COST EFFECTIVE DEADLINE CONSTRAINED WORKFLOW SCHEDULING IN CLOUD COMPUTING

A Project Report

Submitted as a Part of Major Project-2

Master of Technology in Information System

By

SHUBHAM JAIN

2K16/ISY/15

Under the Guidance of:

Mr. JASRAJ MEENA

(Assistant Professor - Information Technology)

Delhi Technological University

(Formerly Delhi College of Engineering).

ShahbadDaulatpur, Bawana Road, Delhi – 110042

May-2018

CERTIFICATE

This is to certify that Major Project report-2 entitled "**COST EFFECTIVE DEADLINE CONSTRAINED WORKFLOW SCHEDULING IN CLOUD COMPUTING**" submitted by **SHUBHAM JAIN (RollNo. 2K16/ISY/15)** for partial fulfillment of the requirement for the award of degree Master Of Technology (Information System) is a record of the candidate work carried out by him under my supervision.

Assistant Prof:Mr. JASRAJ MEENA

Project Guide

Department of Information Technology

Delhi Technological University

DECLARATION

We hereby declare that the Major Project-2 work entitled "**COST EFFECTIVE DEADLINE CONSTRAINED WORKFLOW SCHEDULING IN CLOUD COMPUTING**" which is being submitted to Delhi Technological University, in partial fulfillment of requirements for the award of degree of Master of Technology(Information System) is a bonafide report of Major Project-2 carried out by me. The material contained in the report has not been submitted to any university or institution for the award of any degree.

SHUBHAM JAIN

2K16/ISY/15

ACKNOWLEDGEMENT

As a matter of first importance i might want should thank those Almighty, who need continuously guided me on worth of effort on the right way of the life. My best much appreciated would with my guardians who gave capability What's more quality over me to complete this worth of effort.

I owe An profound gratitude on my one task aide right hand **Assistant Prof. Mr. JASRAJ MEENA**who need been An consistent sourball about impulse will me All around those time from claiming this venture. It might have been as much skillful guidance, steady consolation What's more basic assessment that aided me with create another knowledge under my undertaking. Her calm, gathered and professionally immaculate style for taking care of circumstances not best steered me through each problem, as well as aided me with develop as An developed man.

SHUBHAM JAIN

2K16/ISY/15

ABSTRACT

The technology of cloud computing is growing very quickly, thus it is required to manage the process of resource allocation. In this paper, load balancing algorithm based on honey bee behavior is proposed. Its main goal is distribute workload of multiple network links in the way that avoid underutilization and over utilization of the resources. This can be achieved by allocating the incoming task to a virtual machine (VM) which meets two conditions; number of tasks currently processing by this VM is less than number of tasks currently processing by other VMs and the deviation of this VM processing time from average processing time of all VMs is less than a threshold value. The proposed algorithm is compared with different scheduling algorithms; honey bee, ant colony, etc.. The results of experiments show the efficiency of the proposed algorithm in terms of cost, flow time and span time

Keywords: *Cloud computing; honey bee; load balancing; swarm intelligence; virtual machine*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

LIST OF ABBREVIATIONS

S No	Abbreviated Name	Full Name
1	VMs	virtual machines (VMs).
2	RR	Round robin (RR)
3	ACO	ant colony optimization (ACO)
4	SI	Swarm Intelligence algorithms (SI)
6	PEs	Processing elements (PEs)

LIST OF EQUATIONS

CHAPTER 1

INTRODUCTION

1.1 Introduction

Cloud computing is Associate in Nursing rising technology entirely supported net, within which all the applications are hosted on a cloud, that consists of thousands of computers interlinked along during a advanced manner. The client needn't get the software package or computation platforms, as a result of it uses a pay as you utilize model. By mistreatment net facility, the client will use computation power or software package resources by paying cash just for the period he/she has used the resource.

Clients, datacenter, and distributed servers are the three elements of a cloud system. To manage information's associated with the cloud, finish users move with the purchasers. purchasers typically comprise 3 main classes like mobile, skinny and thick. Datacenter could be a assortment of servers hosting totally different applications. so as to subscribe for various applications, user has to connect with the datacenter. A datacenter is settled at a bigger distance from the purchasers. Distributed servers are the components of a cloud that are gift throughout the web hosting totally different applications.

Load leveling is that the method of redistributing the full load of a distributed system into individual nodes to confirm that no node is overloaded and no nodes were beneath loaded or idle. thus during a cloud atmosphere load leveling ensures that no VMs are overloaded, wherever some VMs are beneath loaded or doing little work. Load leveling tries to hurry up the execution time of applications. It conjointly ensures the system stability. it's an honest backup set up within the case of failovers.

In cloud computing atmosphere qualitative metrics like performance, resource utilization, measurability, time interval, fault tolerance and migration time may be improved by higher load leveling. the advance within the on top of factors can guarantee smart QoS to the shoppers. Load leveling algorithms are chiefly classified into two; Static and dynamic. Static load leveling algorithms doesn't take into account the previous state of the node, whereas distributing the load and it works well once nodes have a little variation within the hundreds. thus it's not appropriate for cloud atmosphere. The bee colony algorithmic

program primarily based load leveling technique could be a dynamic load leveling technique, that considers the previous state of a node whereas distributing the load.
Cloud computing provides shared processing resources and data. This can occur through the presence of a host application service provider so that the user does not need to buy a server or pay for the electricity of power and cooling. It's also convenient for communications and travels where remote workers, who can simply log in and use their applications wherever they are [1]. As increasing the number of users in cloud computing environment, the demand of shared resources is rapidly increased. Therefore, load balancing between these resources for scheduling tasks becomes a key challenge.
Load balancing is the process of distributing workloads and computing resources in a cloud computing environment. It allows enterprises to manage application or workload demands by allocating resources among multiple computers, networks or servers. Load balancing is often used to avoid the bottleneck, so that several characteristics of load balancing can be achieved such as: equal division of tasks across all hosts, facilitation in achieving service quality, improve overall performance of the system, reduce response time, and improve resource utilization [2].

Fig. 1 shows the load balancer of virtual machines (VMs). It assigns multiple tasks to VMs that execute them simultaneously by a way that guarantees a balance between these VMs. The primary goal of load balancing in a cloud environment is to balance the workload of the hosts in proportion to their capacities, which is measured in terms of their processor speed, available memory space, and bandwidth.

Load balancing algorithms are classified into two types; static and dynamic. Static algorithms are much simpler as compared to dynamic algorithms. Static algorithms work properly only when hosts have low variations in the load, since they do not take into account the previous state or the behavior of a host while distributing the load. Dynamic load balancing algorithms are more suitable for widely distributed systems such as cloud computing [3,4]. Round robin (RR) [5] is a well-known straightforward static scheduling algorithm. It allocates tasks to each node in turn, without considering the resource quantity of each VM and the execution time of tasks. Modified throttled algorithm [6] is a dynamic load balancing algorithm that uniformly distributes the incoming tasks among available VMs. However it doesn't consider resource utilization during task allocation.

