CHAPTER 2
LITERATURE REVIEW

2.1 INTRODUCTION

The study ensures the necessity and managing of resource allocation on cloud environment with impression of distributed system as model. The primary objective of this effort is to minimize the application completion time, at the expense of redundant resource usage. The selection of a particular scheduling algorithm depends upon various factors like the parameter to be optimized such as cost or time, quality of service to be provided and various aspects of task.

In this chapter, various works available in the literature with regard to fault tolerance and identification techniques, meta and heuristic scheduling methods and clustering techniques are reviewed.

2.2 FAULT TOLERANCE AND IDENTIFICATION TECHNIQUES USED

CC infrastructure encompasses many design challenges. Unreliability is one of the important design challenges in cloud platform. It also provide variety of services for variety of clients. Malik et al (2012) presents a model for the reliability assessment of the cloud infrastructures considering virtual machines as computing nodes. This reliability assessment mechanism helps to do the scheduling on cloud infrastructure. It also provide fault tolerance based on the reliability values acquired during reliability assessment. The model is based on Figure 2.1. Every compute instance such as virtual machine in PaaS or physical node in IaaS have reliability values associated with them.
The system assesses the reliability for different types of applications. It follows different mechanism to assess the reliability based on the application type such as general applications and real time applications. For real time applications, time based reliability assessment is followed. All the algorithms are more convergent towards failures. In the case of a failure, reliability decreases at a faster rate.

Figure 2.1 Layered Cloud Computing Design

Ganga & Karthik (2013) states that fault tolerance is a configuration that prevents a computer or network device from failing in the event of unexpected problem or error such as hardware failure, link failure, unauthorized access, variations in the configuration of different systems and system running out of memory or disk space. The integration of fault tolerance measures with scheduling gains much importance. Scientific workflows using
distributed heterogeneous resources in cloud interface are often hard to program.

Zheng (2010) represents that map reduce has been used in Google, Yahoo, Facebook etc., even for their production tasks. However, according to a recent study, a single failure on a Hadoop task could cause a 50% increase in completion time. Amazon Elastic Map Reduce has been provided to help users perform data-intensive tasks for their applications. These applications may have high fault tolerance and/or tight service level agreement requirements. However, map reduce fault tolerance in the cloud is more challenging as topology control and (data) rack locality are not possible.

Oliner et al (2006) state, that cooperative checkpointing increases the performance and robustness of a system by allowing checkpoints requested by applications to be dynamic at runtime. A robust system must be more than merely resilient to failures. It must be adaptable and flexible to new and evolving challenges. A simulation-based experimental analysis using both probabilistic and harvested failure distributions reveals that cooperative checkpointing enables an application to make progress under a wide variety of failure distributions that periodic checkpointing lacks to handle. Cooperative checkpointing can be easily implemented on top of existing application-initiated. Checkpointing mechanisms is used to enhance other reliability techniques like QoS guarantees and fault-aware task scheduling. The simulations also support a number of theoretical predictions related to cooperative checkpointing, including the non-competitiveness of periodic checkpointing.

As the cloud environment grows the scalability and complexity place the critical role to ensure the stability, availability and reliability of the systems. Due to the dynamic environment, removal of system component, upgradation, online repairs, workload are the main reasons to induce failure and faults in CC.
The reliability of such system can be satisfied if the proactive measures are taken to solve against the possible failures emerging in cloud system. For example Google has reported a 20% revenue loss, when an experiment caused by an additional delay of 500 ms in the response time as stated by Greenberg et al (2009).

To achieve, reliability as a counter measure for failure and faults, the cloud service provider use various mechanism to implement fault tolerance at the system level. Fault tolerance is a vital issue in CC platforms and application. Fault tolerance enables a system to continue operation at reduced level rather than complete failure. Fault tolerance also plays an important role is maintaining SLA as well as QoS. The SLAs and QoS requirements significantly increase the complexity and unpredicted problem in cloud environment.

### 2.3 EVOLUTIONARY SCHEDULING ALGORITHM

Scheduling is a major problem of cloud. Cloud scheduling aims to satisfy the user demands like response time, processing cost and also to attain the objective of resource provider like efficient resource utilization, profit and minimum makespan while scheduling the tasks to the resource. In the dynamic cloud environment, the number of resources and tasks to be scheduled is usually variable. This kind of feature makes the scheduling a complex one. The scheduling of tasks in a cloud environment has been proved to be a Non-deterministic Polynomial-hard (NP-hard) problem. NP-hard problems are solved using heuristic algorithms. Heuristic scheduling algorithms such as Min-min, Genetic Algorithm (GA) and Simulated Annealing (SA) algorithms had achieved good results as reported by Jiayin et al (2010).
Elnikety et al (2004) explained that workflows are invoked via web requests. The workflows are part of a web application that spans multiple resources in the grid. They use the workflow execution time and cost to perform workflow scheduling. In addition to the scheduling, admission control also is used for the outstanding workflow queue to keep the system load under a desired threshold. The workflows share the underlying resources that have different execution length and resource requirements. Therefore, some scheduling is required for each workflow in order to sustain a reasonable system performance. Admission control is implemented upon request, arrival and a workflow is only invoked if the overall system load does not exceed a certain threshold, to avoid the system crash. Once admitted, the workflows are ordered in terms of their size, hence named as Shortest Task First (STF) scheduling heuristic. Experiments reveal that the workflow completion time is reduced by a factor of fourteen when compared to scheduling and admission control.

Uurgaonkar & Chandra (2005) deal with the similar problem by monitoring a network of queues containing the workflows across the grid. A multi-tier grid configuration with workload queue is considered. A continuous monitoring of the queue is used to establish the resource requirements and the additional resources are provisioned when necessary. Although both of the previous studies are geared towards E-commerce web sites and an enterprise grid which is a host to many large scale service oriented web applications.

Arash et al (2012) proposed a reliable scheduling algorithm in Cloud environment. It divides major task into sub tasks. In order to balance the tasks the request and acknowledge time are calculated separately. The scheduling of each task is done by calculating the request and acknowledges time in the form of a shared task, so that the efficiency of the system is increased.
Open Nebula is an open source project aiming at managing datacenter’s virtual infrastructure to build IaaS clouds [Opennebula]. It was established by complutense university of Madrid in 2005, and released its first software in 2008. It supports private Cloud creation based on local virtual infrastructure in datacenters, with the managing capabilities of user, virtual network, multi-tier services and physical infrastructures. It also supports combination of the local resources and remote commercial cloud to build hybrid clouds. A single or multiple clouds supplements a local computing capacity to it. In addition, it can be used as interfaces to turn local infrastructure into a public cloud.

