CHAPTER 1

INTRODUCTION

1.1 CLOUD COMPUTING

Clouds are a large pool of usable and accessible resources such as hardware, software and application (Zhong & Yuan 2013). Cloud computing is the delivery of these computing resources as services over the Internet to customers through virtualization of hardware and software (Buyya et al 2009). These resources can be dynamically reconfigured to adjust to a variable user load allowing for an optimum resource utilization (Randles 2010). Cloud computing is based on a market oriented business model where users are charged for using cloud services in everyday life on a pay-as-you-go basis like conventional utilities; for example water, electricity, telephony etc. (Verma & Kaushal 2011).

Cloud service providers offer these services based on customized Service Level Agreements which defines user’s required Quality of Service parameters (Tsai et al 2011). Cloud computing reduces investment on various resources like hardware, software and allow resources to be leased and released and also reduces maintenance and operating costs.

1.2 CHARACTERISTICS OF CLOUD

Cloud computing is a model for enabling ubiquitous, convenient and on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications and services that
can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics (Shrivastava & Yadav 2013) as shown in Figure 1.1.

**Figure 1.1 Characteristics of Cloud**

- **On-demand self-service**: A consumer can unilaterally provision computing capabilities such as server time and network storage, as needed automatically without requiring human interaction with each service’s provider.

- **Broad network access**: Capabilities are available over the network and accessed through standard mechanisms that promote usage by heterogeneous thin or thick client platforms like mobile phones, laptops and PDAs.

- **Resource pooling**: The provider’s computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact
location of the provided resources but may be able to specify location at a higher level of abstraction.

- **Rapid elasticity**: Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and any quantity can be purchased at any time. Elasticity is one of the key differentiable characteristics which separate modern clouds from other previous utility computing forms.

- **Measured service**: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service like storage, processing, bandwidth, and active user accounts. Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

These characteristics allows flexible delivery of integrated content, applications, and services to any device, anywhere, anytime, in a seamlessly scalable model, using and paying for only the resources needed. It allows organizations to free themselves from having to procure and allocate expensive hardware, software, and networking resources, or employ large teams to manage and support infrastructure.

### 1.3 COMPUTING AS A SERVICE

Cloud computing services (Xu et al 2009) can be delivered to customers in three different ways. They are Software as a Service, Platform as a Service and Infrastructure as a Service as shown in Figure 1.2
1.3.1 Software as a Service (SaaS)

The capability provided to the consumer is the ability to use the provider’s applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser, email, or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings.
1.3.2 **Platform as a Service (PaaS)**

The capability provided to the consumer is being able to deploy onto the cloud infrastructure, consumer-created or acquired applications created using programming Languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

1.3.3 **Infrastructure as a Service (IaaS)**

The consumer is provided the capability to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components such as host firewalls.

1.4 **DEPLOYMENT MODELS**

A cloud deployment model (Lin & Huang 2012) represents a specific type of cloud environment, primarily distinguished by ownership, size, and access. The four common cloud deployment models are Private cloud, Public cloud, Hybrid cloud and Community cloud as shown in Figure 1.3.
1.4.1 Private Cloud

The cloud infrastructure is provisioned for exclusive use by a single organization which can comprise of multiple consumers, for example, multiple business units. It may be owned, managed, and operated by the organization, a third party, or some combination of these. It may exist within the premises of the organization or could be located elsewhere. All the services are deployed behind the company's firewall and are subject to existing mechanisms for physical, electronic and procedural security in the facility. Accordingly, private cloud offers a high degree of control and data security, making it a popular option for enterprises uncomfortable with storing information on someone else's infrastructure. Private clouds are more expensive but also more secure when compared to public clouds.

1.4.2 Public Cloud

The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the
premises of the cloud provider. Public Cloud customers benefit from economies of scale, because infrastructure costs are spread across all users, allowing each individual client to operate on a low-cost, pay-as-you-go model. Another advantage of public cloud infrastructures is that they are typically larger in scale than an in-house enterprise cloud, which provides clients with seamless, on-demand scalability. These clouds offer the greatest level of efficiency in shared resources; however, they are also more vulnerable than private clouds.

1.4.3 Hybrid Cloud

The cloud infrastructure is a composition of two or more distinct cloud infrastructures: private, community, or public. Each of the infrastructures remains unique entities, but is bound together by standardized or proprietary technology that enables data and application portability.

1.4.4 Community Cloud

The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns, such as, security requirements, policy and/or compliance considerations. It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.

1.5 VIRTUALIZATION TECHNOLOGY

Virtualization (Uddin et al 2012) is the faithful reproduction of an entire architecture in software which provides the illusion of a real machine to all software running above it. Prior to the development of modern virtualization technologies, many high-end servers lay wholly underutilized as
each server was responsible for the management of a specific piece of software or application component. It was common for companies to have separate servers for different applications and functions, such as a print server, mail server and domain server. This led to enormous under-utilization of the available resources even though it was considered best practice at that time. The reason why this was done was to isolate the possibility of system wide failures due to a single fault or bug proliferating throughout the server. Thus if the print server was to fail, the mail server would not be affected.