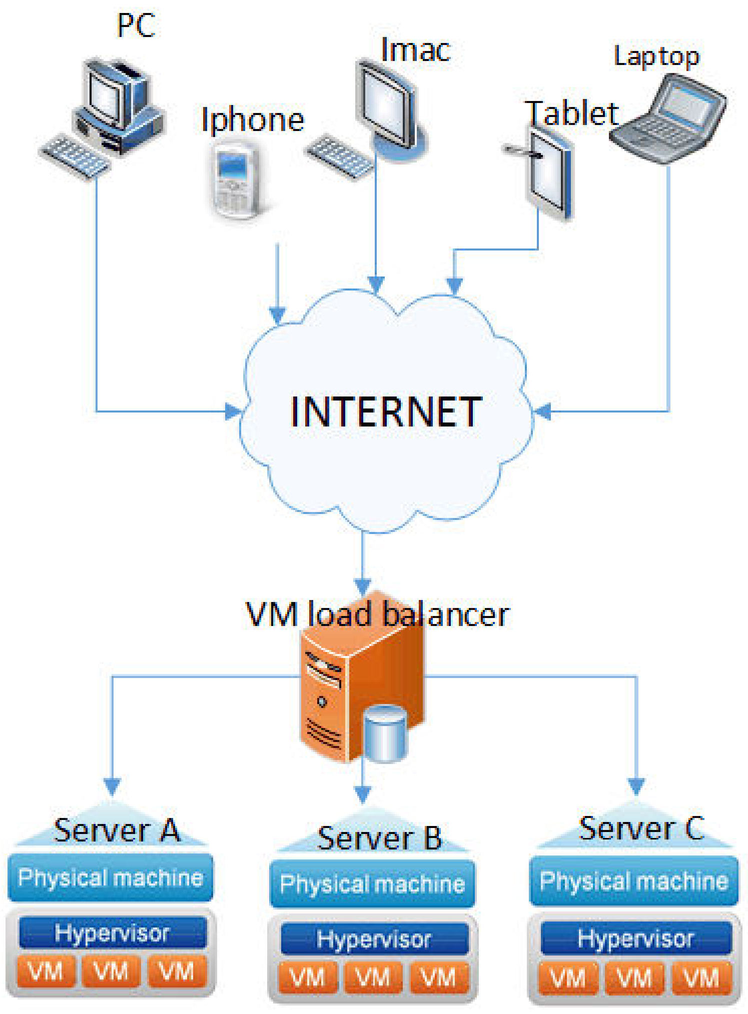


Fig. 1.1 VM Load Balancer[41].

Conventional load balancing algorithms have many drawbacks in cloud environment due to the changing workload dynamics. To address these challenges, Swarm Intelligence algorithms (SI), such as ant colony optimization (ACO), and artificial bee colony (ABC), are provided in recent decades [7]. They achieve a great progress in the dynamic situation of cloud computing. So many researches tended to study the algorithms based on SI to balance load among cloud environment such as foraging for food.

However, some of these algorithms have drawbacks such as causing many hosts overloaded, and getting low throughput..

The objective of this paper is to propose a load balancing algorithm aims to distribute the dynamic workload smoothly to all the hosts in the cloud to gain an improvement in both the utilization of resources and the speed of execution time. It allocates the incoming tasks to all available VMs. In order to achieve fairness and avoid congestion, the proposed algorithm allocates tasks to the least loaded VM and prevents the allocation of tasks to a VM when the variation of this VM processing time from average processing time of all VMs becomes more than or equal to a threshold value. This leads to a reduction of the overall response time and the processing time of hosts. In the proposed algorithm, variation of processing time of VM is the key limiting factor during the task allocation process because it avoids underutilization and over utilization of VMs. It also has a highly effect of the standard deviation that preserves the load balance of all system.

In this paper, a Load Balancing Algorithm based on Honey Bee behavior (LBA_HB) is proposed. It is completely inspired by the natural foraging behavior of honey bees. The allocated task updates the remaining tasks about the VM status in a manner similar to the bees finding an abundant food source, updating the other bees in the bee hive through its waggle dance[8]. The proposed LBA_HB algorithm has been simulated using CloudSim [9]. The proposed algorithm is compared with both conventional and SI based load balancing algorithms; round robin, modified throttled, ant colony, and honey bee algorithms. The results of experiments show the efficiency of LBA_HB in terms of span time, flow time and cost.

1.2 Existing Work

L-ACO employs ant colony optimization to carry out deadline-constrained cost optimization: the ant constructs an ordered task list according to the pheromone trail and probabilistic upward rank, and uses the same deadline distribution and service selection methods as ProLiS(ProLiS distributes the deadline to each task, proportionally to a novel definition of probabilistic upward rank, and follows a two-step list scheduling methodology: rank tasks and sequentially allocates each task a service which meets the sub-deadline and minimizes the cost) to build solutions. Moreover, the deadline is relaxed to guide the search of L-ACO towards constrained optimization

1.2.1 Disadvantages

- ❖ Absence Of Load Sharing.
- ❖ Not Cost Efficient
- ❖ Complex Mechanism

1.3 Proposed Work:

In bee hives, foraging honeybees give information to other bees about the position of the food source they have visited. A potential forager bee starts her career as an unemployed naive worker, that is, she has as no information of a food source in the field yet. She can start search for a source and thus become a scout (explorer). The initiation to fly out and start foraging is not due to following a waggle dance but due to some unknown internal, motivational factor or perhaps to some unknown external cue. Alternatively, a bee can start searching for a source as a response to attending a waggle dance and thus becomes a recruit. So the distinction between a recruit and a scout is that the recruit has stored estimated positional information in her memory, whereas the scout has not. As soon as a bee finds a source, it registers the essential s of this source in its memory and starts exploiting it, the bee is then an employed forager (exploiter) . The proposed LBA_HB is completely inspired by the natural foraging behavior of honey bees. The allocated task updates the remaining tasks about the VM status in a manner similar to the bees finding an abundant food source, updating the other bees in the bee hive through its waggle dance . This task updates the status of the VM availability and the load of the VMs.

1.3.1 Advantages

- ❖ Proper Load Sharing
- ❖ Cost Efficient
- ❖ Easy to Understand

1.4 LACO

Nowadays it is becoming more and more attractive to execute workflow applications in the cloud because it enables workflow applications to use computing resources on demand. Meanwhile, it also challenges traditional workflow scheduling algorithms that only concentrate on optimizing the execution time. This thesis investigates how to minimize execution cost of a workflow in clouds under a deadline and cost constraint and also discussses about the existing metaheuristic algorithm L-ACO as well as a simple heuristic ProLiS. ProLiS distributes the deadline to each task, proportionally to a novel definition of probabilistic upward rank, and follows a two-step list scheduling methodology: rank tasks and sequentially allocates each task a service which meets the sub-deadline and minimizes the cost. L-ACO employs ant colony optimization to carry out deadline-constrained cost optimization: the ant constructs an ordered task list according to the pheromone trail and probabilistic upward rank, and uses the same deadline distribution and service selection methods as ProLiS to build solutions. Moreover, the deadline is relaxed to guide the search of L-ACO towards constrained optimization. Experimental results show that compared with traditional algorithms, the performance of ProLiS is very competitive and L-ACO performs the best in terms of execution costs and success ratios of meeting deadlines. Nowadays, scientific and business applications can contain hundreds or thousands of computing tasks. The workflow
model is extensively applied to represent these applications via direct acyclic graphs (DAGs), where nodes represent tasks and edges represent dependencies among them [1]. These large-scale workflow applications are deployed in a distributed computing environment in order to execute the workflows in a reasonable amount of time. However, traditional distributed computing platforms, such as highperformance clusters and grid systems, are not only very expensive to build and maintain, but also inefficient in adapting to the surge of resource demands. With the emergence of cloud computing and the rapid deployment of cloud infrastructure, more and more large-scale workflow applications have been moved or are in active transition to the cloud, where users can dynamically acquire and release cloud resources on demand, and pay based on the usage.

Workflow scheduling aims to allocate each task in the workflow to a certain time slice of computing resources for execution, in order to meet some performance criterion. As a well-known NP-hard problem, it has been a hot research topic in the distributed computing community for many years [2]. Although a great number of algorithms have been proposed to minimize the total time of executing workflows, these algorithms cannot be directly applied to the cloud environment [3, 4]: in clouds, users should first select appropriate types and amount of services from the 'infinite' cloud resources to execute workflows; the cloud provider offers heterogeneous resources with various processing capabilities and costs, and the total usage cost of cloud ser-vices is also a critical user requirement besides execution time.