A heuristic task scheduling algorithm called BAlance-Reduce (BAR) proposed by Jiahui et al (2011), considers an initialize task allocation at first, and then the task completion time. The task completion time can be reduced gradually by tuning the initial task allocation. By considering the network state, BAR can adjust data locality dynamically. In this system, a set of independent tasks on a homogeneous platform with $m$ tasks and $n$ servers is considered, where each task processes an input block on a server. A task is not completed until all tasks are finished. Cloud computer clusters workload to design an allocation strategy that minimizes the makespan time of tasks. BAR called Balance-Reduce is a data locality driven task scheduling algorithm, which finds a good solution in time $O(\max \{m+n, n \log n\} \cdot m)$. BAR is split into two phases, balance and reduce. In balance phase, a balanced total allocation is produced where all tasks are allocated to their preferred servers evenly. In reduce phase, it generates a sequence of total allocations, and reduce the make span iteratively.

Workload - Aware Task Scheduling (WATS) scheme is detailed by Maria et al (2009). It adopts history- based task allocation and preference-based task stealing. The history-based task allocation is based on a near-optimal, static
task allocation using the historical statistics collected during the execution of a parallel application. The preference-based task stealing, steals tasks based on a preference list. In the history-based task allocation the workloads are dynamically adjusted, if the task allocation is less optimal due to approximation.

In cloud computing, three different modes of renting the computing resources are available, such as, Advance Reservation (AR), best-effort and immediate from a cloud provider. To overcome the problems involved in resource utilization, AR and best-effort can be combined. It is assumed that few tasks submitted in the cloud system are in the AR mode, while the rest of the tasks are in the best-effort mode. The tasks in AR mode have higher priorities and able to preempt the executions of the best-effort tasks. Two algorithms for the task scheduling are proposed. They are Adaptive List Scheduling (ALS) and Adaptive Min-Min Scheduling (AMMS). Once a task is submitted to a scheduler, it will be partitioned into tasks in the form of Directed Acyclic Graph (DAG). Both ALS and AMMS include a static task scheduling for resource allocation. Then the scheduler will repeatedly re-evaluate the remaining static allocation with a pre-defined frequency, based on the latest information of task execution. To generate the static allocation two greedy algorithms, Cloud List Scheduling (CLS) and Cloud Min-Min Scheduling (CMMS) are used. The actual completion time of a task may be different from the estimated, due to the resource contention within the cloud. An online adaptive scheduling procedure is proposed to adjust the resource allocation dynamically based on the latest information.

Cost of each task differs depending on user task. Scheduling of user tasks in cloud is not the same as in traditional scheduling methods. Improved cost-based scheduling algorithm as explicated by Selvarani & Sudha (2010) is used for making efficient mapping of tasks to the available resources in cloud.
This scheduling algorithm measures both resource cost and computation performance. It improves the computation communication ratio by grouping the user tasks according to a particular cloud resource's processing capability and sent the grouped tasks to the resource.

2.4 WORKFLOW SCHEDULING ALGORITHM

Priority Impact Scheduling Algorithm (PISA) proposed by Hu Wu et al (2012), states that user priority is to be taken into account. The difference among users priority may be based on the fee paid. First, a prototype algorithm based on the workflow priority is used to classify the workflow, where the priority is set by the users. The priority may be based on the fee that user could pay for the workflow. Service provider may classify the users as free users, Very Important Person (VIP). Higher priority workflow can access the resource faster whereas the low priority cannot. A counter is set initially to zero and after each round of scheduling it is increment by one.

Workflow applications are the applications which require various sub-tasks to be executed in a particular fashion to complete the whole task. These tasks have parent child relationship. The parent task needs to be executed before its child task. Workflow scheduling mechanism is supposed to preserve dependency constraints implied by their nature and structure. Resources are allocated to various sub-tasks of the original task by keeping into account these constraints. Some algorithm have been found to optimize cost, time, some focuses on reliability, availability, energy efficiency, load balancing and combination of these parameters as described by Lovejit & Sarbjeet (2013).

Genez et al (2012) state that CC is used to avoid maintenance costs and upfront investment, while providing elasticity to the available computational power in a pay-per-use basis. Customers can make use of the cloud as a software (SaaS), platform (PaaS) or infrastructure (IaaS) provider.
When one customer utilizes an environment provided by a SaaS cloud, the details about the computational infrastructure where the requests are being processed is unknown. Therefore, such infrastructure can be composed of computational resources from a datacenter owned by the SaaS or its resources can be leased from a cloud infrastructure provider. This work presents an Integer Linear Program (ILP) formulation for the problem of scheduling SaaS customer's workflows into multiple IaaS providers where SLA exists at two levels. In addition, it presents heuristics to solve the relaxed version of the presented ILP. Simulation result show that the proposed ILP is able to find low-cost solutions for short deadlines, while the proposed heuristics are effective when deadlines are larger.

Huifang et al (2014) stated a new model for service provisioning. In this model, one of the most challenging problems is workflow scheduling, i.e., the problem of satisfying users' QoS while minimizing the execution cost of cloud workflow. A workflow scheduling algorithm called Control Structure Reduction algorithm (CSR) is proposed in this model. In CSR, the workflows represented by DAG can be converted into an equivalent sequence control structure by means of mergers and reduction. By using time float distribution algorithm, total time float is allocated to each task based on critical tasks, eventually to enlarge the cost optimization times of all tasks.

2.5 RESOURCE ALLOCATION STRATEGIES

Resource management policies for cloud can be grouped into five classes. They are admission control, capacity allocation, load balancing, energy optimization and QoS guarantees. The drawback of admission control policy is proposed by Gupta & Harchal (2009). The explicit goal of an admission control policy is to prevent the system from accepting workload in violation of high-level system policies. For example, a system may not accept additional workload which would prevent it from completing work already in progress.
Limiting the workload requires some knowledge of the global state of the system.

Spawn is the first implementation of an auction-based management system for distributed computing resources. Waldspurger et al (1992) aims to utilize the idle Central Processing unit (CPU) time in a network of workstations. Spawn employs periodical, single-good, single-sided, sealed-bid, second-price auctions. The buyers are end-users and the seller is the owner of the workstation. The auctioneer is the Spawn system running on the seller’s workstation. The trading procedure is a buyer finds a seller and proposes the CPU time of the workstation. The seller determines the winner and the winning buyer runs the program on the seller’s workstation. It is noticed that the auctions are distributed i.e. there are as many auctions as there are sellers in the network. The buyer needs to find an appropriate seller using nearest-neighbor connections provided. The spawn’s scheme does not fit the enterprise cloud environment because it prevents the system from scaling beyond a local network and prevents the buyers from bidding on a combination of multiple resources from different sellers.

