USN										

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

Internal Assessment Test 2 – May 2023 (Answer Key)

Sub:	Cloud Computing and its Applications					Sub Code:	18CS643	Branch	ISE
Date:	24/05/2023	Duration:	90 min's	Max Marks:	50	Sem/Sec:	VI / A and B		OBE

Answer any FIVE questions

		MARKS	CO	RBT
1	a. Elucidate the broad definition of Cloud Computing.	3 7	CO2	L2

A broad definition of the phenomenon could be as follows:
"Cloud computing is a utility- oriented and Internet-centric way of delivering IT services on demand. These services cover the entire computing stack: from the hardware infrastructure packaged as a set of virtual machines to software services such as development platforms and distributed applications."

b. Explain the Cloud Reference Model with a suitable diagram.

The Cloud Reference Model

Cloud computing supports any IT service that can be consumed as a utility and delivered through a network, most likely the Internet. Such characterization includes quite different aspects: infrastructure, development platforms, application and services.

Architecture

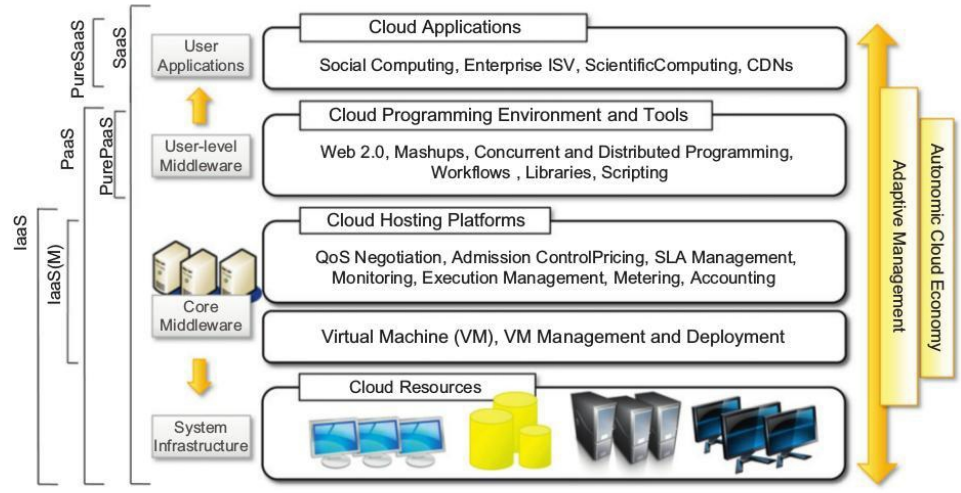


FIGURE 4.1
The cloud computing architecture.

It is possible to organize all the concrete realizations of cloud computing into a layered view covering the entire stack (see Figure 4.1), from hardware appliances to software systems. Cloud resources are harnessed to offer "computing horsepower" required for providing services. Cloud infrastructure can be heterogeneous in nature because a variety of resources, such as clusters and even networked PCs, can be used to build it.

The physical infrastructure is managed by the core middleware, the objectives of which are to provide an appropriate runtime environment for applications and to best utilize resources. At the bottom of the stack, virtualization technologies are used to guarantee runtime environment customization, application isolation, sandboxing, and

quality of service. Hardware virtualization is most commonly used at this level. Hypervisors manage the pool of resources and expose the distributed infrastructure as a collection of virtual machines. By using virtual machine technology it is possible to finely partition the hardware resources such as CPU and memory and to virtualize specific devices, thus meeting the requirements of users and applications. This solution is generally paired with storage and network virtualization strategies, which allow the infrastructure to be completely virtualized and controlled.

Infrastructure management is the key function of core middleware, which supports capabilities such as negotiation of the quality of service, admission control, execution management and monitoring, accounting, and billing.

The combination of cloud hosting platforms and resources is generally classified as an Infrastructure-as-a-Service (IaaS) solution. We can organize the different examples of IaaS into two categories: Some of them provide both the management layer and the physical infrastructure; others provide only the management layer (IaaS (M)).