It facilitates cloud computing as it allows larger physical servers to be partitioned into multiple smaller virtual machines. These machines are insulated from each other, with respect to fault tolerance, security and operating system heterogeneity. Figure 1.4 shows working principle of virtualization technique in cloud systems.

![Configuration of Virtualization](image-url)
A cloud data center consists of a centralized repository of physical host servers. A hypervisor, also called a virtual machine manager, is a program that allows multiple operating systems to share a single hardware host. The hypervisor controls the host processor and allocate resources to each operating system, such that guest operating systems, called as virtual machines (VMs) cannot disrupt each other.

With the advancement of virtualization technology (Averitt et al 2007) multiple applications could reside in apparent isolation on the same physical resource. Secure from each other, a known vulnerability in one virtual machine or a malware infection in another could not impact or spread to all the other virtual machines. Now a single large physical server could host all applications and business functions securely on the same host machine thereby, reducing the server overhead significantly.

Cloud service providers manage the underlying infrastructure and aim to use the resources efficiently while gaining maximum profit (Yogita & Bhonsle 2012). Task scheduling (Lovejit & Sarbjeet 2013) is therefore a challenging issue as resource usage has to be maximized without affecting the services provided by the cloud (Kaur & Kinger 2014).

1.6 WORKFLOW SCHEDULING

Applications hosted and executed using clouds are often composed from a set of tasks which form a workflow. Workflow processing requires tasks to be executed based on their control and data dependencies (Abrishami et al 2013). Workflow applications (Kumar & Verma 2012) can be commonly represented or modelled as a Directed Acyclic Graph (DAG), defined by a tuple G (T, E), where T is the set of n tasks \{t_1, t_2, ......, t_n\}, and E is a set of e edges, represent the dependencies. Each t_i \in T, represents a task in the application and each edge (t_i, ...... , t_j) \in E represents a precedence constraint,
such that the execution of \( t_j \in T \) cannot be started before \( t_i \in T \) finishes its execution. If \((t_i, t_j) \in T\), then \( t_i \) is the parent of \( t_j \), and \( t_j \) is the child of \( t_i \). This is illustrated with an example shown in Figure 1.5.

![Figure 1.5 Sample Workflow Application](image)

Figure 1.5 shows the dependencies among the different tasks, numbered 0 to 9. Task 0 is an entry task as it has no parents. Task 9 is the exit task as it has no children. Task 0 is the parent of child tasks 1, 2, 3 and 4.

Workflow scheduling is the assignment of the application tasks to the different VMs for execution (Wu et al 2013). The scheduling process maps and manages the execution of the inter-dependent tasks on the distributed resources. It allocates suitable resources to workflow tasks such that the execution can be completed to satisfy the SLAs agreed with the users. Workflow scheduling is a well-known (Elzeki et al 2012) NP-complete problem and many heuristic and meta-heuristics methods have been proposed for distributed systems like grids.

Once the workflow has been defined, any task for which its parent tasks have completed can then be executed. Thus a workflow scheduling can distribute tasks amongst the available resources as and when they become available, harnessing the potential of parallel hardware in a process intensive way. However scheduling algorithms must be cognizant of the fact that
placing a task on the next available resource can be a naive strategy. This is particularly prevalent in data intensive workflows where scheduling child operations on the same resources as their parents reduces data transfer operations dramatically. In addition during the execution process a schedule that was valid in the beginning may become invalid due to resources going offline or increased network latencies/congestion.

QoS is the collective effort of services performance, which determines the degree of the satisfaction of a user for the services and can be expressed in terms of qualitative measures like makespan (completion time), cost, availability, reliability, priority, load balance and deadline (Yee & Peng 2012).

Ke Liu et al (2010) have proposed a Compromised-Time-Cost (CTC) algorithm to achieve lower cost while meeting the user-designated deadline. However, this was found to increase the overall job completion time. FIFO scheduling was proposed by evaluating the entire group of tasks in the job queue (Yujiage et al 2010). It, however, failed to optimize the scheduling to adapt to changing loads. A heuristic based scheduling on Particle Swarm Optimization (PSO) (Pandey et al 2010) lead to minimization of total cost of execution but did not cater to optimizing the completion time. A priority based job scheduling (Shamsollah & Othman 2012) model was proposed for assigning priority to workflow tasks and makespan, but, cost was not considered.

In Ant Colony Optimization, a single VM which has better performance was used for executing all tasks. This leads to load imbalance as there would be heavy load on one VM while others would be idle. The algorithm was found to be not scalable and the makespan of the tasks was also large. The PCM algorithm (Abrishami & Naghibzadeh 2012) was proposed to
execute the workflows with minimum time within the user specified deadline, but does not address the task completion deadline constraint. Bi-Objective Priority based Particle Swarm Optimization (Amandeep 2014) schedule workflows to minimize execution cost while meeting a user defined deadline. However, minimization of completion time was not considered.