Nisan et al (1998) extend the scope of spawn’s network of workstations idea from local network to global network. It aims to be an infrastructure for globally distributed computation over the whole Internet. POPCORN employs a periodical, single-good, single- or double-sided, sealed-bid, second-price auction. The buyers are Java programs written using POPCORN framework, the sellers are people using Java-enabled browsers, the auctioneer is an independent market service on the Internet, and the good is a right to run a Java applet computelet.

The trading procedure followed are a programmer writes a program using POPCORN framework and runs it on a local computer as a buyer, the buyer connects to the market and bids for execution of computelets, a seller
visits a web page where a POPCORN applet is embedded, the applet connects to the market and asks for execution of a computelet, the market determines the winners and the winning buyer spawns computelets and sends them to the winning sellers so that they execute the Computelets on their browsers and return the results back to the local computer as described in Regev & Noam (2000). The POPCORN scheme overcomes the scalability problem of spawn. However, it still does not fit the enterprise cloud environment because it does not support multiple kinds of resources and combination of them.

G-commerce compares two different market models, commodities market and auctions for resource allocation in grid computing environment. It on the one hand, employs a commodities market with price adjustment scheme for multiple interrelated goods and on the other hand employs periodical, single-good, single-sided, sealed-bid, second-price auctions. The buyers are the user agents and the sellers are the owner agents, the market is an independent service and the goods are CPU slots and disk capacities. The trading procedure followed are the buyers bid for CPU and disk, the sellers ask for CPU or disk that can be provided, the market determines the winners either by the commodities market or by the auctions and the winning buyers execute their tasks on the winning sellers as stated by Wolski et al (2001).

In this auction model, there are separate auctioneers for each goods and thus the buyers have an exposure risk. For example, if a buyer wins a CPU but not a disk, it should wait for the next chance to bid on another disk while paying for the CPU. Because of this shortcoming the authors concluded that the commodities market model is more suitable for grids than the auction model. However, this comparison is unfair because the auction model employed is not a combinatorial auction but a traditional single-good auction. Wolski et al (2004) briefed additionally that, the commodities market is typically unable to
deal with arbitrary kinds of goods. Therefore, the G-commerce scheme does not directly fit the enterprise cloud environment.

Nimrod/G was a negotiation-based Grid scheduler extended from Nimrod built on the top of Globus Toolkit as reported by Foster & Kesselman (1997). It aimed to help researchers run a parameter survey program on heterogeneous resources while meeting the deadline and budget constraint. Nimrod/G employed a commodity market model with the parameters of time and cost. The buyer was an end-user, the seller was a resource provider i.e. a computing node, the market was Nimrod/G components and the good is the resource.

The trading procedure was that the buyer requests the scheduler via the parametric engine to arrange a resource that satisfies deadline and cost, the scheduler negotiates with the resource providers and selects one that met the deadline at a minimum cost and the scheduler dispatches the user task to the selected resource as explained by Buyya et al (2000). Nimrod/G is specially designed for massively parallel applications on the scientific Grid environment, and thus it did not fit the enterprise cloud environment where several kinds of applications need to run in conjunction with others as compared by Abramson et al (1997).

The Resource Allocation Strategies (RAS) must be based on a disciplined approach, rather than ad hoc methods. It is outlined in Figure 2.2. Basic mechanisms for the implementation of resource management policies are provided. Capacity allocation means to allocate resources for individual instances. An instance is an activation of a service. Locating resources subject to multiple global optimization constraints, requires a search in a very large search space when the state of individual systems changes rapidly. Load balancing and energy optimization are correlated in order to affect the cost of providing the services. It can be done locally. Kusic et al (2008) stated that
global load balancing and energy optimization policies encounter the same difficulties as in the capacity allocation. Quality of service is probably the most challenging aspect of resource management at the same time, the most critical for the future of cloud computing.

Basic mechanisms for the implementation of resource management policies are, control theory, machine learning, utility-based, economic models. Control theory uses the feedback to guarantee system stability and to predict transient behavior. It can be used only to predict local, rather than global behavior. The major advantage of machine learning techniques that it does not need a performance model of the system. This technique could be applied for coordination of several autonomic system managers. Utility based approaches require a performance model and a mechanism to correlate user-level performance with cost. Economic models are nothing but auction models, cost-utility models or macroeconomic models that are an intriguing alternative and have been the focus of research in recent years.

OCEAN provides a market-based framework that enhances various middlewares for cluster and grid computing. It aims to be an infrastructure for high performance computing environment where the resources are traded as commodities. Padala et al (2003) state that OCEAN employs a hybrid model
Figure 2.2 Hierarchal representation of RAS
of tendering and bargaining in a Peer To Peer (P2P) network. The buyers and the sellers are represented by OCEAN nodes, the matchmakers are implemented within OCEAN nodes and the good is resources specified by CPU, memory, disk, network, auxiliary hardware, software, database etc. The trading procedures are a buyer propagates the trade proposal to nearby sellers, a seller also propagates the trade proposal to nearby buyers, the buyer starts negotiations by offering contracts to the potential sellers, the seller accepts or rejects the contract and return it back to the buyer, the buyer examines the returned contracts and signs the preferable ones, the seller checks the signed contract and counter-sign it and the buyer starts using the resources. OCEAN’s scheme does not fit the enterprise cloud environment because of its Peer to Peer (P2P) based structure. It does not provide a single marketplace where worldwide information on resource supply and demand are exchanged all together and its negotiation based procedure does not guarantees neither an economically efficient allocation of resources nor an fair competition among the participants.

Bellagio was the first combinatorial auction-based marketplace for distributed computing resources designed by AuYoung et al (2004). It aims to be a resource discovery and allocation system for distributed computing infrastructures such as PlanetLab by Peterson et al (2003). The buyers are end-users, the seller is a computer site, the auctioneer is centralized Bellagio server, and the good is a resource discovered by SWORD. The trading procedures are the buyer discovers the sellers satisfied by requirements, the buyer bids for a bundle of resources, the auctioneer determines the winners once an hour and the winning buyer uses the resources. Oppenheimer et al (2004) identified Bellagio scheme is well designed so that the buyer can express substitutability as well as complementarities between different kinds of resources. However, it does not fit the enterprise cloud environment because the sellers are out of
competition and the buyers cannot bid for a workflow consisting of multiple resources used on different timeslots.