In this second case, the management layer is often integrated with other IaaS solutions that provide physical infrastructure and adds value to them.

IaaS solutions are suitable for designing the system infrastructure but provide limited services to build applications. Such service is provided by cloud programming environments and tools, which form a new layer for offering users a development platform for applications.

The range of tools include Web-based interfaces, command-line tools, and frameworks for concurrent and distributed programming. In this scenario, users develop their applications specifically for the cloud by using the API exposed at the user-level middleware. For this reason, this approach is also known as Platform-as-a-Service (PaaS) because the service offered to the user is a development platform rather than an infrastructure.

The top layer of the reference model depicted in Figure 4.1 contains services delivered at the application level. These are mostly referred to as Software-as -a-Service (SaaS). In most cases these are Web-based applications that rely on the cloud to provide service to end users. The horsepower of the cloud provided by IaaS and PaaS solutions allows independent software vendors to deliver their application services over the Internet.

Table 4.1 summarizes the characteristics of the three major categories used to classify cloud computing solutions. In the following section, we briefly discuss these characteristics along with some references to practical implementations.

Table 4.1 Cloud Computing Services Classification

Category	Characteristics	Product Type	Vendors and Products
SaaS	Customers are provided with applications that are accessible anytime and from anywhere.	Web applications and services (Web 2.0)	SalesForce.com (CRM) Clarizen.com (project management) Google Apps
PaaS	Customers are provided with a platform for developing applications hosted in the cloud.	Programming APIs and frameworks Deployment systems	Google AppEngine Microsoft Azure Manjrasoft Aneka Data Synapse
IaaS/HaaS	Customers are provided with virtualized hardware and storage on top of which they can build their infrastructure.	Virtual machine management infrastructure Storage management Network management	Amazon EC2 and S3 GoGrid Nirvanix

2	Analyze and explain the open challenges facing the Cloud Computing paradigm with suitable examples. Cloud computing presents many challenges for industry and academia. There is a significant amount of work in academia focused on defining the challenges	10	CO2	L3

brought by this phenomenon.

In this section, we highlight the most important ones.
• Cloud definition
• Cloud interoperability and standards
• Scalability and fault tolerance
• Security, trust, and privacy
• Organizational aspects

Cloud definition

There have been several attempts made to define cloud computing and to provide a classification of all the services and technologies identified as such. NSIT characterizes cloud computing as on-demand self-service, broad network access, resource-pooling, rapid elasticity, and measured service; classifies services as SaaS, PaaS, and IaaS; and categorizes deployment models as public, private, community, and hybrid clouds.

Alternative taxonomies for cloud services. David Linthicum, founder of Blue Mountains Labs, provides a more detailed classification, which comprehends 10 different classes and better suits the vision of cloud computing within the enterprise.

These characterizations and taxonomies reflect what is meant by cloud computing at the present time, but being in its infancy the phenomenon is constantly evolving, and the same will happen to the attempts to capture the real nature of cloud computing.

Cloud interoperability and standards

To fully realize this goal, introducing standards and allowing interoperability between solutions offered by different vendors are objectives of fundamental importance. Vendor lock-in constitutes one of the major strategic barriers against the seamless adoption of cloud computing at all stages.

The presence of standards that are actually implemented and adopted in the cloud computing community could give room for interoperability and then lessen the risks resulting from vendor lock-in.

The first steps toward a standardization process have been made, and a few organizations, such as the Cloud Computing Interoperability Forum (CCIF), the Open Cloud Consortium, and the DMTF Cloud Standards Incubator, are leading the path.

Another interesting initiative is the Open Cloud Manifesto, which embodies the point of view of various stakeholders on the benefits of open standards in the field.

The Open Virtualization Format (OVF) is an attempt to provide a common format for storing the information and metadata describing a virtual machine image. Even though the OVF provides a full specification for packaging and distributing virtual machine images in completely platform-independent fashion, it is supported by few vendors that use it to import static virtual machine images.

