Answer Key

CMRIT
CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

Internal Assessment Test 3 – Set 3 – July 2022

Sub:	Cloud Computing and its Applications				Sub Code:	18CS643	Branch	ISE
Date:	11/07/2022	Duration:	90 min's	Max Marks:	50	Sem/Sec:	VI / A, B and C	OBE

	Answer any FIVE questions	MARKS	CO	RBT
1	Explain the Aneka Map Reduce programming model with a neat diagram.	10	CO2	L2

Aneka MapReduce programming

Aneka provides an implementation of the MapReduce abstractions introduced by Google and implemented by Hadoop.

Introducing the MapReduce programming model

1 Programming abstractions
2 Runtime support
3 Distributed file system support

Example application

1 Parsing Aneka logs
2 Mapper design and implementation
3 Reducer design and implementation
4 Driver program
5 Running the application

Introducing the MapReduce programming model

The MapReduce Programming Model defines the abstractions and runtime support for developing MapReduce applications on top of Aneka.
Figure 8.7 provides an overview of the infrastructure supporting MapReduce in Aneka.
The application instance is specialized, with components that identify the map and reduce functions to use.

These functions are expressed in terms of **Mapper and Reducer classes** that are extended from the Aneka MapReduce APIs.

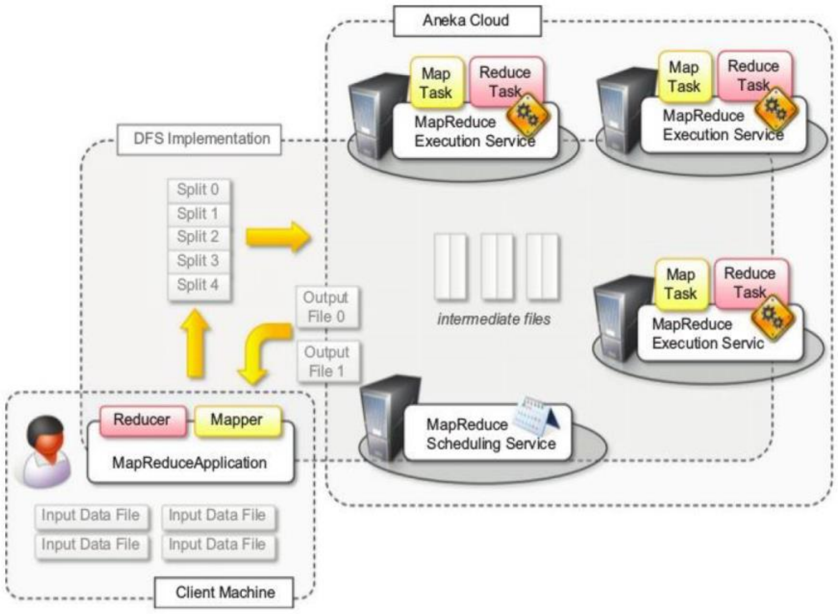


FIGURE 8.7

Aneka MapReduce infrastructure.

The runtime support is composed of **three** main elements:

1. **MapReduce Scheduling Service**, which plays the role of the master process in the Google and Hadoop implementation.
2. **MapReduce Execution Service**, which plays the role of the worker process in the Google and Hadoop implementation.
3. A specialized **distributed file system** that is used to move data files.

Client components, namely the MapReduce Application, are used to submit the execution of a MapReduce job, upload data files, and monitor it.

The management of data files is transparent: local data files are automatically uploaded to Aneka, and output files are automatically downloaded to the client machine if requested.

In the following sections, we introduce these major components and describe how they collaborate to execute MapReduce jobs.