Faster cloud services are usually more expensive than slower ones, and therefore users face a time-cost tradeoff in selecting services. A general way to address this trade-off is to minimize monetary cost under a deadline constraint. Up to present, only a few approaches have been presented to address this problem in the literature [5-10] and they can be divided into two categories: local heuristics [5, 6] and metaheuristics [7-10]. The former employs a simple heuristic based on a deadline distribution and can fall into a locally optimal solution easily. By contrast, existing metaheuristic methods do not exploit the characteristics of the scheduling problem fully to conduct an effective search. Furthermore, existing methods pay little attention to the situation where the sub-deadline of a task is not met during the scheduling procedure, which is not equivalent to violation of the workflow deadline. ProLiS distributes the deadline to each task based on a novel definition of probabilistic upward rank and then follows a two-step list scheduling method: order tasks according to probabilistic upward rank and then sequentially allocate each task to a service that meets the sub-deadline and minimizes the cost. L-ACO carries out the deadline-constrained cost optimization based on ant colony optimization : the ants construct an ordered task list according to a pheromone trail and the probabilistic upward rank of a task, and then use the same deadline distribution and service selection methods as Pro-LiS to build solutions. The MMAS (Min Max Ant System) framework is utilized for the pheromone updating in LACO.

Moreover, in order to guide the search towards a near-optimal solution meeting the deadline, the deadline constraint is relaxed and this relaxation is gradually diminished until is removed completely. Correspondingly, an ε comparison between solutions is introduced. In the experiments, the proposed approaches are compared with the two most-cited ones from the literature (i.e., IC-PCP and PSO), and the experimental results reveal that the performance of

ProLiS is very competitive and in each setting L-ACO attains the highest success ratio of meeting deadlines, and the least execution cost.

CHAPTER 2
LITERATURE REVIEW

The study dignifies different types of algorithms used for scheduling workflows in cloud environment[42].

2.1 Particle Swarm Optimization(PSO):

Proposed by Eberhart and Kennedy, PSO[9] is an evolutionary algorithm . It is a computational method which tries to improve the efficiency of a solution iteratively. Here each particle represents a solution and is depicted in the form of vectors of position and velocity. Position represents the solution to the problem and velocity specifies movement in search space, both of which are computed using mathematical formulas. Each particle has a local best position called pbest and a best position of population called gbest. Apart from finding pbest the algorithm also guide the particle towards gbest solution . After each generation these positions(pbest and gbest)are updated. This is done to guide the algorithm towards best solution. Once solution is achieved the total execution cost and time of solution is calculated to satisfy the objectives. However PSO is not so efficient with tighter deadlines and also has weaker performance in terms of cost and time when compared with other algorithms.

2.2 Dynamic Objective Genetic Algorithm(DOGA):

DOGA[10] make use of dynamic objective strategy and is similar to that of natural selection process. Proposed by Holland, it make use of selection, crossover and mutation process to achieve its optimization objective. With a number of similarities, thing that makes DOGA different from traditional GA is it's dynamic objective strategy. It make use of the strategy to evaluate the chromosomes. Encoding in DOGA is similar to that of PSO except that here only integer part is considered and not float. In selection process, roulette wheel method is used which states that the chromosome which is fittest is more likely to get selected. Crossover process is based on probability. To take the decision of crossover, a random number is generated between 0 and 1 and if the numbers value is found to be less than the already fixed crossover probability than crossover will be done. With every coordinate of chromosome, a mutation probability is associated. Mutation is also a random number based event. To avoid the modification in best solution with upcoming generations, best solution is stored and is only updated if the best solution in new generation is better than the existing one. Dynamic objective strategy is used in case if a chromosome violates the deadline constraint, it's fitness value will be changed to something far from acceptance. If all chromosomes fails then DOGA changes the optimization objective to just execution time until some chromosome achieves a feasible solution. This helps DOGA to find feasible solution even in tighter deadlines.

2.3 Multi-Cloud Partial Critical Paths(MCPCP):

MCPCP[11] is based on the concept of Partial Critical Path(PCP). PCP is a path from the starting unscheduled task to it's critical parent(an immediate predecessor which has latest arrival time to the task) and each task in the path should have latest arrival time to it's successor. MCPCP works by assigning sub-deadlines to PCP's instead of individual tasks. Then Best Fit Instance is found for a PCP so that each task of the PCP can finish it's execution before it's latest finish time and with least increment in cost. In Best Fit Instance, the algorithm make use of greedy approach , in which it will schedule the path on all applicable existing instance in order to find an instance that shows minimum cost growth. An instance is called applicable if: when a path is scheduled on that instance, then each task of that path is able to get completed before it's latest finish time and the increment in cost by scheduling new path on the instance should be less than initiating a new instance of the same machine. If none of such applicative instance exist that meet the criteria then a new instance of the cheapest service will be initiated.

2.4 Proportional Deadline Constrained (PDC):

PDC[12] also works with the same objective of cost effective schedule within a deadline. It usually produce lower failure rates with tighter deadlines. It is divided in four parts: Workflow Leveling, Deadline distribution, Task selection and Instance selection. In workflow leveling all tasks are distributed in different levels so that parallel execution of numerous tasks can be maximized while maintaining the dependencies. Tasks on the same level did not have any dependencies among them. It can be defined as an integer value that denotes the maximum number of edges from a task to end task. Deadline distribution phase distributes the user deadline as sub-deadline or level deadline among all the levels.

Level deadline is assigned in such a way that each task of that level should be able to complete its execution under that deadline. Point to remember is that the level with longer tasks receives a larger deadline. Task selection is done for the ready tasks. Readiness of a task is based on its parents execution time and required data transfer time. For prioritizing the ready tasks downward ranking mechanism is used. During instance selection we need to maintain a balance between cost and time for which we make use of cost time tradeoff factor. Each task is executed on each type of instance for the cost and time measures, keeping in mind that if time value comes out to be negative means task on current instance will exceed the deadline.

2.5 Ant Colony System(ACS):

ACS[13] make use of pheromone and heuristic information to build a good solution. Encoding scheme for a solution is dimensional where each dimension represents a task and its value represents the resource. Its fitness objectives are same as of other algorithms that is to minimize execution cost under deadline constraint. In the cloud computing model unlike traditional times we use greedy approach to find a solution using the set of tasks which we get as input. Calculation of total execution cost of the solution is performed, which is used as a parameter for setting initial pheromone value of ants. Then comes the process of solution construction. In this at each step each ant will select a resource for some particular task, that is parallel mechanism will be used. Selection will be done in the way of exploration or exploitation. For this a random number is generated and if the criteria is met exploitation will be chosen means ant will choose higher pheromone and heuristic value otherwise roulette wheel method. Pheromone updation happens globally as well as locally. Global updation occurs after a generation whereas during a solution construction due to both new pheromone deposition by ants on path and pheromone evaporation local updation also occurs.