Scalability and fault tolerance

The ability to scale on demand constitutes one of the most attractive features of cloud computing. Clouds allow scaling beyond the limits of the existing in-house IT resources, whether they are infrastructure (compute and storage) or applications services. To implement such a capability, the cloud middleware has to be designed with the principle of scalability along different

	dimensions in mind— for example, performance, size, and load. The cloud middleware manages a huge number of resource and users, which rely on the cloud to obtain the horsepower. In this scenario, the ability to tolerate failure becomes fundamental, sometimes even more important than providing an extremely efficient and optimized system. Hence, the challenge in this case is designing highly scalable and fault-tolerant systems that areeasy to manage and at the same time provide competitive performance. **Security, trust, and privacy** Security, trust, and privacy issues are major obstacles for massive adoption of cloud computing. The traditional cryptographic technologies are used to prevent data tampering and access to sensi-tive information. The massive use of virtualization technologies exposes the existing system to new threats, which previously were not considered applicable. Information can be stored within a cloud storage facility using the most advanced technology in cryptography to protect data and then be considered safe from any attempt to access it without the required permissions. The lack of control over data and processes also poses severe problems for the trust we give to the cloud service provider and the level of privacy we want to have for our data. **Organizational aspects** More precisely, storage, compute power, network infrastructure, and applications are delivered as metered services over the Internet. This introduces a billing model that is new within typical enterprise IT departments, which requires a certain level of cultural and organizational process maturity. In particular, the following questions have to be considered: • What is the new role of the IT department in an enterprise that completely or significantly relies on the cloud? • How will the compliance department perform its activity when there is a considerable lack of control over application workflows? • What are the implications (political, legal, etc.) for organizations that lose control over some aspects of their services? • What will be the perception of the end users of such services? From an organizational point of view, the lack of control over the management of data and processes poses not only security threats but also new problems that previously did not exist.			
3	a. What is Aneka Cloud? •Aneka is a software platform for developing cloud computing applications. •Aneka is a pure PaaS solution for cloud computing. •Aneka is a cloud middleware product that can be deployed on a heterogeneous set of resources: Like: a network of computers, a multi core server, data centers, virtual cloud infrastructures, or a mixture of all •The framework provides both middleware for managing and scaling distributed applications and an extensible set of APIs for developing them. b. Briefly explain the different types of application services offered by Aneka Cloud. **Application services** Application Services manage the execution of applications and	3 7	CO3	L2

	constitute a layer that differentiates according to the specific programming model used for developing distributed applications on top of Aneka. Two types of services are: **1. The Scheduling Service** Scheduling Services are in charge of planning the execution of distributed applications on top of Aneka and governing the allocation of jobs composing an application to nodes. Common tasks that are performed by the scheduling component are the following: o Job to node mapping o Rescheduling of failed jobs o Job status monitoring o Application status monitoring **2. The Execution Service** Execution Services control the execution of single jobs that compose applications. They are in charge of setting up the runtime environment hosting the execution of jobs. Some of the common operations that apply across all the range of supported models are: • Unpacking the jobs received from the scheduler • Retrieval of input files required for job execution • Sandboxed execution of jobs • Submission of output files at the end of execution • Execution failure management (i.e., capturing sufficient contextual information useful to identify the nature of the failure) • Performance monitoring • Packing jobs and sending them back to the scheduler			
4	Explain the domain and functional decomposition models of parallelizing tasks in Cloud Computing with suitable examples **Domain decomposition** Domain decomposition is the process of identifying patterns of functionally repetitive, but independent, computation on data. This is the most common type of decomposition in the case of throughput computing, and it relates to the identification of repetitive calculations required for solving a problem. The master-slave model is a quite common organization for these scenarios: • The system is divided into two major code segments. • One code segment contains the decomposition and coordination logic. • Another code segment contains the repetitive computation to perform. • A master thread executes the first code segment. • As a result of the master thread execution, as many slave threads as needed are created to execute the repetitive computation. • The collection of the results from each of the slave threads and an eventual composition of the final result are performed by the master thread. **Embarrassingly parallel** problems constitute the easiest case for parallelization because there is no need to synchronize different threads that do not share any data. Embarrassingly parallel problems are quite common, they are based on the strong assumption that at each of the iterations of the decomposition method, it is possible to isolate an independent unit of work. This is what makes it possible to obtain a high computing throughput. If the values of all the iterations are	10	CO2	L2

dependent on some of the values obtained in the previous iterations, the problem is said to be **inherently sequential**. Figure 6.3 provides a schematic representation of the decomposition of embarrassingly parallel and inherently sequential problems.