Meta heuristic techniques discussed above, have considered only one or two QoS parameter (Rak & Venticinque 2009). However, SLAs require multiple QoS criteria to be met and efficient workflow methodology are required to meet these requirements. Hence, workflow scheduling remains a key challenge for cloud computing applications.

1.7 MOTIVATION AND OBJECTIVES

For, a utility service like cloud computing, pricing is fixed by the service provider. The cost is generally dependent on the level of Quality of Service offered and service providers may charge higher prices for higher QoS. Users may not always need to complete workflows earlier than they require and instead, may prefer to use cheaper services with lower QoS that are sufficient to meet their requirements. Workflow scheduling has to be carried out so as to meet the QoS requirements agreed with users and documented in SLAs (Emeakaroha et al 2011).

Elasticity is an important characteristic of cloud computing. VM resources are assigned to user jobs/tasks as and when needed. This dissertation focuses on the development of an efficient workflow scheduling methodology that optimizes resource utilization in Cloud computing. This methodology can be used to determine whether existing VMs can process the user jobs and meet the SLAs or whether additional VMs would be required (Chitra et al 2013).
The main objectives of the research are:

- To develop an efficient workflow scheduling to optimizes resource utilization in Cloud computing.
- To schedule workflows on basis of multiple QoS constraints: completion time, cost, and deadline.
- To achieve task load balance among the multiple VMs available in the cloud.

Execution time is defined to be taken to execute a task on a particular VM. Completion time is the time taken to complete the set of tasks that comprise the job or application. Deadline is defined as the maximum time that can be taken to complete the application tasks. Load balancing looks at dividing the task evenly on different machine for optimum provisioning of cloud resources.

1.8 CONTRIBUTION SUMMARY

The contributions of the research work are briefly summarized as follows:

- This thesis work has developed a workflow scheduling methodology, WFS-ABC, which considers multiple QoS parameters, namely, completion time, cost, and deadline. A priority assignment was developed to arrive at a specific order for the execution of the tasks while taking into consideration the cost and execution time. Prior work has primarily carried out workflow scheduling using only one parameter, such as, either completion time or cost or deadline.
In WFS-ABC methodology, scheduling is carried out by taking into consideration the precedence relationship among the various tasks and aims to complete all tasks within their permitted deadline. The Scheduling is done for the tasks to enable efficient provisioning of the cloud resources. PCM the prior work on deadline has used partial critical path method with user defined deadline. Other QoS criteria were not considered along with deadline.

WFS-ABC methodology in addition to QoS parameters mentioned above also provides for efficient balance of task load among the different VMs. Prior work ACO-LB has proposed the assignment of all tasks to high performance VMs, leading to load imbalance and idle VMs.

The research work has proposed the use of ABC algorithm for optimization of the cloud resources. Fitness value is computed to determine the best solution. Prior work have used GA, ACO and PSO for workflow scheduling. ABC algorithm provides optimum solutions to complex problems and has significant advantages when compared with the other optimization techniques. The comparative study is given in Chapter 2.

WFS-ABC methodology has been implemented and tested using synthetic data sets. The performance has also been compared with prior work, such as, GA, ACO and PSO. Results show that the performance of WFS-ABC is superior to the other methods. The performance results have been discussed in detail in Chapters 3, 4, 5 and 6.
1.9 THESIS STRUCTURE

A brief outline of the various chapters of the thesis is as follows:

**Chapter 2** discusses the literature review regarding workflow scheduling with Quality of Services of execution time, cost and deadline. And also discusses the load balancing of various Virtual Machines in the cloud environment.

In **Chapter 3**, the proposed model WFS-ABC: Workflow Scheduling on Completion Time (makespan) has been explained. The methodology of arriving efficient scheduling of tasks while minimizing the overall job completion time of the tasks has been discussed. A sample DAG has been presented and the performance of the proposed work has been analyzed using GA, ACO and PSO.

**Chapter 4** describes the Workflow Scheduling on completion time with cost has been explained. A Priority Assignment (order of execution) was developed to arrive at a specific order for the execution of the tasks while taking into consideration the cost and execution time. A sample DAG has been presented and the performance of the proposed work has been analyzed using GA, ACO and PSO.

**Chapter 5** describes the Workflow Scheduling on deadline with completion time and cost has been explained. Using Latest finish of all tasks, scheduling of tasks on different VMs is explained. A sample DAG has been presented and the performance of the proposed work has been analyzed using GA, ACO and PSO.

**Chapter 6** describes the Workflow Scheduling on load balancing of VMs with completion time, cost and deadline has been explained. It also shows that proper VM scheduling can reduce the number of migration that
will increase the overall performance of the system. A sample DAG has been presented and the performance of the proposed work has been analyzed using GA, ACO and PSO.

Chapter 7 discusses the critical analysis of the work and summarizes the results and contributions of the research work. The suggestions for future work have been presented.