2.6 Deadline Constrained Critical Path(DCCP):

DCCP[14] is a list based scheduling divided in two parts: Task Prioritization and Task Assignment. Task Prioritization aims at dividing set of tasks into different levels so as to increase the measure of parallel execution of tasks since tasks in a level donot have any dependencies. Level of a task is the maximum number of edges in the paths from the current task to the exit task. For level distribution Deadline Bottom Level(DBL) and Deadline Top Level(DTL) are the algorithms used. Once leveling is done we distribute the user deadline to each level as a level deadline such that each task in a level should be able to complete its execution under the deadline. In this algorithm we find all the CCP in the workflow. CCP stands for Constrained Critical Path. Critical Path is the longest path from starting to end node of a graph. CCP consists of a set of tasks which are ready for execution, where readiness further depends on the parents execution completion time and required data transfer time. We find the CCP using modified upward and downward rank. All the tasks are then sorted on the basis of rank and the task with highest values are selected so as to from a CP. Similarly all other CPs are found. In task assignment phase we aims to find best suitable task for executing CCP such that all tasks in a CCP are executed on the same instance so as to minimize the data transfer and execution time and hence meet CCP sub-deadline. While selecting an instance algorithm prefers instances having idle billing interval as execution will be cost free. If no such instance exist then a new instance is launched. Even if deadline is tight to meet and no solution satisfy the constraint, we select the best instance as overall deadline can be met.

2.7 PEFT based genetic algorithm(PGA):

With the objective of minimizing the execution cost while keeping the execution time under deadline, PGA[15] is based on Genetic Algorithm(GA) which uses PEFT algorithm to generate an optimal solution. PEFT is divided in two stages: prioritizing and resource selection. It can do the forecasting of child tasks effect on current task due to which makespan can be improved. For this it uses OCT. Optimistic Cost Table(OCT) is a matrix with rows representing the tasks and column representing resources. Here cost is calculated using backward approach which make OCT a look ahead feature, providing the cost of current task’s child execution until it reaches end node. In prioritizing phase tasks are prioritized on the basis of cumulative OCT in decreasing order. For resource selection objective Optimistic EFT(OEFT) is calculated. Its aim is to shorten the finish time of ahead tasks. PGA algorithm consist of three parts: Encoding, Fitness calculation and Genetic operators. In encoding ,a chromosome represents a solution whose length represents the number of tasks and value of gene represents VM task it is assigned to. Fitness function deals with the objective discussed earlier. Either the one satisfying the objective is selected or one with minimum objective violation. In genetic operators Binary tournament selection is used to find the fittest chromosome and copy good ones from a generation to another.

2.8 Frequency Based Optimization(FBO):

Algorithm[16] aims at choosing appropriate frequency so as to minimize the cost while meeting the deadline constraint. Cost of each resource is based on the frequency allocated to a task. In this algorithm three types of pricing model has been used to meet the goals: linear, sub-linear, super-linear and the total cost of executing the workflow is calculated by subtracting the pricing model cost value from the execution time value of running tasks on maximum frequency. Two variants of the algorithm are proposed. In first variant a makespan based scheduling[17] algorithm such as HEFT is used to generate the initial schedule. HEFT assigns tasks of workflow to maximum frequencies so as to ensure that deadline will be attained not considering the cost parameter. In second variant of algorithm the task, resource and frequency allocation keeps on changing iteratively so as to minimize the cost while ensuring about deadline too. For the second variant a weight table is prepared which contains values of all combinations of task, resource and cpu frequency allocations. This reallocation continues until execution cost is reduced without violating deadline. A point to remember is that algorithm can try for reassignment of task to a resource and frequency pair only once. So when no untried combination will be left, the algorithm will terminate.

2.9 Probabilistic Listing(PROLIS):

ProLis[18] also aims to provide a cost effective schedule under deadline constraint. It distributes the deadline of workflow to individual tasks. Then it performs tasks prioritization followed by task selection. ProLis performs deadline distribution on the basis of Probabilistic Upward Rank. Upward Rank of a task is the longest path from it to the end task. Sub deadline is assigned to a task in proportion to the longest path from entry node to current task. Differentiation factor in Probabilistic upward rank and Upward rank is inclusion of a Boolean variable to ensure that tasks having larger transmission time are assigned on same resources. For task ordering algorithm uses the probabilistic upward rank methodology since data transmission in this may become zero. In the service selection step, initially such service is searched that can minimize the cost under deadline constraint. If none of such service exist than objective is narrowed down to minimization of execution time.

2.10 LACO:

LACO make use of Ant Colony Optimization (ACO) to redefine task ordering step of ProLis. LACO also largely depends on heuristic and pheromone trail. Pheromone can be understood as an aspire to select a task just after another task. Same as ProLis, LACO is divided in three parts. Deadline distribution, task ordering and task selection, but for it's organized task list it depends on pheromone (unlike ProLis which depends on probabilistic upward rank) and heuristic information. In order to ensure that the proposed solution of algorithm doesn't contain violation of dependencies between tasks, it adapts Kahn's algorithm for topological sorted ordered schedule. In order to meet the problem of deadline violation algorithm make use of improved independent optimization method, where the deadline is relaxed in proportion to the iteration number of LACO and then comparison between solution meeting the relaxed deadline is done on the basis of some parameter. With the increase in the iteration number, the relaxation goes on deceasing until the number reaches to terminating iteration number. In each iteration after local best solution is figured, pheromone trail is updated. At last global best solution is returned.

2.11 Iaas Cloud- Partial Critical Paths(ICPCP):

Most important concepts of ICPCP[19] are critical path, assigned node and critical parents. Assigned node speaks for a node whose service has been selected. Critical parent of a task is an unassigned parent of the task whose data reaches to the current task most lately. Partial Critical Path of workflow will be empty if task doesn't have any unassigned parents. Otherwise the path contains the critical parent of the task as well as partial critical path of the parent if it exists. In algorithm initially some parameters are calculated and entry and exit tasks are assigned followed by calling another algorithm for assigning parents. This called algorithm takes an assigned node as input and allocate all the unassigned parents of input node to some service before the start time of input node. Through this scheduling[20] strategy , the algorithm finds all critical paths of the workflow. Once done another procedure is approached for path assignment to services. This new procedure schedules received critical paths on applicable instances that can complete each task's execution before it's latest finish time with minimum price.

2.12 Iaas Cloud Partial Critical Path with Deadline Distribution(ICPCPD2):

ICPCPD2 works in two phases and has three main concepts to depend on which are Partial Critical Path, critical parent and assigned node. While the definition of first two concepts remains same as in ICPCP, assigned node here stands for a node to which sub-deadline has already been assigned. As mentioned earlier the two phases are: Deadline distribution and Planning. In deadline distribution phase, complete workflow deadline is distributed over individual tasks followed by calling separate procedure for parents assignment. This procedure is completely same as the one discussed in ICPCP. Once assignment is done the execution shifts to another procedure for assigning path and this is where difference exists in ICPCP and ICPCPD2 algorithms. Unlike ICPCP, In ICPCPD2 , the path assigning procedure performs assignment of sub-deadlines to all the unallocated parents of input node. For this it distributes the deadline of path among tasks which are a part of path in proportion to their minimum execution time. In the planning phase exists another procedure which plans and assigns each task to cheapest applicable instance which can make task to finish it's execution before the sub-deadline.

.

CHAPTER 3

THE PROPOSED WORK

3.1 Problem Denotation

Workflow scheduling mainly aims at assigning resources to tasks and sequencing their order of execution while satisfying the qos constraints. This dissertation presents analysis of workflow scheduling algorithms to mainly address two problems, cost minimization and deadline constraint during workflow execution.

Workflow scheduling is broadly divided in two steps: as per requirement of tasks ,selecting a set of resources from the available pool and their distribution. Next step is generating a schedule using desired strategy and mapping of tasks on these resources keeping in mind the qos constraints.

The problem which is discussed through these algorithms is to perform the complete execution of workflow application (scheduling) within user given deadline (time) and also in the minimized budget (cost).