$$C_{ij} = \sum_{k=0}^{n-1} A_{ik} B_{kj}$$

The matrix product computes each element of the resulting matrix as a linear combination of the corresponding row and column of the first and second input matrices, respectively. The formula that applies for each of the resulting matrix elements is the following:

Two conditions hold in order to perform a matrix product:

- Input matrices must contain values of a comparable nature for which the scalar product is defined.
- The number of columns in the first matrix must match the number of rows of the second matrix.

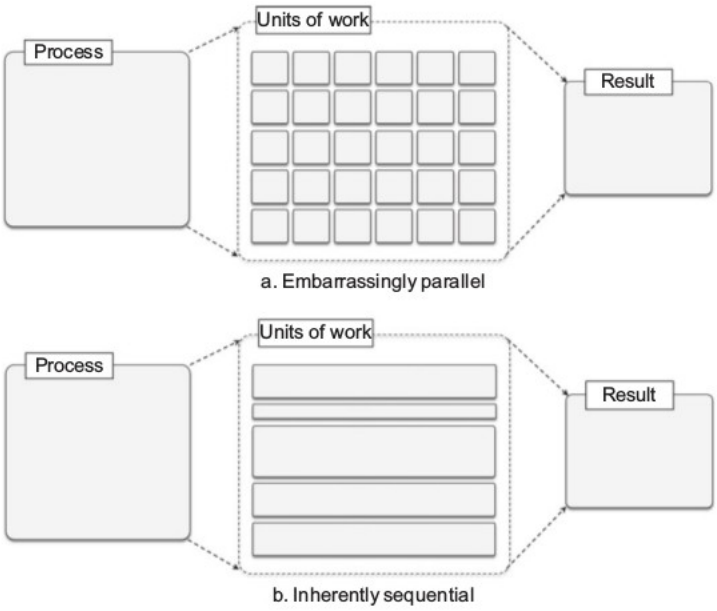


FIGURE 6.3

Domain decomposition techniques.

The problem is embarrassingly parallel, and we can logically organize the multithreaded program in the following steps:

- Define a function that performs the computation of the single element of the resulting matrix by implementing the previous equation.
- Create a double for loop (the first index iterates over the rows of the first matrix and the second over the columns of the second matrix) that spawns a thread to compute the elements of the resulting matrix.
- Join all the threads for completion, and compose the resulting matrix.

Functional decomposition

Functional decomposition is the process of identifying functionally distinct but independent computations. The focus here is on the type of computation rather than on the data manipulated by the computation.

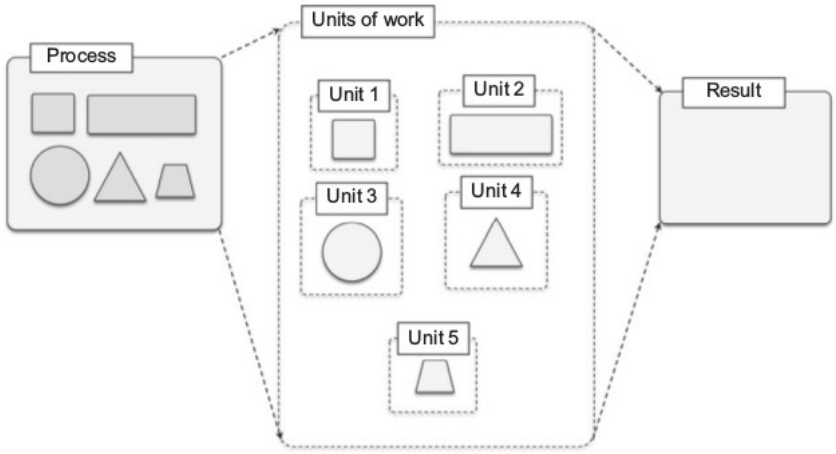


FIGURE 6.5

Functional decomposition.

