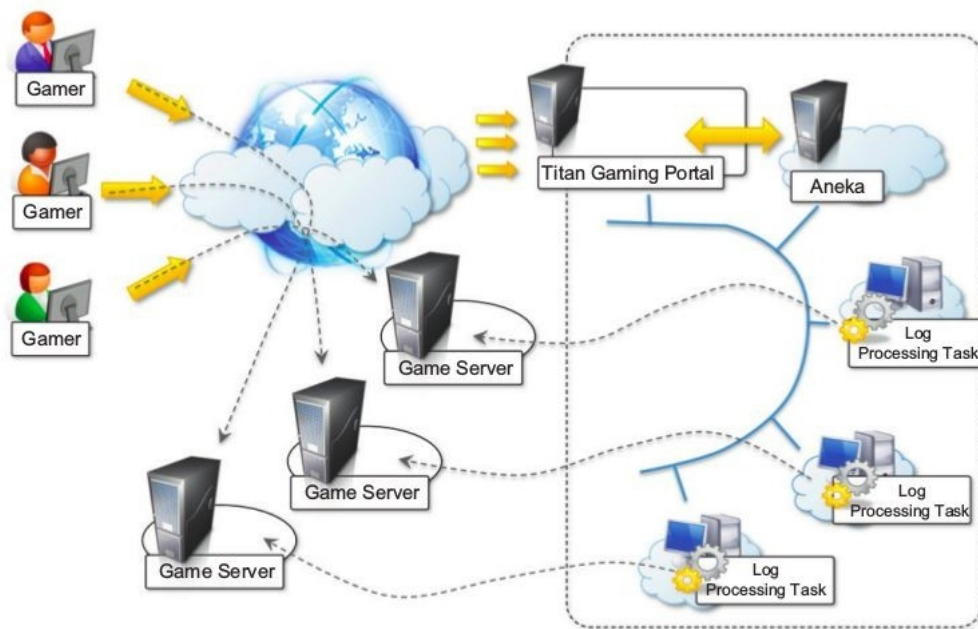
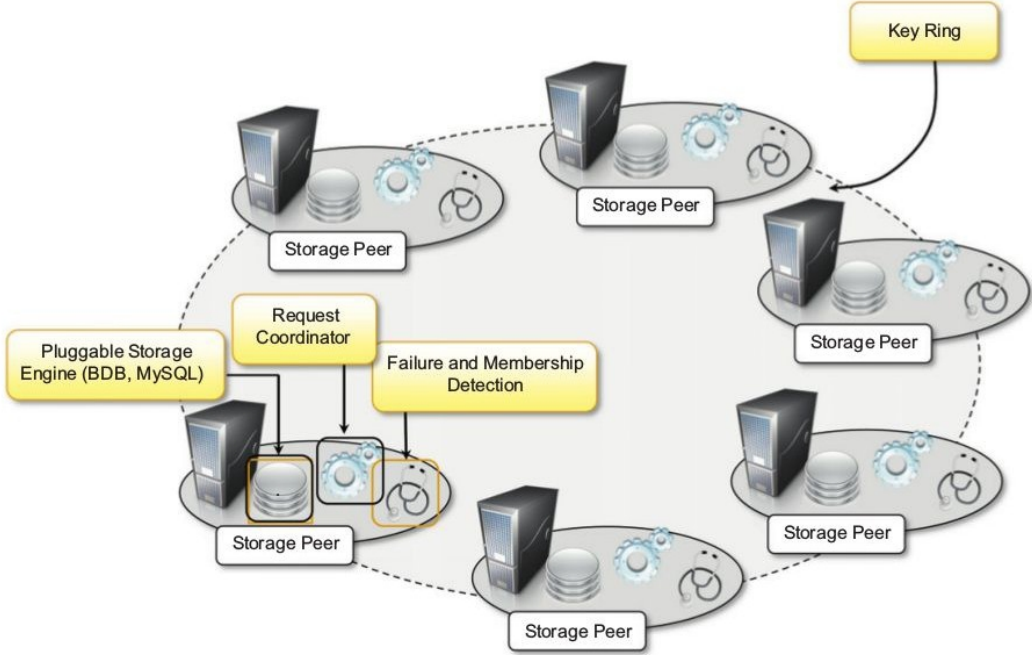


FIGURE 10.10

Scalable processing of logs for network games.