Mirage was another combinatorial auction-based resource management system for Sensor Net test beds as elucidated by Chun et al (2005). It had been deployed in a real-world testbed and brought practical knowledge about the user’s behavior. Mirage employs periodical, combinatorial, single-sided, sealed-bid auctions. The buyers are end-users, the seller is an agent on behalf of the sensors, the auctioneer is Mirage daemon running on front-end web server and the good is access to the sensor. The trading procedure is the buyer bids for sensors specified by abstract requirements, the auctioneer translates the requirements into concrete set of sensors using resource discovery service and the auctioneer determines the winners and the winning buyers get access to the sensors. Mirage’s scheme is much similar to Bellagio’s and it does not fit the enterprise cloud environment.

Tycoon by Lai et al (2005) explained that a market-based distributed resource allocation system based on proportional share, where the resources are allocated in proportion to the amount of money the user spends. It aims to allow users to differentiate the value of their tasks in a cluster computing environment. Tycoon employs auction share algorithm to allocate resources instantly and reliably. The buyers are user agents, the sellers are hosts in the cluster, the auctioneer is a process running on the host and the good is a right to use resources like CPU cycles. Note that there are multiple independent auctioneers on each host. The trading procedure is the seller registers himself to the service location service, the buyers bid on an auctioneer for the resource, the auctioneer determines the quota on the resource for the buyers, and the buyers use the resource. Tycoon’s scheme does not fit the enterprise cloud environment because its distributed single-good auction model causes an exposure risk for the buyers who want a bundle of multiple resources.
CATNETS compares the decentralized Catallactic approach with the centralized auction-based approach for resource allocation in Grid computing environment demonstrated by Eymann et al (2005). They have concluded that the decentralized approach fits to the Grid environment in terms of scalability, hereafter only the centralized approach is focused as it is the main concern of this thesis. The centralized approach employs periodical, combinatorial, double-sided, sealed-bid, K-pricing auctions. CATNETS divides the trading into two layers: a service market and a resource market. In the service market, the buyers are complex service agents on behalf of the end-users, the sellers are basic service agents, and the goods are services like PDF generation service.

In the resource market, the buyers are Basic Service Agents, the sellers are resource service agents on behalf of the owners, and the goods are resources like CPU / memory / storage / etc. In both markets idea of Schnizler et al (2005), the auctioneer is an independent entity. The trading procedure is a resource service agent asks for resource as seller in the resource market, a basic service agent asks for service as seller in the service market, a complex service agent bids for a bundle of services as a buyer in the service market, the auctioneer of the service market determines the winners. The winning basic service agent bids for a bundle of resources as a buyer in the resource market. The auctioneer of the resource market determines the winners. The winning Basic Service Agent uses the resources to provide his service to the winning complex service agent and the winning the complex service agent uses the service. Unlike former literatures, CATNETS’ scheme is much likely to fit the cloud computing environment as its resource / basic / complex services respectively correspond to the IaaS / PaaS / SaaS layers in the cloud as stated by Veit et al (2006 & 2007). Nonetheless, the CATNETS’ combinatorial auction model lacks the ability to deal with a workflow-oriented application which consists of multiple services running not at the same time.
SCDA by Tan & Gurd (2007) proposed an iterative combinatorial exchange for resource allocation in grid computing environments. It emulates a combinatorial auction by repeating a single-good auctions. It aims to eliminate unnecessary volatility of the market price observed in conventional continuous double auctions. SCDA employs continuous, single-good, double-sided, sealed-bid, K-pricing auctions. The buyers are user agents, the seller is an owner agent, the auctioneer is an independent agent and the good is an arbitrary kind of computing resource. The trading procedure is that the buyers bid for a certain amount of resource that they wish to use, the seller asks for available resource, in turn the auctioneer determines the winners and the winning buyer uses the resource of the winning seller. SCDA’s scheme does not fit the enterprise cloud environment because the buyers still have an exposure risk.

2.6 CLUSTERING TECHNIQUES

2.6.1 Task Clustering

Task clustering is nothing but grouping of small tasks together as one executable unit eliminating the data movement. Task clustering groups task so that the intermediate files produced by each task in the group remains in the same computing node. At the same time other tasks in the same group can now access the file locally. This scheme reduces the need to transfer the intermediate output if the tasks in the group were scheduled to different computing nodes. Clustering also eliminates the overhead of running small tasks.

Singh et al (2005) explored approaches for restructuring the workflows to reduce the dependencies in the workflow graph. The independent task at the same level is grouped into clusters. The task clustering does not imply that the tasks in a group is scheduled to one processor or executed sequentially. It shows workflow performance using clustering with centralized
and distributed task submission. In the centralized submission, the whole workflow is submitted and executed using a single submit host. In order to increase the dispatch rate of tasks for execution, their distributed task submission strategy has a central manager, multiple submit hosts and worker nodes. The workflow is restructured with multiple clusters at each level. The number of clusters at each level is equal to the number of submit hosts in the pool. The schedulers on the submit hosts then try to find suitable nodes for the submitted tasks.

Pandey et al (2009) used task clustering to schedule data intensive tasks for a medical application workflow. The tasks are clustered based on their execution time, data transfer and level. If tasks having high deviation and value of average execution time, they were executed without clustering. Tasks with lower deviation and less execution time were clustered together. It shows that clustering tasks for data intensive application workflows has better makespan than scheduling the workflow without clustering. Hence it is identified that the decrease in file transfers between tasks are in the same cluster.

Gang Scheduling is an efficient technique for scheduling parallel tasks. Application of gang scheduling in a Cloud Computing model, is based on the architecture of the Amazon Elastic Compute Cloud. The number of Virtual Machines available at any moment is dynamic and scales according to the demands of the tasks being serviced. This model is studied through simulation in order to analyze the performance and overall cost of gang scheduling with migrations and starvation handling. The results highlights that this scheduling strategy can be effectively deployed on clouds. The cloud platforms can be possible for high performance computing or high performance enterprise applications as detailed by Moschakis & Karatza (2011). The study takes both performance and cost while integrating task migration and starvation.
Ioannis & Helen (2012) proposed a gang scheduling algorithm with task migration and handling starvation time in scheduling parallel tasks. It is already applied in the areas of Grid and Cluster computing. The number of virtual machines available at any moment is dynamic and scales according to the demands of the tasks being serviced. The mentioned model is studied through simulation in order to analyze the performance and overall cost of Gang Scheduling with task migrations and starvation. Results highlight that this scheduling strategy can be effectively deployed on Clouds and that cloud platforms can be viable for HPC or high performance enterprise applications.