3.2 Overview

In bee hives, foraging honeybees give information to other bee's aboutthe position of the food source they have visited. A prospective searcher bee starts her career as an unemployed naive worker, that is, she has as no information of a food source in the field yet. She can start search for a source and thus become a scout (explorer). The initiation to fly out and start foraging is not due to following a waggle dance but due to some unknown internal, motivational factor or perhaps to some unknown external cue. Alternatively, a bee can start searching for a source as a response to attending a waggle dance and thus becomes a recruit. So the distinction between a recruit and a scout is that the recruit has stored estimated positional information in her memory, whereas the scout has not. As soon as a bee finds a source, it registers the essentials of this source in its memory and starts exploiting it, the bee is then an employed forager (exploiter)[8].

The proposed LBA_HB is completely inspired by the natural foraging behavior of honey bees. The allocated task updates the remaining tasks about the VM status in a manner similar to the bees finding an abundant food source, updating the other bees in the bee hive through its waggle dance. This task updates the status of the VM availability and the load of the VMs.

3.3 Bee Colony Based Load Balancing Algorithm

The proposed technique utilizes the foraging behavior of honey bees for load balancing across VMs. Here the honey bee foraging behaviour is mapped to cloud environment to perform load balancing.

Table 3.1: How To Map Bee Colony With Cloud Environment

Honey Bee Hive	Cloud Environment
Honey bee	Task(cloudlet)
Food Source	VM
Honey bee foraging a food source	Loading of a task to a VM
Honey bee getting depleted at a food source	VM is overloaded
Foraging bee finding a new food source	Removed task will be scheduling to an under loaded VM

Cloudlets (Tasks) in the cloud environment can be treated as the honey bees. As the honey bees forage for food source, cloudlets will be assigned in VMs for execution. The VM has a particular capacity. Sometimes some VMs may be overloaded and others will be under loaded.

In this case in order to accomplish better performance load balancing is needed. So when a VM is overloaded with multiple tasks then some tasks are removed from that particular VM and it is assigned to an underloaded VM. The removing task will be selected based on priority. The lowest priority tasks will be selected for transferring. This is similar as honey bees depleted from the food source.

Figure 3.1 represents the proposed load balancing architecture. CIS (Cloud Information Service) is the repository which contains the resources available in the cloud. CIS is specifically a registry of Datacenters. When a datacenter is created it will register to CIS. Datacenter has some specific characteristics. Many hosts are present in a datacenter. Characteristics of datacenter are actually the characteristics of hosts. Hosts have specific processing elements (PEs), RAM and bandwidth characteristics. Cloud computing works based on the concept of virtualization. The host is virtualized into number of VMs. VMs also has some specific characteristics like hosts.

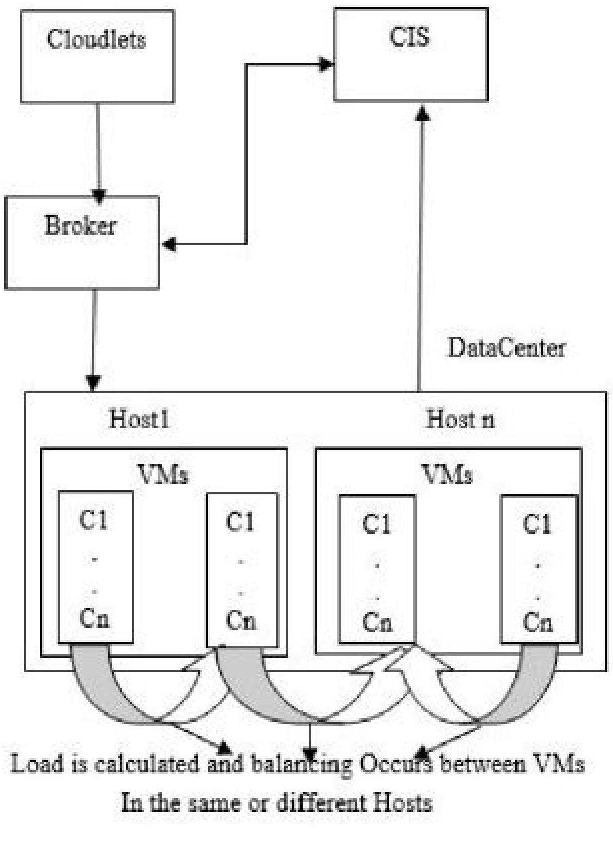

Cloudlets
CIS
Broker
DataCenter
Host1
Host n
VMs
VMs
C1
Cn
Load is calculated and balancing Occurs between VMs
In the same or different Hosts

Figure 3.1: Load balancing architecture[41]

At the initial stage datacenter broker interacts with the CIS to collect information about the resources specifically about the datacenters. Tasks in cloud environments are represented as cloudlets in CloudSim framework. When these cloudlets are arrived at the broker, broker will submit it to the VMs in a datacenter for execution. During task execution when certain VMs are overloaded and some are sitting idle or doing very little work, tasks i.e. cloudlets from overloaded VMs are removed and assigned to under loaded VMs for efficient execution. Load balancing approach based on bee colony can be divided into following modules:

- Finding the load on a VM
- Load Balancing Decision
- VM Grouping
- Task Transfer

3.4 Pseudo Code Of LBA_HB Algorithm

```
Input: List of available host(lh), List of available Vm(lv)

Output: VM(j)

// returns number of (i) hosts with minimum processing time (i is the total number of hosts)

Let i =-1 && minimum processing time(mPT) = maximum value of integer

For each host in lh
  If host is available then
    If processing time of host(PTh) < mPT then
    minPT= PTh
    i= number of current host
    End if
  End if
End for
```

```
// returns number of (j) VM which has minimum number of tasks (j is the total number of
specified VM)
Let j=-1 && minimum count of tasks(mC) = maximum value of integer
For each Vm in lv(i) // that is for each Vm present in selected host
  If Vm is available then
    If count required in Vm(cV) < mC then
    mC = cV
    End if
  End if
End for
If j==-1 then
Add coming task in waiting queue until one Vm become available.
  Else
  Allocate task to Vm.
  End else
Update allocated information // that is current processing time of host and Vm and their
availability.
After task execution get's completed , deallocate the task from Vm.
Update the information again to notify the changes.
End if
```

3.5 Flow Chart

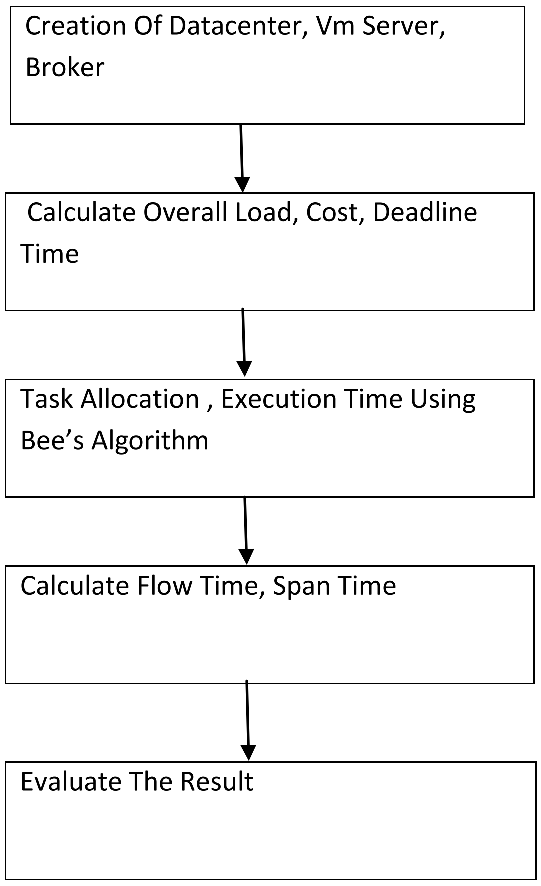

Creation Of Datacenter, Vm Server, Broker
Calculate Overall Load, Cost, Deadline Time
Task Allocation , Execution Time Using Bee’s Algorithm
Calculate Flow Time, Span Time
Evaluate The Result

Fig 3.2 :FlowChart

A **flowchart** is a type of diagram that represents an algorithm or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows.This diagrammatic representation illustrates a solution to a given problem. Process operations are represented in these boxes,and arrows; rather, they are implied by the sequencing of operations. Flowcharts are used in analyzing,designing, documenting or managing a process or program in various fields.