This kind of decomposition is less common and does not lead to the creation of a large number of threads, since the different computations that are performed by a single program are limited. Functional decomposition leads to a natural decomposition of the problem in separate units of work. Figure 6.5 provides a pictorial view of how decomposition operates and allows parallelization.

The problems that are subject to functional decomposition can also require a composition phase in which the outcomes of each of the independent units of work are composed together.

In the following, we show a very simple example of how a mathematical problem can be parallelized using functional decomposition. Suppose, for example, that we need to calculate the value of the following function for a given value of x:

$$f(x) = \sin(x) + \cos(x) + \tan(x)$$

Once the value of x has been set, the three different operations can be performed independently of each other. This is an example of functional decomposition because the entire problem can be separated into three distinct operations.

5	Analyze and elucidate the limitations of Aneka thread model compared to the normal thread model for running distributed applications. The pictures of Aneka thread model and the related normal thread model are given below:	10	CO3	L3

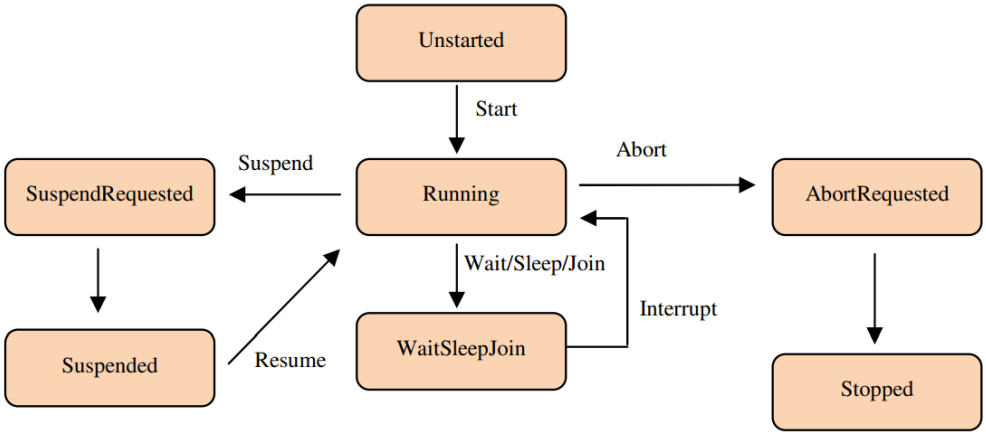


Figure 4: The life cycle of local threads in .Net

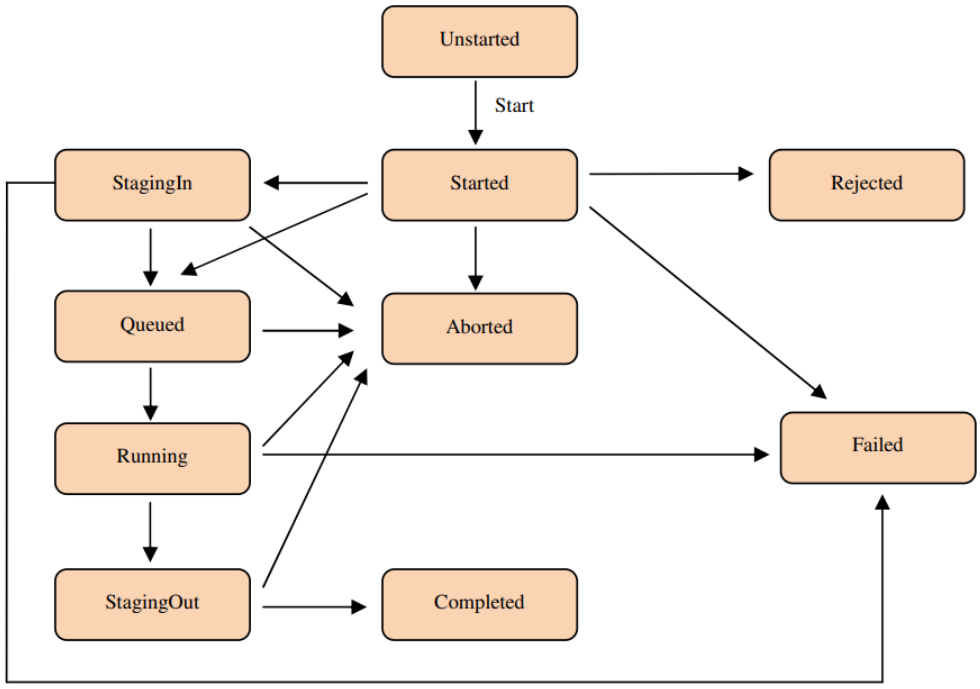


Figure 5: The life cycle of AnekaThreads

Limitations of Aneka Thread model

- Even though a distributed facility can dramatically increase the degree of parallelism of applications, its use comes with a cost in term of application design and performance.
- For example, since the different units of work are not executing within the same process space but on different nodes both the code and the data needs to be moved to a different execution context.
- the same happens for results that need to be collected remotely and brought back to the master process.
- Moreover, if there is any communication among the different workers it is necessary to redesign the communication model eventually by leveraging the APIs provided by the middleware if any.

	• In other words, the transition from a single process multi-threaded execution to a distributed execution is not transparent and application redesign and re-implementation are often required. • The amount of effort required to convert an application often depends on the facilities offered by the middleware managing the distributed infrastructure. • Aneka, as a middleware for managing clusters, Grids, and Clouds, provides developers with advanced capabilities for implementing distributed applications. • In particular, it takes traditional thread programming a step further. It lets you write multi-threaded