In parallel computing environments, the context of directed acyclic graph scheduling as stated by Gerasoulis & Yang (1992), grouping of tasks is done through clustering to reduce communication dependencies. It is proposed that scheduling strategy focuses on grouping independent tasks with small processing requirements, into suitable tasks with larger processing requirements. It also schedules the task in accordance with in-deterministic network conditions. Traditional way of task scheduling in cloud computing uses the direct tasks of users as the overhead application base. The problem is that there may be no relationship between the overhead application base and the way that different tasks cause overhead costs of resources in cloud systems. Activity-based costing measures both the cost of the resources and the computation performance. The cost of every individual resource is different. Tasks are sorted according to their priority and they are placed in three different lists based on three levels of priority namely high priority, medium priority and low priority. For computation of tasks, the system can take from high priority list first, then medium and then low. This proposed algorithm is applied to group the above lists in order to allocate the task-groups to different available resources.
The improved activity based costing method selects a set of resources to be used for computing. It groups tasks according to the processing capability of resources available. Coarse-grained tasks are processed in the selected resources, so that the computation-communication ratio is reduced. The experimental results using a simulator show that the time taken to complete tasks after grouping the tasks is very less when compared with time taken to complete the tasks without grouping the tasks.

2.6.2 Resource Clustering

Aneka described by Chu et al (2007), recommended the commercialized through Manjrasoft, a .NET-based service-oriented resource management platform. It is designed to support multiple application models, persistence, security solutions and communication protocols. The selection can be changed at any time without affecting an existing aneka ecosystem. To create an anekacloud, the service provider needs to start an instance of the configurable aneka container hosting required services on each selected desktop computer. The purpose of the aneka container is to initialize services and acts as a single point for interaction with the rest of the aneka Cloud. Aneka provides SLA support such that the user can specify QoS requirements such as maximum execution time and budget.

The user can access the anekacloud remotely through the gridbus broker. The gridbus broker proposed by Venugopal et al (2006) also enables the user to negotiate and agree upon the QoS requirements to be provided by the service provider. Aneka is initially developed as a 3rd generation enterprise grid technology. Recently, various new capabilities have been added to exhibit properties and potentials of the cloud computing paradigm.

An enterprise grid based by Chien et al (2003), harnesses computing resources of desktop computers within an enterprise without affecting the
productivity of their users. Hence, it increases the amount of computing resources available within an enterprise to accelerate application performance. This capability can be combined with other dedicated resources in the enterprise to enhance the overall system capability and reliability. To support scalability, the aneka container is designed to bare minimum functionality needed for an anekacloud node. It provides the base infrastructure that consists of services for persistence, security such as authorization, authentication and auditing and communication. The aneka container can host any number of optional services that can be added the capabilities of an anekacloud node. The optional services are indexing, scheduling, execution and storage services. This provides a single, flexible and extensible framework for orchestrating various application models. This section describes how pricing can be implemented in an anekacloud with advanced reservations.

Young et al (2009) investigated the problem of scheduling workflow applications on grids and presented a novel scheduling algorithm as a solution. The scheduling is performed by accounting both completion time and resource usage. Since the performance of grid resources changes dynamically and to estimate their performance is very difficult. This algorithm is incorporated with rescheduling to deal with unforeseen performance fluctuations.

To create an anekacloud, a bi-hierarchical advance reservation mechanism is implemented with a reservation service at a master node. It coordinates multiple execution nodes and an allocation service at each execution node. It also keeps track of the reservations at that node. This architecture was previously introduced in Venugopal et al (2008). To use the anekacloud, the resource user first makes advanced reservations during the reservation phase. If the reservation phase is successful, the user/broker can then submit applications later during the execution phase. Figure 2.3 shows that the process of allocating advanced reservations happens in two levels namely
the allocation service at each execution node and the reservation service at the master node.

Both services are designed to support pluggable policies so that the provider has the flexibility to easily customize and replace existing policies for different levels without interfering nodes with the overall resource management architecture. During the reservation phase, the user/broker submits reservation requests through the reservation service at the master node. The reservation service discovers available execution nodes in the anekacloud by interacting with the allocation service on them. The allocation service at each execution node keeps track of all reservations that have been confirmed for the node and can thus check whether a new request can be satisfied or not.

Juve et al (2010) studied the performance and cost of using different storage systems that could be used as data hosts during execution of scientific workflows on Clouds. The storage systems used for execution the workflow applications were Amazon S3, NFS, GlusterFS11 and PVFS as in Haddad (2000). It shows the impact of type of storage system used on the application’s runtime and cost. Specifically, it is observed that Amazon S3 provide improved performance due to the caching of data at the client side, but performed poorly when the number of small files were large. In addition, amazon would charge per transaction for accessing files, which meant the total cost would increase if amazon alone was used as a storage server. In terms of performance benefit, it is observed that addition of more cloud resources used for workflow executions improved the performance, but only to a certain extent. In order to minimize total cost as well as make use of Cloud resources, a single virtual cluster instead of large number of resources is proposed, so that multiple workflows could be executed in sequence.

Adaptive resource allocation algorithm for the cloud system with pre-emptable tasks by Jiayin et al (2010) is proposed. These algorithms adjust
the resource allocation adaptively based on the updated state of the actual task executions. The experimental results show that these algorithms work significantly in intense resource contention situation. An IaaS cloud system is considered. The computational resources are available in the form of VMs deployed in a provider's data center. Cloud computing users can request number of cloud services simultaneously. So there must be a provision for all resources to be available for the requesting user in efficient manner to satisfy their need. A review of various policies for dynamic resource allocation in cloud computing is shown based on Topology Aware Resource Allocation (TARA), linear scheduling strategy for resource allocation and dynamic resource allocation for parallel data processing are stated by Ronak & Sanjay (2013).

Deterministic Resource Rental Planning (DRRP) model, using a mixed integer linear program, to generate optimal rental decisions by fixed cost parameters. It systematically analyses the predictability of the time-varying spot instance prices in Amazon EC2 and finds that the best achievable prediction is insufficient to provide a close approximation to the actual prices. This fact motivates to propose a Stochastic Resource Rental Planning (SRRP) model that explicitly considers the price uncertainty in rental decision making are stated by Han et al (2012).