3.6 Implementation Modules

- Host and Vm Creation
- Load Identification

- Job allocation
- Evaluation

3.6.1 Host and VM Creation

In this unit use to the well-known classification of planning in spread calculating structures by CA savant and Kohl, from the point of view of the featureof the clarifications a scheduler is able to form, any forecast system can be categorized into prime or sub-optimal. The previous describes preparation procedures that, built on whole info about the state of the circulated atmosphere (e.g., hardware capabilities and load) as well as source wants (e.g., time required by each job on each computing resource),bring out optimum job-resource mappings. When this evidence is not obtainable, or the interval to calculate a answer is impossible, sub-optimal procedures are used as an alternative.

3.6.2 Load Identification

In this module is Clouds, as any other distributed computing environment, is not free from the problem of accurately estimate aspects such as job duration. Another aspect that makes this problem more difficult is multi-tenancy, a distinguishing feature of Clouds by which several users and hence their (potentially heterogeneous) jobs are served at the same time via the illusion of several logic infrastructures that run on the same physical hardware. This also poses challenges when estimating resource load. Indeed, optimal and sub-optimal-approximate algorithms such as those based on graph or queue theory need accurate information beforehand to perform correctly, and therefore in general heuristic algorithms are preferred. In the context of our work, we are dealing with highly multi-tenant Clouds where jobs come from different Cloud users (people performing multi-domain PSE experiments) and their duration cannot be predicted.

3.6.3 Job allocation

To decide which job should be allocated to which Vm, we need to collect information of last allocate and deallocate. Process is similar to which honey bee should go in search for which food source out of many available which depends on whether that food source contains honey or not. The information received contains two data. Threshold information which specifies availability of a host. Same kind of availability check is done in Vm selection before allocating a job. Another information is priority information which contains current load value. At host level , load indicator stands for processing time and at

Vm level it indicates number of tasks. So during allocation first of all host selection is done which on priority information , that is host with minimum level. Then comes suitable Vm selection. In this, task should be allocated to which of the available Vm's of selected host depends on number of tasks already allocated to that Vm.

3.6.4 Evaluation

In order to assess the effectiveness of our proposal for executing PSEs on Clouds, we have processed a real case study for solving a very well-known benchmark problem proposed in the literature, see for instance. Broadly, the experimental methodology involved two steps. First, we executed the problem in a single machine by varying an individual problem parameter by using a finite element software, which allowed us to gather real job data, i.e., processing times and input/output file data sizes By means of the generated job data, we instantiated the Clouds simulation toolkit, which is explained. Lastly, the obtained results regarding the performance of our proposal compared with some Cloud scheduling alternatives are reported.

3.7 Metrics Of Proposed Algorithm

Cloud computing consist of a number of data centers. Let each data center consists of *n* hosts(say) and each host consist of *m* VM(say). Then:

- The processing time of host(i):

$$PT_{host}(i) = \frac{TL_{host}(i)}{NPR_{host}(i) * SPR_{host}(i)} = \frac{\sum_{k=1}^{x1} REQ_{length}(k)}{NPR_{host}(i) * SPR_{host}(i)} \tag{3.1}$$

  Where $TL_{host}(i)$ is total length of tasks submitted to host(i), $NPR_{host}(i)$ is number of processors in host(i), $SPR_{host}(i)$ is the processor speed of host(i), $REQ_{length}(k)$ is the length of request number k.

- Average processing time of all hosts:

$$PT_{Avghost} = \frac{1}{n}\sum_{i=1}^{n} PT_{host}(i) \tag{3.2}$$

- Processing time of VM(j):

$$PT_{VM}(j) = \frac{TL_{VM}(j)}{NPR_{VM}(j) * SPR_{VM}(j)} = \frac{\sum_{k=1}^{x2} REQ_{length}(k)}{NPR_{VM}(j) * SPR_{VM}(j)} \tag{3.3}$$

  Where $TL_{VM}(j)$ is total length of tasks submitted to VM(j), $NPR_{VM}(j)$ is number of processors in VM(j), $SPR_{VM}(j)$ is the processor speed of VM(j), $REQ_{length}(k)$ is the length of request number k.

- Average processing time of all VM's:

$$PT_{AvgVM} = \frac{1}{m} \sum_{j=1}^{m} PT_{VM}(j) \tag{3.4}$$

CHAPTER 4

EXPERIMENTAL RESULTS

In this section we present the result of our existing and proposed work. The performance of both the algorithms are shown using their flow time and span time parameters. In order to show comparison between the two already discussed work we have made use of a tabular graph showing performance bar in terms of cost.

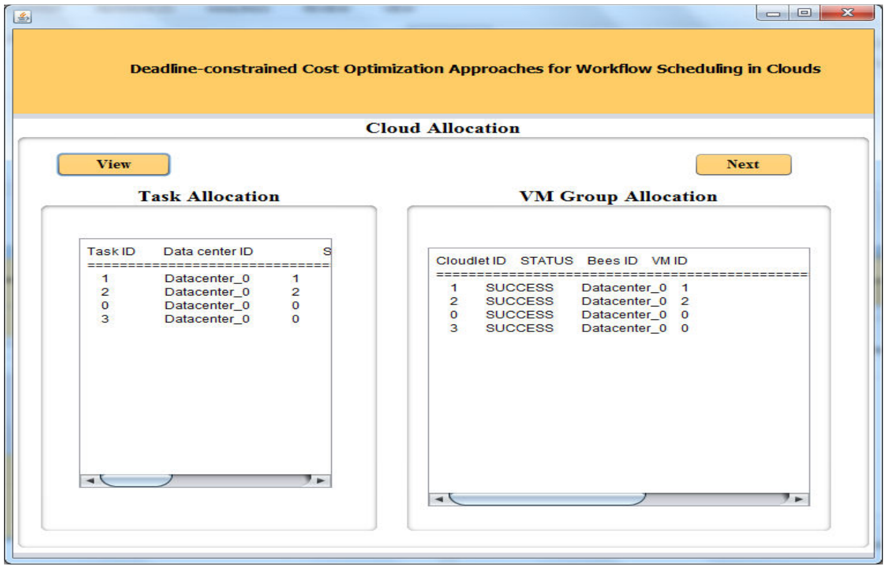
This section is organized as follows: Section 4.1 contains Honey Bee algorithm performance evaluation. Section 4.2 contains L-ACO algorithm performance evaluation and comparison graph between two algorithms.

4.1 Deadline-constrained Cost Optimization Approaches for Workflow Scheduling in Clouds(Honey Bee)

The below diagram represents:

- Task Allocation
- VM Group Allocation

tasks of Honey Bee Algorithm for cost optimization of workflow scheduling in clouds.