applications in the traditional way, with the added twist that each of these threads can now be executed outside the parent process and on a separate machine. • In reality, these "threads" are independent processes executing on different nodes, and do not share memory or other resources, but they allow you to write applications using the same thread constructs for concurrency and synchronization as with traditional threads. • Aneka threads, as they are called, let you easily port existing multi-threaded compute intensive applications to distributed versions that can run faster by utilizing multiple machines simultaneously, with a minimum conversion effort.			
6	a. What are the differences between embarrassingly parallel and parameter sweep applications? **Embarrassingly parallel applications** Embarrassingly parallel applications constitute the most simple and intuitive category of distributed applications. The tasks might be of the same type or of different types, and they do not need to communicate among themselves. This category of applications is supported by the majority of the frameworks for distributed computing. Since tasks do not need to communicate, there is a lot of freedom regarding the way they are scheduled. Tasks can be executed in any order, and there is no specific requirement for tasks to be executed at the same time. Scheduling these applications is simplified and concerned with the optimal mapping of tasks to available resources. Frameworks and tools supporting embarrassingly parallel applications are the Globus Toolkit, BOINC, and Aneka. There are several problems: image and video rendering, evolutionary optimization, and model forecasting. In image and video rendering the task is represented by the rendering of a pixel or a frame, respectively. For evolutionary optimization meta heuristics, a task is identified by a single run of the algorithm with a given parameter set. The same applies to model forecasting applications. In general, scientific applications constitute a considerable source of embarrassingly parallel applications. **Parameter sweep applications** Parameter sweep applications are a specific class of embarrassingly parallel applications for which the tasks are identical in their nature and differ only by the specific parameters used to execute. Parameter sweep applications are identified by a template task and a set of parameters. The template task defines the operations that will be performed on the remote node for the execution of tasks. The parameter set identifies the combination of variables whose assignments	4 6	CO2	L2

specialize the template task into a specific instance. Any distributed computing framework that provides support for embarrassingly parallel applications can also support the execution of parameter sweep applications.