Michael & Abishek (2009) used resource usage histograms which provide an accurate representation of an individual node’s dynamic resource usage pattern. It enables the satisfaction of statistical resource requirements. It potentially creates a scalability problem in a large wide-area distributed system. The statistical information for each node would be represented by multiple histograms, to different resources and different time scales. Disseminating this kind of detailed data over the network can create significant network traffic, making it infeasible for each node to have a global node-level behavioral view
of the entire system. Moreover, if the goal is to find multiple nodes meeting a certain requirement, it would be desirable to combine this discovery process rather than having to find individual suitable nodes separately.

Aggregation, particularly hierarchical aggregation, is a common technique employed in large distributed systems for the scalable dissemination of information. Aggregation essentially compresses the amount of transmitted data in the system while preserving the overall information content. In the context of resource discovery, a suitable “compression” of the node is achieved. The resource usage patterns is used to achieve a desirable tradeoff between the quality of resource discovery and the overhead of network data transmission in the system. The goal is to achieve the same quality of resource discovery as a global resource discovery system with full historical node-behavioral knowledge, but to significantly compress the amount of necessary node-behavioral representation data in the system in order to achieve scalability. Such an aggregation of node resource usage distributions for a group of nodes can be used to represent an accurate approximation of any individual node’s resource usage.
This is important for the discovery of desirable nodes based on a resource requirement and the collective resource usage behavior of the group of nodes. This can provide information about load patterns or resource usage behavior for a set of related nodes. For example, those in the same geographical area or running the same application. The overall capacity of the group is used to track the overall resource usage, i.e. for audit or accounting purposes. A naive approach to aggregation for a set of nodes would be to compute the average resource capacity distribution across all nodes. An example of this
naive approach, shows the aggregation of two nodes, one skewed towards low capacity and the other skewed towards high capacity. Average representation loses this information about the individual nodes and appears to represent a set of bimodal nodes. While averaging allows the estimation of overall capacity of the group of nodes, it is a poor representation of individual node-level behavior. This is because it does not account for the heterogeneity of the nodes in the system and fails to capture important behavioral differences between individual nodes. Thus, this approach could result in a highly inaccurate view of individual node resource capacities.

To account for the heterogeneity of nodes, the notion of a resource bundle is used. It is nothing but an aggregation of a group of nodes with similar resource capacity distributions. By combining only similar nodes together, an aggregation process will preserve the individual node distributions more accurately. First, a group of nodes are bundled based on the similarity of their resource capacity histograms. Second, each bundle produces a representative distribution that can be used to characterize the whole bundle. Clustering algorithm must be able to handle such multi-element data. The data to be clustered is not single-point, but multi-element (consisting of multiple histogram bins). The node resource usage histograms could represent arbitrary distributions, and cannot be assumed to conform to standard distributions (e.g., Gaussian, uniform, etc.).

The clustering algorithm must be distribution-free, i.e., it must not assume the existence of a standard distribution or certain parameters. The clustering algorithm must not only identify the closely related set of nodes, but it is desirable if it produce a compact representation of the collective resource usage of these nodes. Such a representation can be used to easily characterize the nodes in a bundle. (e.g., high-capacity/low-capacity, etc.). During aggregation, the creation of resource bundles that closely approximate
individual node behavior for a similar set of nodes, the question is whether bundles can be further aggregated to provide a meaningful view of the combined set of nodes corresponding to multiple bundles. The ability to combine resource bundles is particularly desirable in a large system where one may want to get concise estimates of resource capacities of nodes at different granularities; for instance, at a local site level, at an administrative domain level or at a global level.

Building on the assumed ability of the system-employed overlay to support a hierarchical information structure, the use of hierarchical aggregation in combination with resource bundles through the use of recursive clustering, i.e., successive clustering of bundle representatives at different levels of the hierarchy is proposed. Figure 2.4 is a high-level illustration of this process. Groups at the bottom level are individual sets of node distributions. These nodes are initially clustered, producing bundle representatives which are then propagated to the next level of the hierarchy to create level-2 representatives, thus beginning the recursive representative clustering process. Note that this recursive aggregation must incorporate individual bundle cardinalities during the clustering process to minimize the loss of representative data. Representatives with highly different cardinalities might be combined. If the bundle representatives are all treated alike, low cardinality representatives might have an unusually large influence in the formation of higher-level representatives.

Algorithm proposed by Waheed et al (2011) is suitable for web applications where response time is one of the important factors. For web applications guaranteeing average response time is difficult because traffic patterns are highly dynamic and difficult to predict accurately and also due to the complex nature of the multi-tier web applications it is difficult to identify bottlenecks and resolving them automatically. This provisioning technique
proposes a working prototype system for automatic detection and resolution of bottlenecks in a multi-tier cloud hosted web applications. This improves response time and also identifies over provisioned resources.

![Hierarchical aggregation](image)

**Figure 2.4 Hierarchical aggregation**

VM based resource management is a heavy weight task. This is less flexible and less resource efficient. To overcome this, a lightweight approach called Elastic Application Container (EAC) is used for provisioning the resources where EAC is a virtual resource unit for providing better resource efficiency and more scalable applications. This EAC–oriented platform and algorithm is to support multitenant cloud used by Sijin et al (2012). Dynamic creation of the tenant is done by integrating cloud based services on the fly. But dynamic creation is by building the required components from the scratch. Even though multitenant systems save cost, but incur huge reconfiguration costs. This approach allows clients to specify their requirements which are not supported in previous techniques. This approach proposes a novel user interface-tenant selector model which enables cloud based services to be
systematically modeled and provisioned as variants of existing service tenants in the cloud. This considers functional, non-functional and resource allocation requirements which are explicitly specified by the client via the user interface component of the model. So the cost and time is saved in this approach as stated by Lakshmi et al (2012).

Technique proposed in makes use of the provisioner called Adaptive Power-Aware Virtual Machine Provisioner (AP-AVMP) where the resources are provisioned dynamically from the resource pool. This is from Infrastructure-as-a-Service provider point of view where the custom Virtual machines are launched in appropriate server in a data center. The cloud data center considered is heterogeneous and large scale in nature. The proposed Meta scheduler maps efficiently a set of VM instances onto a set of servers from a highly dynamic resource pool by fulfilling resource requirements of maximum number of workloads. This technique reduces power consumption without affecting performance.