Deadline-constrained Cost Optimization Approaches for Workflow Scheduling in Clouds
Cloud Allocation
View
Next
Task Allocation
VM Group Allocation

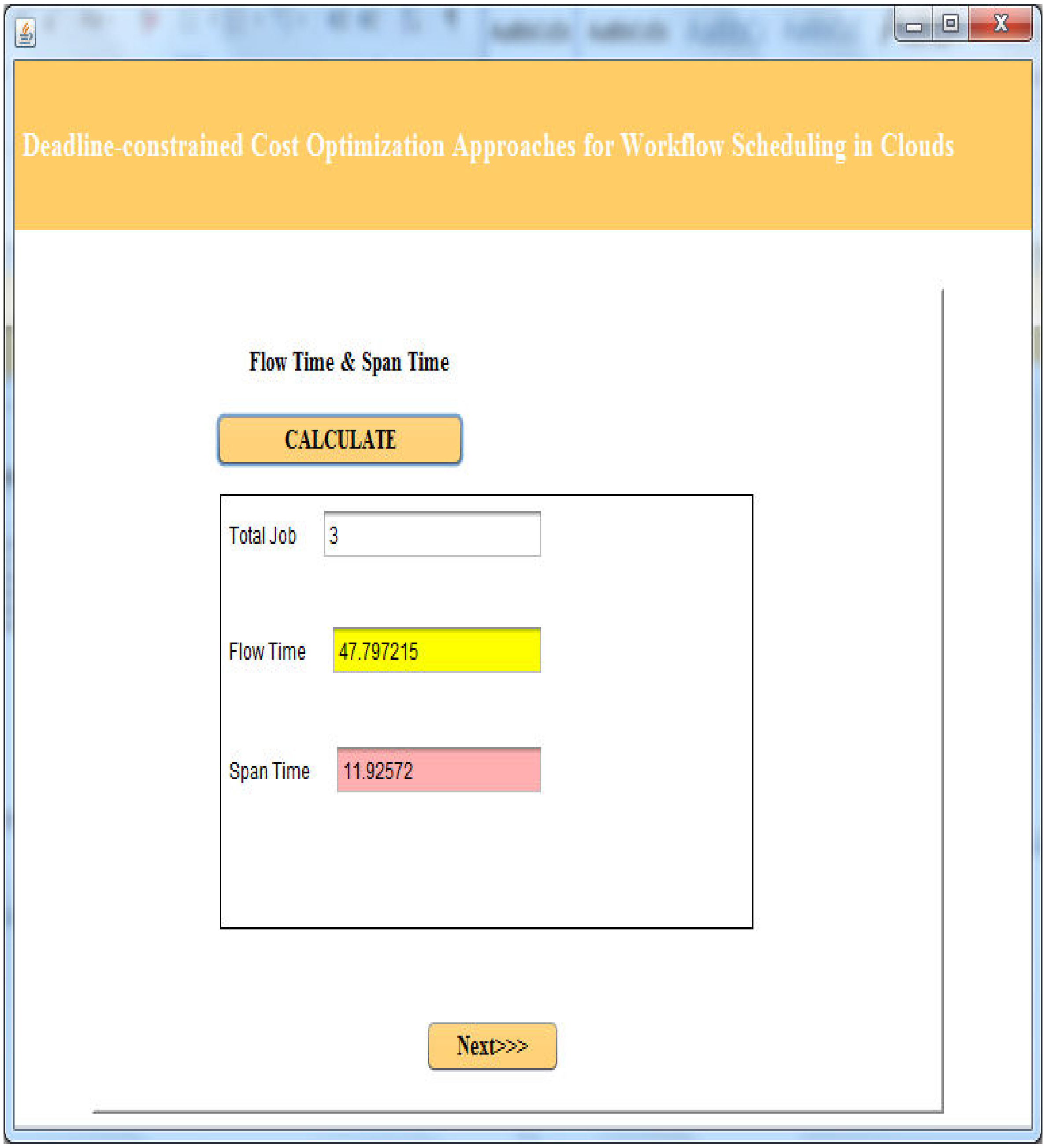
```
Task ID     Data center ID          S
================================
   1        Datacenter_0        1
   2        Datacenter_0        2
   0        Datacenter_0        0
   3        Datacenter_0        0
```

```
Cloudlet ID   STATUS    Bees ID    VM ID
==============================================
  1     SUCCESS     Datacenter_0   1
  2     SUCCESS     Datacenter_0   2
  0     SUCCESS     Datacenter_0   0
  3     SUCCESS     Datacenter_0   0
```

Fig 4.1: Cloud Allocation for Workflow Scheduling in Clouds (Honey Bee)

Here efficiency of Honey Bee algorithm is depicted in terms of :

- Total No. of Jobs
- Flow Time
- Span Time


Deadline-constrained Cost Optimization Approaches for Workflow Scheduling in Clouds

Flow Time & Span Time

Total Job	3
Flow Time	47.797215
Span Time	11.92572

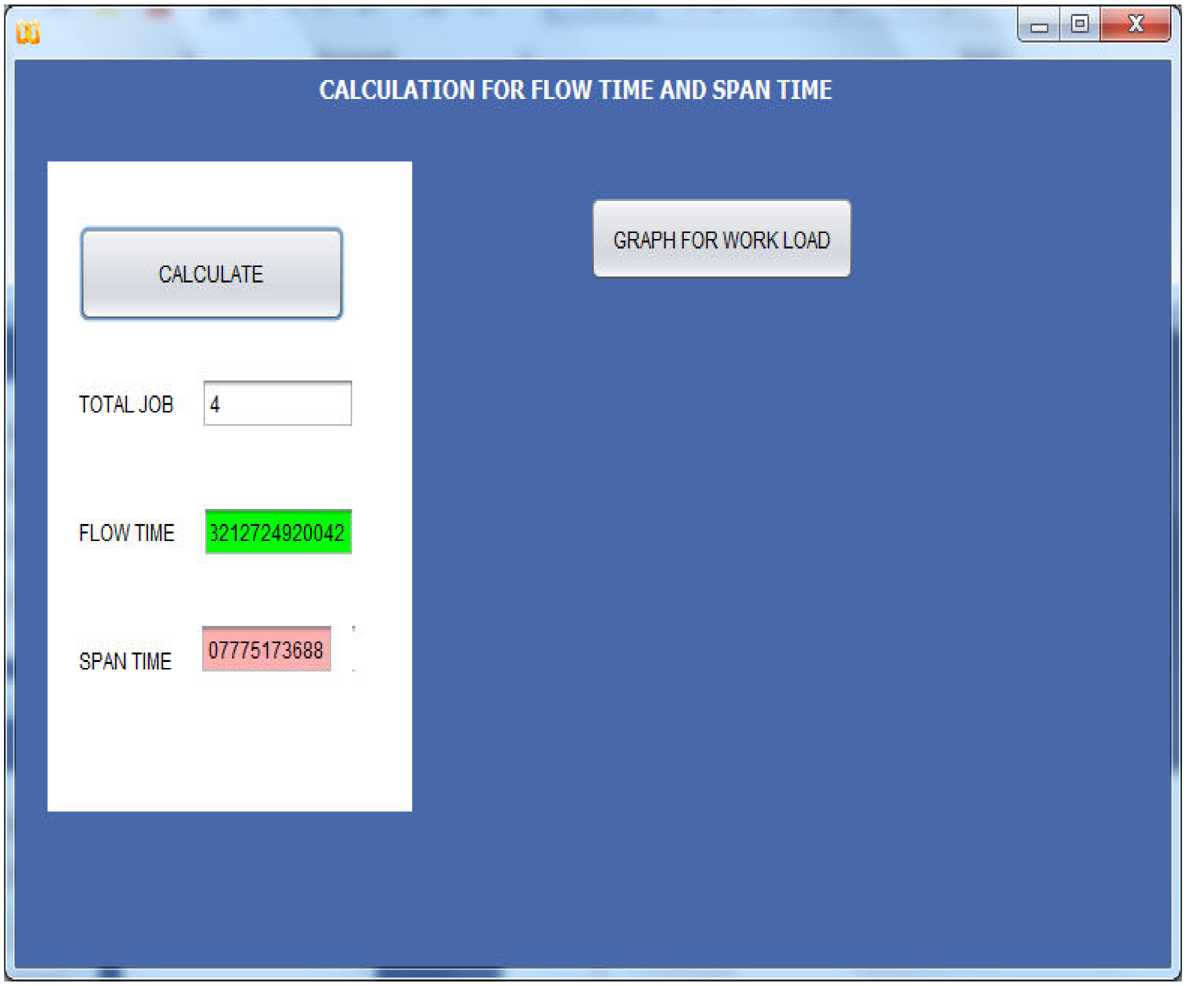
Fig 4.2: Cost Optimization for Workflow Scheduling Using Honey Bee