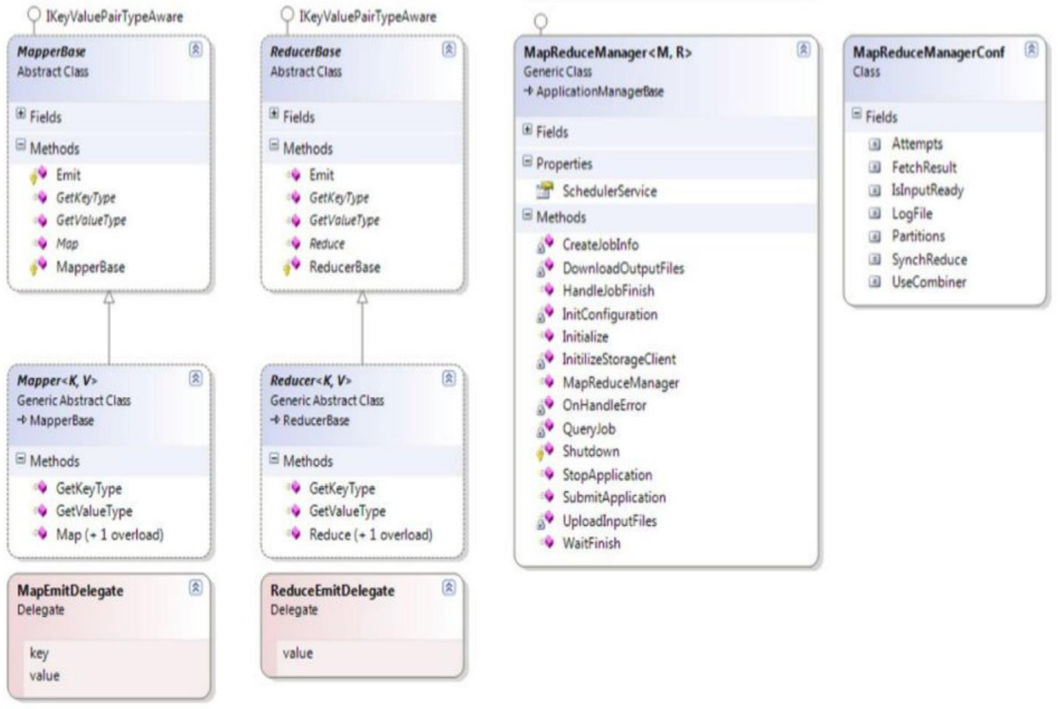
1 Programming abstractions

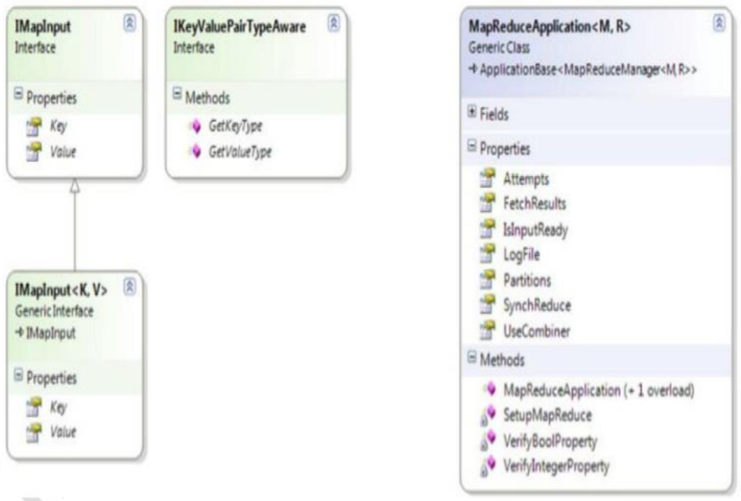
Aneka executes any piece of user code within distributed application.

The task creation is responsibility of the infrastructure once the user has defined the map and reduce functions.

Therefore, the Aneka MapReduce APIs provide developers with base classes for developing Mapper and Reducer types and use a specialized type of application class—Map Reduce Application — that supports needs of this programming model.

Figure 8.8 provides an overview of the client components defining the MapReduce programming model. Three classes are of interest for application development: Mapper<K,V>, Reducer<K,V>, and MapReduceApplication<M,R>. The other classes are internally used to implement all the functionalities

	 IKeyValuePairTypeAware MapperBase Abstract Class Fields Methods Emit GetKeyType GetValueType Map MapperBase IKeyValuePairTypeAware ReducerBase Abstract Class Fields Methods Emit GetKeyType GetValueType Reduce ReducerBase MapReduceManager<M, R> Generic Class → ApplicationManagerBase Fields Properties SchedulerService Methods CreateJobInfo DownloadOutputFiles HandleJobFinish InitConfiguration Initialize InitilizeStorageClient MapReduceManager OnHandleError QueryJob Shutdown StopApplication SubmitApplication UploadInputFiles WaitFinish MapReduceManagerConf Class Fields Attempts FetchResult IsInputReady LogFile Partitions SynchReduce UseCombiner Mapper<K, V> Generic Abstract Class → MapperBase Methods GetKeyType GetValueType Map (+ 1 overload) Reducer<K, V> Generic Abstract Class → ReducerBase Methods GetKeyType GetValueType Reduce (+ 1 overload) MapEmitDelegate Delegate key value ReduceEmitDelegate Delegate value **FIGURE 8.8** MapReduce Abstractions Object Model.			