Server consolidation is a technique to save on energy costs in virtualized data centers. The instantiation of a given set of virtual machines to physical Machines can be thought of as a provisioning step where amount of resources to be allocated to a VM is determined and a placement step which decides which VMs can be placed together on physical machines thereby allocating VMs to PMs. Here a provisioning scheme is proposed which takes into account acceptable intensity of violation of provisioned resources. Correlation among aggregated resource demands of VM is considered when VMs are mapped to PMs. This reduces number of servers (up to 32%) required to host 1000 VMs and thus enables to turn off unnecessary servers represented by Kishaloy et al (2012).

In Cloud Computing federated Cloud environment is used when the resource requirement of user requests exceeds the resource limits of Cloud
providers’ resources. It is desirable to reduce SLA violation which can be achieved through load balancing algorithm that is threshold based. This algorithm allocates VMs in order to balance the load among multiple datacenters in a federated cloud environment by focusing on reducing users’ SLA violation as highlighted in Komal & Sarje (2012). Provisioning of collection of virtual machines having different placement constraints given a set of physical machines with known specifications is done by two approaches by Lei et al (2013). The first is based on the formulation of problem of an integer linear programming problem which provides solution for optimal VM placement. The second is a heuristic based on classifying requests into different categories and satisfying the constraint in a particular order using a first fit decreasing (FFD) algorithm planned by Bhavani & Guruprasad (2014).

Kun et al (2014) propose a framework for supporting knowledge discovery application running in cloud environment as well as a holistic approach to predict the application execution times. Uses rough sets theory to determine redact and then compute the execution time prediction. The history of data can be described as an Information System (IS) in which every instance is the history tasks whose execution times and other attributes have been stored. The attributes in the IS are features of history tasks. In accordance with the terms of rough set theory, task execution time is the decision attribute, accordingly, the condition attributes are composed of the other attributes.

Rough set model is intuitive and easy to draw causal relationship between attributes, make it easier to determine the dependencies between the stored attributes and the tasks execution times. Therefore, recognize the similar task according to the condition attributes that are dependent and importance in deciding the execution time. So, some attributes which strongly impact the execution time can constitute a fine similarity template. Having changed the
question of tasks execution time as a IS question, then basis notions that are appropriate in deciding the similarity template.

The function of similarity template in task execution time prediction is to recognize a set of features on the foundation of which to compare tasks. A similarity template should be composed of the higher significant attributes that decide the execution time without any redundant attributes. A redact set should be composed of the minimal number of condition properties which have the same discriminating ability as the whole IS. Accordingly, the similarity template is same as a redact set which consists of the important contribution attributes. Obviously, if there are too many properties in the template, it is not able to better identify similarities. For example, if n features are included in the similarity template, two tasks are similar only because they are recognized according to all n features. However, this method will largely restrict our capacity to detect similar tasks since not all attributes are certainly related in deciding the execution time. On the contrary, this method will cause mistakes, since tasks that are similarities very much may be measured dissimilar even though they are different in one feature which had little impact on the execution time.

Rough sets theory has very apposite and proper constructs for recognizing the attributes that best define similarity for predicting tasks execution time. A similarity template should contain properties which largely impact the execution time and remove the inappreciable factors. This makes sure that the standards with which to compare tasks for similarity have an important impact on deciding execution time. Therefore, tasks that have the same features regarding these standards will have similar execution time. The spirit of the algorithm, which proposed comes from the number of attributes within a given discernibility matrix, proposes to join dynamic data related to the performances of various knowledge discovery services in the cloud
computing environment for supporting the prediction. This information can be joined as additional metadata stored in cloud environment. Experimental result verifies that the proposed algorithm supply a general solution for the problem of web service execution time prediction in cloud environment.

2.7 WORKFLOW METHODS BY DAG

Kwok & Ahmad (1999) surveyed different static scheduling techniques for scheduling application in form of directed acyclic graphs onto homogeneous platforms. In their model, tasks are scheduled onto multiprocessor systems. This model assumes that communication is achieved solely by message passing between processing elements. They proposed a taxonomy that classifies the scheduling algorithm based on their functionality. They also provided survey examples for various algorithms along with the overview of the software tools used for scheduling and mapping.

Kaur & Verma (2012) proposed a modified genetic algorithm for single user tasks in which the fitness is developed for the formation of solutions in order to achieve the time minimization and it in compared with existing heuristics. Experimental result shows that, under the heavy load, the proposed algorithm exhibits a good performance. However, the proposed method is of single user type, and is not suitable for general cloud applications.

Liu et al (2011) proposed a resource preprocessing method for Grid system, but the method is still infancy. Mohsen & Rajkumar (2010) have proposed two market-oriented scheduling policies. It aims to satisfy the application deadline by extending the computational capacity of local resources through hiring resource from Cloud providers. These policies do not have any prior knowledge about the application execution time. The proposed policies are implemented in Gridbus broker as a user-level broker.
Random scheduling algorithm is mainly used service-level scheduling. But before that, the illustration of DAG task graphs is represented in Figure 2.5. DAGs based modeling is widely used in workflow area. As depicted in Figure 2.6, in a DAG, each node represents a workflow task and directed links indicate the task dependencies. To facilitate cloud workflow scheduling, each task node in a DAG also associated with its QoS constraints for example the execution time and execution cost.

![Task Graph](image)

**Figure 2.5 Task Graph**

In a DAG, each node (except the root node, i.e. the first task) will have one and only one parent node which indicates its parent task. A child task will become ready and can be executed until its parent task is completed. However, it is possible that a child task has more than one parent task in real world processes. In such a case, as depicted in Figure 2.6 (b), a transformation process will create multiple instances of the child task for each of its parent tasks. Additionally, for an iterative task as depicted in Figure 2.6 (c), a
transformation process will create the same number of instances for the iterative task as its iteration times. Therefore, DAGs can visually represent common workflow structures with single parent-child relationships. This simple representation of DAGs can benefit workflow scheduling algorithms in the efficient validation of the precedence relationships between workflow tasks.

![Sample DAG Graph](image)

**Figure 2.6 Sample DAG Graph**

An example workflow defined by a DAG is presented in Figure 2.6 (d). Kumar et al (2015) state that tasks or tasks of the users would require to be executed in a particular order to complete the whole task. Workflow scheduling manages the execution of the inter-dependent tasks on the distributed resources. Workflow scheduling algorithms are used to allocate the resources to workflow
tasks in a manner that preserves the dependency constraints. At the same time, the tasks must be scheduled efficiently in order to minimize the execution time as well as cost incurred in using the heterogeneous resources of the cloud.