The only difference is that the tasks that will be executed are generated by iterating over all the possible and admissible combinations of parameters. Nimrod/G is natively designed to support the execution of parameter sweep applications, and Aneka provides client-based tools for visually composing a template task, defining parameters, and iterating over all the possible combinations. A plethora of applications fall into this category. Scientific computing domain: evolutionary optimization algorithms, weather-forecasting models, computational fluid dynamics applications, Monte Carlo methods. For example, in the case of evolutionary algorithms it is possible to identify the domain of the applications as a combination of the relevant parameters.

For genetic algorithms these might be the number of individuals of the population used by the optimizer and the number of generations for which to run the optimizer.

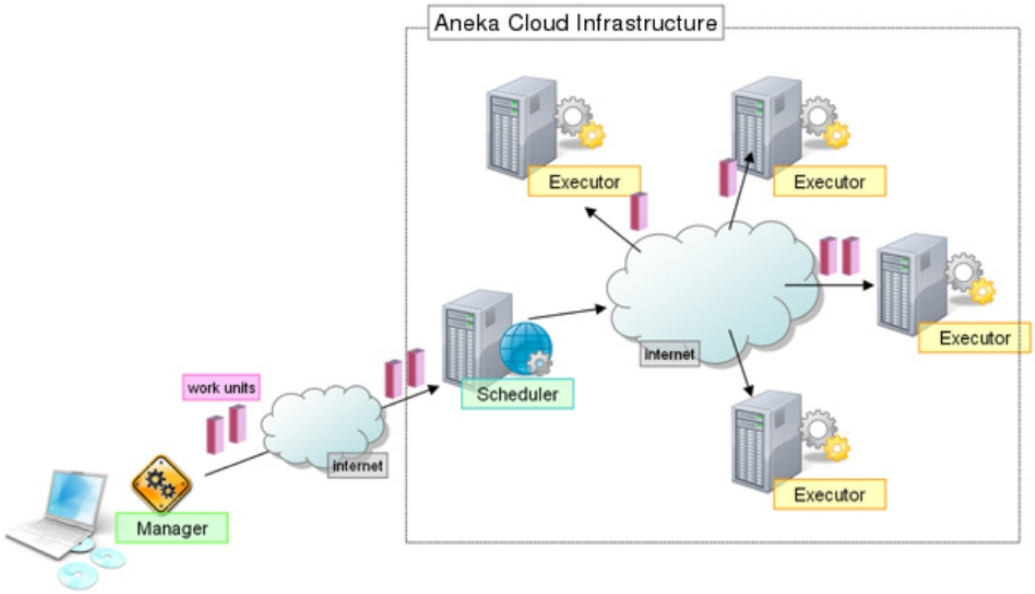
b. Explain the Aneka task programming model with a suitable diagram.

Aneka Task Model

Aneka allows different kind of applications to be executed on the same grid infrastructure. In order to support such flexibility it provides different abstractions through which it is possible to implement distributed applications. These abstractions map to different execution models. Currently Aneka supports three different execution models:

- Task Execution Model
- Thread Execution Model
- MapReduce Execution Model

Each execution model is composed by four different elements: the WorkUnit, the Scheduler, the Executor, and the Manager. The WorkUnit defines the granularity of the model; in other words, it defines the smallest computational unit that is directly handled by the Aneka infrastructure. Within Aneka, a collection of related work units define an application. The Scheduler is responsible for organizing the execution of work units composing the applications, dispatching them to different nodes, getting back the results, and providing them to the end user. The Executor is responsible for actually executing one or more work units, while the Manager is the client component which interacts with the Aneka system to start an application and collects the results. A view of the system is given in Figure 1.



Hence, for the Task Model there will be a specific WorkUnit called AnekaTask, a Task Scheduler, a Task Executor, and a Task Manager. In order to develop an application for Aneka the user does not have to know all these components; Aneka handles a lot of the work by itself without the user contribution. Only few things users are required to know:

- how to define AnekaTask instances specific to the application that is being defined;
- how to create a AnekaApplication and use it for task submission;
- how to control the AnekaApplication and collect the results.

This holds not only for the Task Model but for all execution models supported by the Aneka.