	required by the model and expose simple interfaces that require minimum amounts of coding for implementing the map and reduce functions and controlling the job submission. Mapper<K,V> and Reducer<K,V> constitute the starting point of the application design and implementation. The submission and execution of MapReduce job is performed through class MapReduceApplication<M,R>, which provides the interface to Aneka Cloud to support MapReduce programming model. This class exposes two generic types: M and R. These two placeholders identify the specific types of Mapper<K,V> and Reducer<K,V> that will be used by the application. **Listing 8.1** shows in detail the definition of the Mapper<K,V> class and of the related types that developers should be aware of for implementing the map function. **Listing 8.2** shows the implementation of the Mapper<K,V> component for Word Counter sample. This sample counts frequency of words in a set of large text files. The text files are divided into lines, each of which will become the value component of a key-value pair, whereas the key will be represented by the offset in the file where the line begins. **Listing 8.3** shows the definition of Reducer<K,V> class. The implementation of a specific reducer requires specializing the generic class and overriding the abstract method: Reduce(IReduceInputEnumerator<V> input). **Listing 8.4** shows how to implement the reducer function for word-counter example. **Listing 8.5** shows the interface of MapReduceApplication<M,R>. **Listing 8.6** displays collection of methods that are of interest in this class for execution of MapReduce jobs. **Listing 8.7** shows how to create a MapReduce application for running the word-counter example defined by the previous WordCounterMapper and WordCounterReducer classes.  IMapInput Interface Properties Key Value IKeyValuePairTypeAware Interface Methods GetKeyType GetValueType IMapInput<K, V> Generic Interface → IMapInput Properties Key Value MapReduceApplication<M, R> Generic Class → ApplicationBase<MapReduceManager<M,R>> Fields Properties Attempts FetchResults IsInputReady LogFile Partitions SynchReduce UseCombiner Methods MapReduceApplication (+ 1 overload) SetupMapReduce VerifyBoolProperty VerifyIntegerProperty			
2	Explain SQL Azure architecture with a neat diagram. SQL Azure is a relational database service hosted on Windows Azure and built on the SQL Server technologies. The service extends the capabilities of SQL Server to the cloud and provides developers with a scalable, highly available, and fault-tolerant relational database. SQL Azure is accessible from either the Windows Azure Cloud or any other location that has access to the Azure Cloud. It is fully compatible with the interface exposed by SQL Server, so applications built for SQL Server can transparently migrate to SQL Azure. **Figure 9.4** shows the architecture of SQL Azure. Access to SQL Azure is based on the Tabular Data Stream (TDS) protocol, which is the communication protocol underlying all the different interfaces used by applications to connect to a SQL Server-based installation such as ODBC and ADO.NET.	10	CO3	L2

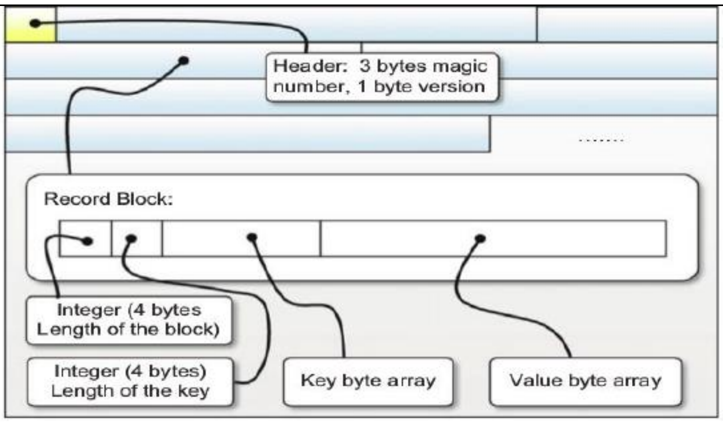

Header: 3 bytes magic number, 1 byte version
....
Record Block:
Integer (4 bytes Length of the block)
Integer (4 bytes) Length of the key
Key byte array
Value byte array

FIGURE 8.11
Aneka MapReduce data file format.

Developers have to sign up for a Windows Azure account in order to use SQL Azure. Once the account is activated, they can either use the Windows Azure Management Portal or the REST APIs to create servers and logins and to configure access to servers.