Table 2.1  Detailed categories of limitation identified from the above Literature Survey

<table>
<thead>
<tr>
<th>Name of Authors &amp; Year</th>
<th>Mechanism</th>
<th>Contribution</th>
<th>Limitation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Huifang et al (2014)</td>
<td>Scheduling algorithm</td>
<td>Control Structure Reduction (CSR) Algorithm based on DAG by means of mergers and reduction</td>
<td>Tasks are allocated based on time float distribution</td>
</tr>
<tr>
<td>Ganga &amp; Karthik (2013)</td>
<td>Fault tolerance Techniques</td>
<td>Prevents node &amp; network failure</td>
<td>Scientific workflows are hard to program</td>
</tr>
<tr>
<td>Lovejit &amp; Sarbjeet (2013)</td>
<td>Scheduling algorithm</td>
<td>Dependency is presented by parent child relationship</td>
<td>Dependency constrains should be presented</td>
</tr>
<tr>
<td>Ronak &amp; Sanjay (2013)</td>
<td>Resource Clustering</td>
<td>Topology Aware Resource Allocation</td>
<td>Parallel data processing</td>
</tr>
<tr>
<td>Arash et al (2012)</td>
<td>Scheduling algorithm</td>
<td>Reliable scheduling algorithm</td>
<td>Efficiency increased. Major tasks are divided into subtasks</td>
</tr>
<tr>
<td>Hu wu et al (2012)</td>
<td>Scheduling algorithm</td>
<td>Priority Impact Scheduling</td>
<td>Higher priority can access resource</td>
</tr>
<tr>
<td>Scheduling algorithm</td>
<td>Algorithm (PISA)</td>
<td>faster than lower priority workflow</td>
<td></td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------</td>
<td>-------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Genez et al (2012)</td>
<td>user priority is taken into account</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jiahui et al (2011)</td>
<td>Scheduling is based on Integer Linear Program (ILP)</td>
<td>Used only for the larger deadlines</td>
<td></td>
</tr>
<tr>
<td>Moschakis &amp; Karatza (2011)</td>
<td>BAalance-Reduce Algorithms</td>
<td>A task is not completed until all tasks are finished</td>
<td></td>
</tr>
<tr>
<td>Zheng (2010)</td>
<td>Gang scheduling by integrating task migration and starvation</td>
<td>Service based on scales according to the demand of the task</td>
<td></td>
</tr>
<tr>
<td>Sudha &amp; Selvarani (2010)</td>
<td>Improved cost based Scheduling algorithm</td>
<td>Topology control and data locality are not possible.</td>
<td></td>
</tr>
<tr>
<td>Greenberg et al (2009)</td>
<td>Proactive measures to solve failures</td>
<td>Applicable for IaaS cloud system</td>
<td></td>
</tr>
<tr>
<td>Maria et al (2009)</td>
<td>Workload Aware Task Scheduling(WATS)</td>
<td>The task allocations is less optimal due to approximation</td>
<td></td>
</tr>
<tr>
<td>Authors</td>
<td>Methodology</td>
<td>Description</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------------</td>
<td>------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------</td>
</tr>
<tr>
<td>Gupta &amp; Harchal (2009)</td>
<td>Resource Allocation Strategies</td>
<td>Using history based Scheduling</td>
<td>Resources management based on admission control policies</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Workload requires some knowledge about global state of system</td>
</tr>
<tr>
<td>Pandey et al (2009)</td>
<td>Task Clustering</td>
<td>To schedule data intensive task for medical application.</td>
<td>Same cluster has less file transfer rate</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Young et al (2009)</td>
<td>Scheduling Algorithm</td>
<td>Scheduling work based on completion time and resource usage</td>
<td>Deals with unforeseen performance fluctuation</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>resource usage pattern</td>
<td></td>
</tr>
<tr>
<td>Tan &amp; Gurd (2007)</td>
<td>Resource Allocation Strategies</td>
<td>Combinatorial exchange for resource allocation in Grid environment</td>
<td>It does not fit for the enterprise cloud environment</td>
</tr>
<tr>
<td>Researcher(s) (Year)</td>
<td>Topic</td>
<td>Description</td>
<td>Applicability</td>
</tr>
<tr>
<td>---------------------</td>
<td>------------------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Urgaonkar &amp; Chandra (2005)</td>
<td>Scheduling algorithm</td>
<td>Monitoring the workflow which containing in network queues</td>
<td>Multi-tier configuration is considered.</td>
</tr>
<tr>
<td>Schnizler et al (2005)</td>
<td>Resource Allocation Strategies</td>
<td>Workflow oriented application which contains multiple services not running at same time in cloud</td>
<td>It leaks the ability to deal with a workflow oriented application</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>Singh et al (2005)</td>
<td>Task Clustering</td>
<td>Restructuring the workflow to reduce the dependencies</td>
<td>Applicable for both centralized and distributed task submission</td>
</tr>
<tr>
<td>Elnikety et al (2004)</td>
<td>Scheduling algorithm</td>
<td>Shortest Task First (STF) Scheduling in Grid</td>
<td>System load is kept under described level</td>
</tr>
<tr>
<td>Waldspurger et al (1992)</td>
<td>Resource Allocation Strategies</td>
<td>Spawn auction based management systems uses idle CPU time</td>
<td>It does not fit for enterprise cloud Environment</td>
</tr>
<tr>
<td>Gerasoulis &amp; Yang (1992)</td>
<td>Task Clustering</td>
<td>Task grouping by clustering to reduce communication dependencies</td>
<td>Priority wise allocation of task group to different available resource</td>
</tr>
</tbody>
</table>

## 2.8 RESEARCH GAP

CC is a computing model in which various tasks are assigned across an environment. It is a combination of connections between task and resources as services that can be accessed over the network. Computing is performed with resources as services efficiently, delivered and utilized making the vision of computing utility.
• To improve the performance of the workflow application in cloud environment by reducing the execution time.

• To proactively identify the fault occurring in the resource.

• To enhance the resource and task clustering mechanism.

2.9 SUMMARY

This chapter presents an outline study about workflow in distributed environment. Several scheduling policies like shortest task scheduling, divide and conquer etc are detailed. Scheduling of tasks plays vital role of executing bag of tasks. Various resource allocation strategies were discussed on distributed and cloud computing. Clustering of tasks like EC2 and resources like Aneka, resource rental were outlined with their advantages to classify their workflow. A Study over a simulation tool CloudSim for implementation and testing the workflow management is carried out. Finally, the clustering methodology is analyzed for efficient task scheduling with resource management by minimizing the task failures and starvation time is identified.