Game log processing is also utilized to build statistics on players and rank them. These features constitute the additional value of online gaming portals that attract more and more gamers. The processing of game logs is a potentially compute-intensive operation that strongly depends on the number of players online and the number of games monitored.

The use of cloud computing technologies can provide the required elasticity for seamlessly processing these workloads and scale as required when the number of

	<p>users increases. A prototype implementation of cloud-based game log processing has been implemented by Titan Inc. (Figure 10.10).</p>			
<p>4</p>	<p>a. What is the need for Azure Dynamo?</p> <p>The main goal of Dynamo is to provide an incrementally scalable and highly available storage system. This goal helps in achieving reliability at a massive scale, where thousands of servers and network components build an infrastructure serving 10 million requests per day. Dynamo provides a simplified interface based on get/put semantics, where objects are stored and retrieved with a unique identifier (key).</p> <p>b. Explain the Azure Dynamo architecture with a neat diagram.</p> <p>The architecture of the Dynamo system, shown in Figure 8.3, is composed of a collection of storage peers organized in a ring that shares the key space for a given application. The key space is partitioned among the storage peers, and the keys are replicated across the ring, avoiding adjacent peers. Each peer is configured with access to a local storage facility where original objects and replicas are stored.</p> <p>Each node provides facilities for distributing the updates among the rings and to detect failures and unreachable nodes.</p>  <p>FIGURE 8.3 Amazon Dynamo architecture.</p>	<p>3 7</p>	<p>CO3</p>	<p>L2</p>
<p>5</p>	<p>a. What is Data Intensive Computing?</p> <p>Data-intensive computing is concerned with production, manipulation, and analysis of large-scale data in the range of hundreds of megabytes (MB) to petabytes (PB) and beyond.</p> <p>Dataset is commonly used to identify a collection of information elements that is relevant to one or more applications. Datasets are often maintained in repositories, which are infrastructures supporting the storage, retrieval, and indexing of large amounts of information.</p>	<p>3 7</p>	<p>CO3</p>	<p>L3</p>

To facilitate classification and search, relevant bits of information, called **metadata**, are attached to datasets. Data-intensive computations occur in many application domains.

Computational science is one of the most popular ones. People conducting scientific simulations and experiments are often keen to produce, analyze, and process huge volumes of data. Hundreds of gigabytes of data are produced every second by telescopes mapping the sky; the collection of images of the sky easily reaches the scale of petabytes over a year.

Bioinformatics applications mine databases that may end up containing terabytes of data.

Earthquake simulators process a massive amount of data, which is produced as a result of recording the vibrations of the Earth across the entire globe.

b. Articulate the characteristics of Data Intensive Computing.

Characterizing data-intensive computations Challenges ahead

Historical perspective

- 1 The early age: high-speed wide-area networking
- 2 Data grids
- 3 Data clouds and “Big Data”
- 4 Databases and data-intensive computing

Characterizing data-intensive computations

Data-intensive applications deal with huge volumes of data, also exhibit compute-intensive properties. **Figure 8.1** identifies the domain of data-intensive computing in the two upper quadrants of the graph. Data-intensive applications handle datasets on the scale of multiple terabytes and petabytes.