SQL Azure servers are abstractions that closely resemble physical SQL Servers: They have a fully qualified domain name under the database.windows.net (i.e., server-name.database.windows.net) domain name. This simplifies the management tasks and the interaction with SQL Azure from client applications.

Currently, two different editions are available: Web Edition and Business Edition. The former is suited for small Web applications and supports databases with a maximum size of 1 GB or 5 GB. The latter is suited for independent software vendors, line-of-business applications, and enterprise applications and supports databases with a maximum size from 10 GB to 50 GB, in increments of 10 GB.

Windows Azure platform appliance

The Windows Azure platform can also be deployed as an appliance on third-party data centers and constitutes the cloud infrastructure governing the physical servers of the datacenter. The Windows Azure Platform Appliance includes Windows Azure, SQL Azure, and Microsoft- specified configuration of network, storage, and server hardware. The appliance is a solution that targets governments and service providers who want to have their own cloud computing infrastructure.

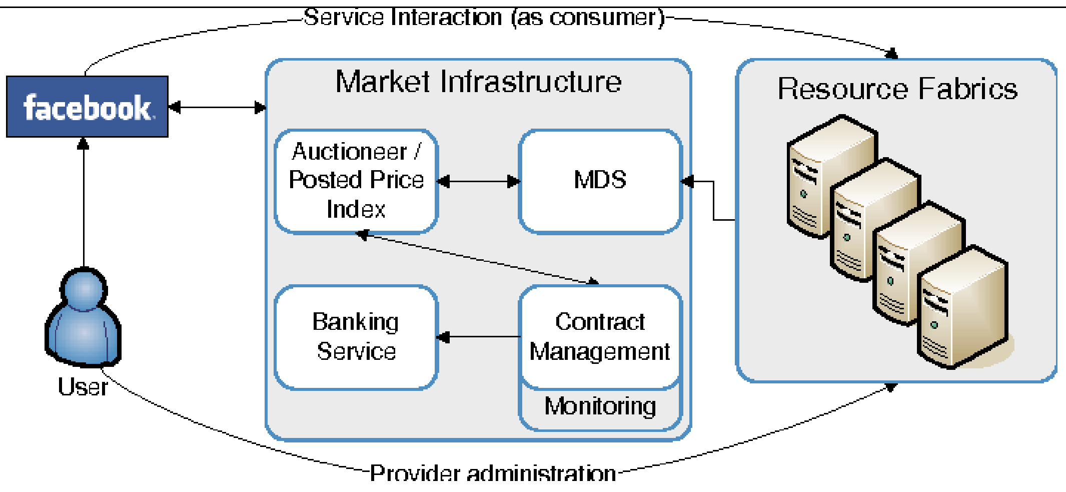
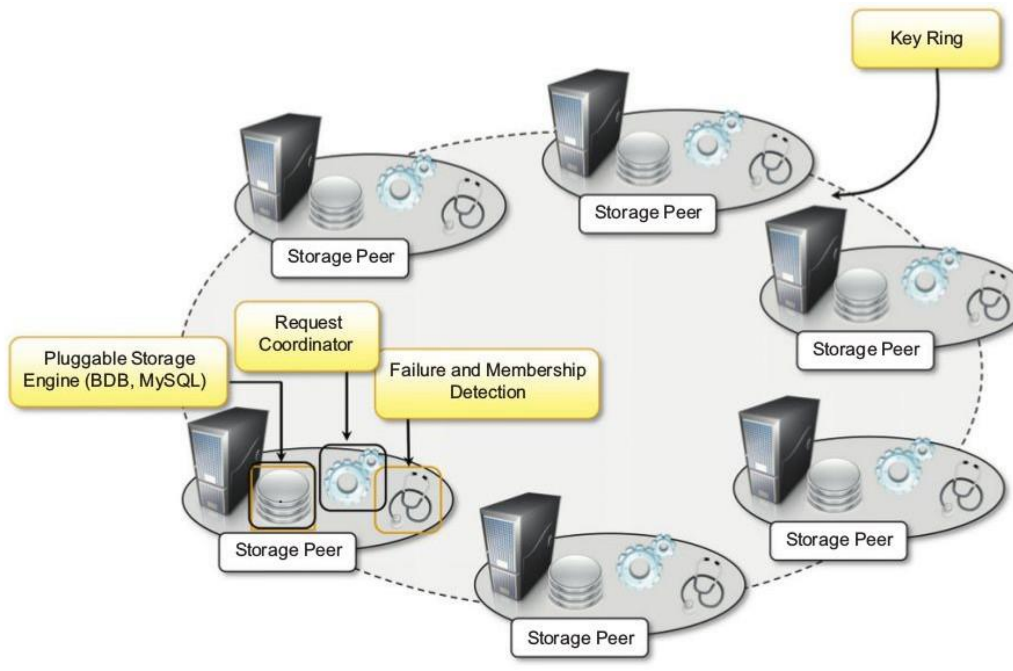
Observations

Windows Azure is Microsoft's solution for developing cloud computing applications. Azure is an implementation of the PaaS layer and provides the developer with a collection of services and scalable middleware hosted on Microsoft datacenters that address compute, storage, networking, and identity management needs of applications.

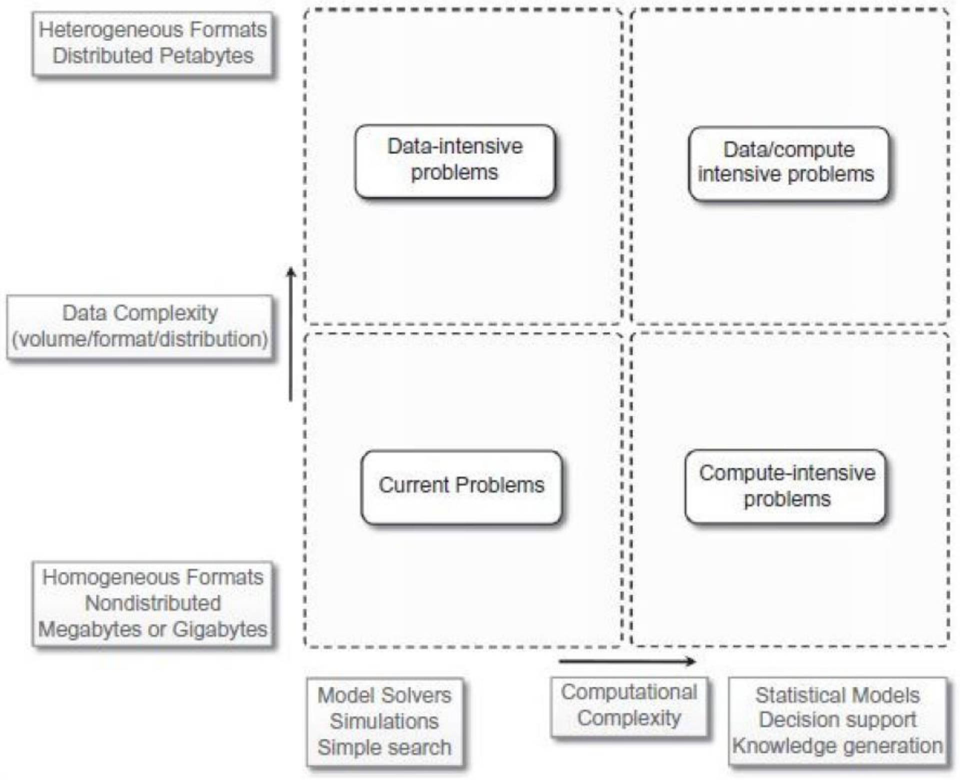
The core components of the platform are composed of compute services, storage services, and middleware. Compute services are based on the abstraction of roles, which identify a sandboxed environment where developers can build their distributed and scalable components.

SQL Azure is another important element of Windows Azure and provides support for relational data in the cloud. SQL Azure is an extension of the capabilities of SQL Server adapted for the cloud environment and designed for dynamic scaling.