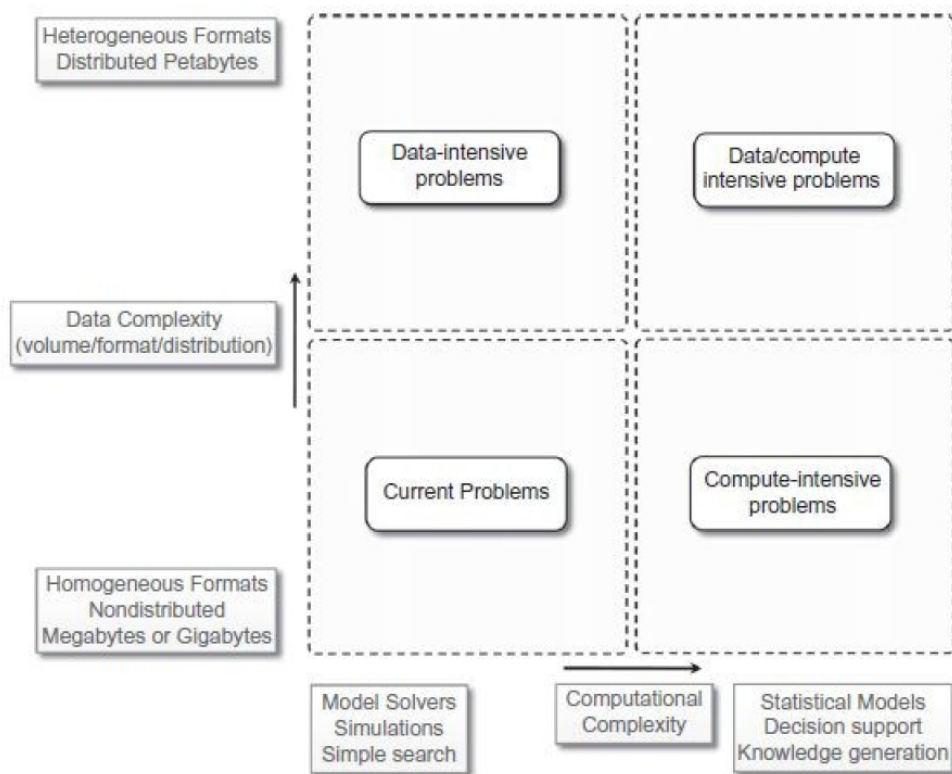


FIGURE 8.1

Data-intensive research issues.

6 a. What is the need for Google AppEngine?

3
7

CO2 L3

An Engine is a fully managed, serverless platform for developing and hosting web applications at scale. You can choose from several popular languages, libraries, and frameworks to develop your apps, and then let App Engine take care of provisioning servers and scaling your app instances based on demand. Google App Engine lets app developers build scalable web and mobile back ends in any programming language on a fully managed serverless platform.

b. With a neat diagram, sketch and explain the architecture of Google AppEngine platform.

Google AppEngine is a PaaS implementation

Distributed and scalable runtime environment that leverages Google's distributed infrastructure to scale out applications.

Architecture and core concepts

AppEngine is a platform for developing scalable applications accessible through the Web. **Figure 9.2.**

The platform is logically divided into four major components: infrastructure, the runtime environment, the underlying storage, and the set of scalable services.

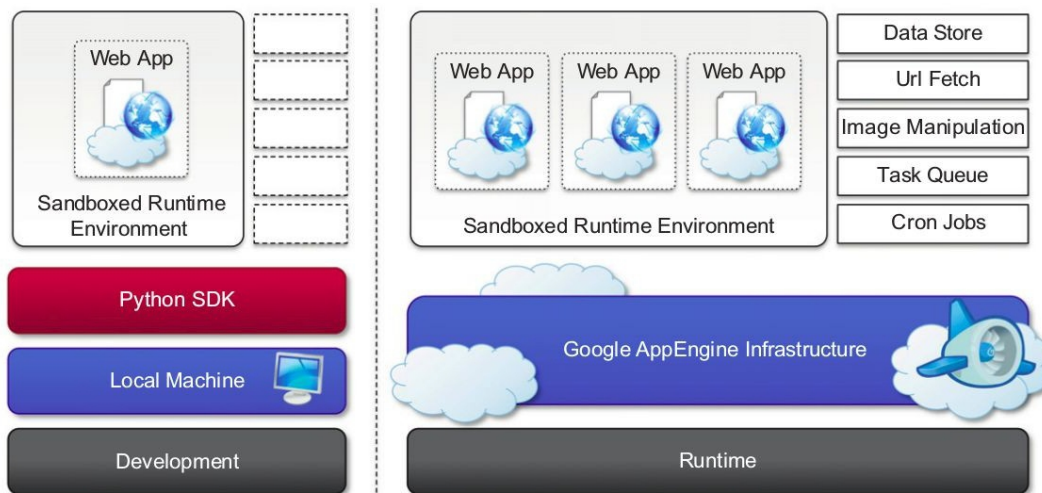


FIGURE 9.2

Google AppEngine platform architecture.

1 Infrastructure

AppEngine hosts Web applications, and its primary function is to serve users requests efficiently.

AppEngine's infrastructure takes advantage of many servers available within Google datacenters. For each HTTP request, AppEngine locates the servers hosting the application that processes the request, evaluates their load, and, if necessary, allocates additional resources or redirects the request to an existing server.

The infrastructure is also responsible for monitoring application performance and collecting statistics on which the billing is calculated.

2 Runtime environment

The runtime environment represents the execution context of applications hosted on AppEngine.