3	Describe how Cloud Computing can be applied to social networking with a required diagram. Social networking applications have grown considerably in the last few years to become the most active sites on the Web. To sustain their traffic and serve millions of users seamlessly, services such as Twitter and Facebook have leveraged cloud computing technologies. The possibility of continuously adding capacity while systems are running is the most attractive feature for social networks, which constantly use their user base. **1 Facebook** Facebook is probably the most evident and interesting environment in social networking. With more than 800 million users, it has become one of the largest Websites in the world. To sustain this incredible growth, it has been fundamental that Facebook be capable of continuously adding capacity and developing new scalable technologies and software systems while maintaining high performance to ensure a smooth user experience. Currently, the social network is backed by two data centers that have been built and optimized to reduce costs and impact on the environment. On top of this highly efficient infrastructure, built and designed out of inexpensive hardware, a completely customized stack of opportunely modified and refined open-source technologies constitutes the back-end of the largest social network. Taken all together, these technologies constitute a powerful platform for developing cloud applications. This platform primarily supports Facebook itself and offers APIs to integrate third-party applications with Facebook's core infrastructure to deliver additional services such as social games and quizzes created by others. The reference stack serving Facebook is based on LAMP (Linux, Apache, MySQL, and PHP). This collection of technologies is accompanied by a collection of other services developed in-house. These services are developed in a variety of languages and implement specific functionalities such as search, news feeds, notifications, and others. While serving page requests, the social graph of the user is composed. The social graph identifies a collection of interlinked information that is of relevance for a given user. Most of the user data are served by querying a distributed cluster of MySQL instances, which mostly contain key-value pairs. These data are then cached for faster retrieval. The rest of the relevant information is then composed together using the services mentioned before. These services are located closer to the data and developed in languages that provide be These services are located closer to the data and developed in languages that provide better performance than PHP. The development of services is facilitated by a set of internally developed tools. One of the core elements is Thrift. This is a collection of abstractions (and language bindings) that allow cross language development. Thrift allows services developed in different languages to communicate and exchange data. Bindings for Thrift in different languages take care of data serialization and deserialization, communication, and client and server boilerplate code. This simplifies the work of the developers, who can quickly prototype services and leverage existing ones. Other relevant services and tools are Scribe, which aggregates streaming log feeds, and applications for alerting and monitoring.	10	CO3	L2

	 Service Interaction (as consumer) facebook Market Infrastructure Auctioneer / Posted Price Index MDS Banking Service Contract Management Monitoring Resource Fabrics User Provider administration			
4	Explain the Azure Dynamo architecture with a neat diagram. 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). 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. Each node provides facilities for distributing the updates among the rings and to detect failures and unreachable nodes.  Key Ring Storage Peer Storage Peer Storage Peer Request Coordinator Pluggable Storage Engine (BDB, MySQL) Failure and Membership Detection Storage Peer Storage Peer Storage Peer **FIGURE 8.3** Amazon Dynamo architecture.	10	CO3	L2

5	What is Data Intensive Computing? Articulate the characteristics of Data Intensive Computing. **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**. **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. 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. **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 dealS 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.	10	CO3	L3

	 Heterogeneous Formats Distributed Petabytes Data-intensive problems Data/compute intensive problems Data Complexity (volume/format/distribution) Current Problems Compute-intensive problems Homogeneous Formats Nondistributed Megabytes or Gigabytes Model Solvers Simulations Simple search Computational Complexity Statistical Models Decision support Knowledge generation **FIGURE 8.1** Data-intensive research issues.			
6	With a neat diagram, sketch and explain the Google AppEngine platform architecture. **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.	10	CO2	L3

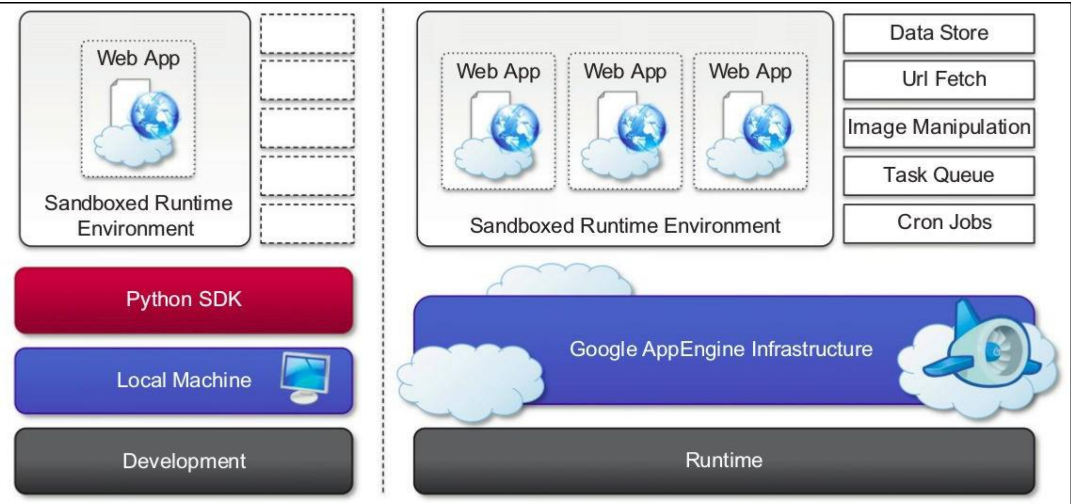


FIGURE 9.2

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.			