Sandboxing- One of the major responsibilities of the runtime environment is to provide the application environment with an isolated and protected context in

which it can execute without causing a threat to the server and without being influenced by other applications. In other words, it provides applications with a sandbox.

If an application tries to perform any operation that is considered potentially harmful, an exception is thrown and the execution is interrupted.

Supported runtimes- Currently, it is possible to develop AppEngine applications using three different languages and related technologies: Java, Python, and Go.

AppEngine currently supports Java 6, and developers can use the common tools for Web application development in Java, such as the Java Server Pages (JSP), and the applications interact with the environment by using the Java Servlet standard.

Support for Python is provided by an optimized Python 2.5.2 interpreter. As with Java, the runtime environment supports the Python standard library.

Developers can use a specific Python Web application framework, called webapp, simplifying the development of Web applications.

The Go runtime environment allows applications developed with the Go programming language to be hosted and executed in AppEngine. Currently the release of Go that is supported by AppEngine is r58.1. The SDK includes the compiler and the standard libraries for developing applications in Go and interfacing it with AppEngine services.

3 Storage

AppEngine provides various types of storage, which operate differently depending on the volatility of the data. **Static file servers-** Web applications are composed of dynamic and static data. Dynamic data are a result of the logic of the application and the interaction with the user. Static data often are mostly constituted of the components that define the graphical layout of the application or data files.

DataStore- DataStore is a service that allows developers to store semi-structured data. The service is designed to scale and optimized to quickly access data. DataStore can be considered as a large object database in which to store objects that can be retrieved by a specified key.

DataStore imposes less constraint on the regularity of the data but, at the same time, does not implement some of the features of the relational model.

The underlying infrastructure of DataStore is based on Bigtable, a redundant, distributed, and semistructured data store that organizes data in the form of tables.

DataStore provides high-level abstractions that simplify interaction with Bigtable. Developers define their data in terms of entity and properties, and these are persisted and maintained by the service into tables in Bigtable.

DataStore also provides facilities for creating indexes on data and to update data within the context of a transaction. Indexes are used to support and speed up queries. A query can return zero or more objects of the same kind or simply the corresponding keys.

4 Application services

Applications hosted on AppEngine take the most from the services made available through the runtime environment. These services simplify most of the common operations that are performed in Web applications **UrlFetch** - The sandbox environment does not allow applications to open arbitrary connections through sockets, but it does provide developers with the capability of retrieving a remote resource through HTTP/HTTPS by means of the UrlFetch service. Applications can make synchronous and asynchronous Web requests and integrate the resources obtained in this way into the normal request- handling cycle of the application.

UrlFetch is not only used to integrate meshes into a Web page but also to leverage remote Web services in accordance with the SOA reference model for distributed

applications.

MemCache- This is a distributed in-memory cache that is optimized for fast access and provides developers with a volatile store for the objects that are frequently accessed. The caching algorithm implemented by MemCache will automatically remove the objects that are rarely accessed. The use of MemCache can significantly reduce the access time to data; developers can structure their applications so that each object is first looked up into MemCache and if there is a miss, it will be retrieved from DataStore and put into the cache for future lookups.

Mail and instant messaging- AppEngine provides developers with the ability to send and receive mails through Mail. The service allows sending email on behalf of the application to specific user accounts. It is also possible to include several types of attachments and to target multiple recipients.

AppEngine provides also another way to communicate with the external world: the Extensible Messaging and Presence Protocol (XMPP). Any chat service that supports XMPP, such as Google Talk, can send and receive chat messages to and from the Web application, which is identified by its own address.

Account management- AppEngine simplifies account management by allowing developers to leverage Google account management by means of Google Accounts.

Using Google Accounts, Web applications can conveniently store profile settings in the form of key-value pairs, attach them to a given Google account, and quickly retrieve them once the user authenticates.

5 Compute services

AppEngine offers additional services such as Task Queues and Cron Jobs that simplify the execution of computations.

Task queues- A task is defined by a Web request to a given URL, and the queue invokes the request handler by passing the payload as part of the Web request to the handler. It is the responsibility of the request handler to perform the “task execution,” which is seen from the queue as a simple Web request.

Cron jobs- the required operation needs to be performed at a specific time of the day, which does not coincide with the time of the Web request. In this case, it is possible to schedule the required operation at the desired time by using the Cron Jobs service.