4.2 Cost Optimization Approaches for Workflow Scheduling in Clouds(L-ACO ALGORITHM)

Here efficiency of L-ACO algorithm is depicted in terms of :

- Total No. of Jobs
- Flow Time
- Span Time

and also a comparison chart is shown containing Honey Bee and L-ACO algorithms cost oriented performance



TOTAL JOB 4

FLOW TIME 3212724920042

SPAN TIME 07775173688

Fig 4.3: Cost Optimization for Workflow Scheduling Using LACO

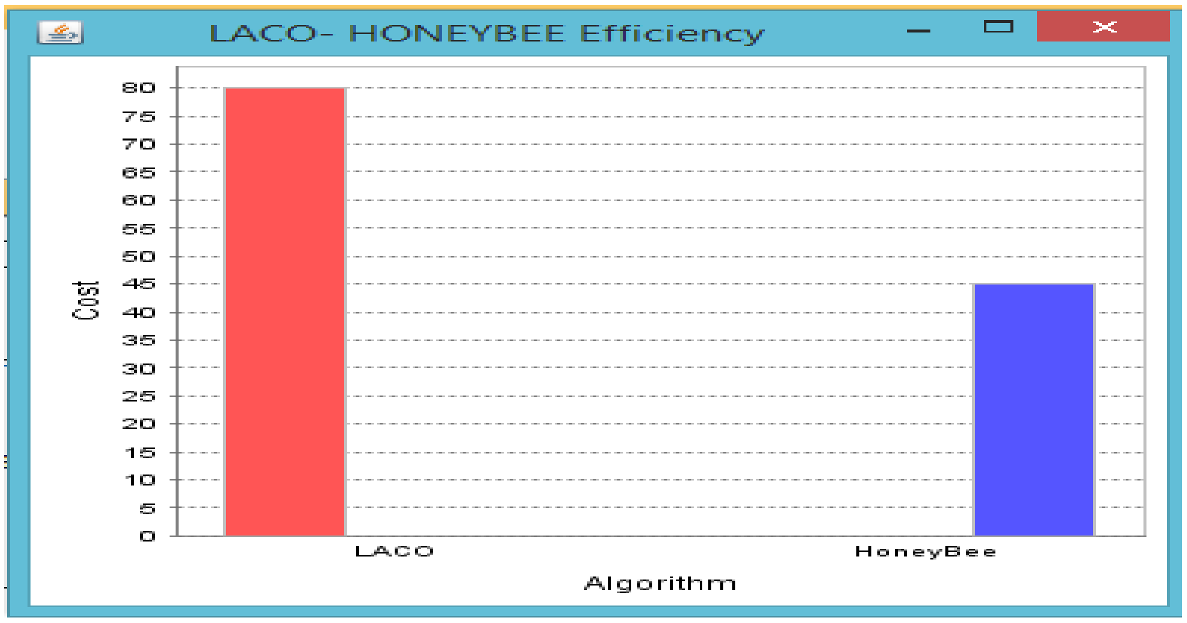


Fig 4.4 LACO - Honeybee Cost Based Efficiency Comparison.

CHAPTER 5

SOFTWARE REQUIREMENT AND TESTING PROCESS

5.1 Software Description

5.1.1 Java

Java may be a artificial language originally developed by James goose at Sun Microsystems (now a subsidiary of Oracle Corporation) and free in 1995 as a core part of Sun Microsystems' Java platform. The language derives a lot of its syntax from C and C++ however incorporates a less complicated object model and fewer low-level facilities.Java applications are usually compiled to computer memory unit code (class file) which will run on any Java Virtual Machine (JVM) notwithstanding pc design. Java may be a all-purpose, concurrent, class-based, object-oriented language that's specifically designed to possess as few implementation dependencies as potential. it's supposed to let application developers "write once, run anyplace." Java is presently one in every of the foremost standard programming languages in use, notably for client-server internet applications.

5.1.2 Net Beans

The Net Beans Platform may be a reusable framework for simplifying the event of Java Swing desktop applications. world wide web Beans IDE bundle for Java SE contains what's required to begin developing web Beans plug-in and web Beans Platform based mostly applications; no further SDK is needed.

Applications will install modules dynamically. Any application will embrace the Update Center module to permit users of the appliance to transfer digitally-signed upgrades and new options directly into the running application.

The platform offers reusable services common to desktop applications, allowing developers to focus on the logic specific to their application. Among the features of the platform are:

- User interface management (e.g. menus and toolbars)
- User settings management
- Storage management (saving and loading any kind of data)
- Window management
- Wizard framework (supports step-by-step dialogs)
- Net Beans Visual Library
- Integrated Development Tools

5.2 Testing

Project testing is the stage of implementation, which aimed at ensuring that project works accurately and efficiently. Testing is the process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an error. A successful test is one that answers a yet undiscovered error.

Testing is vital for the success of a project. Project testing makes a logical assumption that if all parts of the project are correct, the goal will be successfully achieved. Any engineered product can be tested in one of the following ways. Knowing the specified function that a product has been designed to from, test can be conducted to demonstrate each function is fully operational. Knowing the internal working of a product, tests can be conducted to ensure that the internal operation of the product performs according to the specification and all internal components have been adequately exercised.

5.2.1 Unit Testing:

Unit testing is the testing of each module and the integration of the overall project is done. Unit testing becomes verification efforts on the smallest unit of design in the module. This is also known as 'module testing'. The modules of the project are tested separately. This testing is carried out during the programming itself. In this testing step, each model is found to be working satisfactorily as regard to the expected output from the module. There are some validation checks for the fields. For example, the validation check is done for verifying the data given by the user where both format and validity of the data entered is included. It is very easy to find error and debug the system.

5.2.2 Integration Testing:

Data can be lost across an interface, one module can have an adverse effect on the other sub function, when combined, may not produce the desired major function. Integrated testing is systematic testing that can be done with sample data. The need for the integrated test is to find the overall project performance. There are two types of integration testing. They are:

i) Top-down integration testing.

ii) Bottom-up integration testing

CHAPTER 6
CONCLUSION AND FUTURE WORK

6.1 Conclusion

This thesis proposed a load balancing algorithm for cloud environment based on foraging behaviour of honey bees. This algorithm removes the tasks from overloaded VMs and submitted it to the most suitable under loaded VM. It not only balances the load, but also considers the priorities of tasks in the waiting queues of VMs. The task with least priority is selected for migration. So no tasks are needed to wait longer time in order to get processed. The experimental result shows that, the proposed algorithm reduces the makespan, degree of imbalance and number of task migrations that results in better QoS for end users. This dissertation considered priority as the QoS parameter.

6.2 Future Work

This thesis has proposed a task deployment approach L-ACO for the long-term load balancing effect and it has employed a heuristic idea based on Bayes theorem and the clustering process. L-ACO first has narrowed down the search scope by comparing performance values. Then, L-ACO has utilized Bayes theorem to obtain the posteriori probe-ability values of all candidate physical hosts. Finally, LB-Chas combined probability theorem and the clustering ideate pick out the optimal hosts set, where these physical hosts have the most remaining computing power currently, for deploying and executing tasks by selecting the physical host with the maximum posteriori probability value as the clustering center and thus to achieve the load balancing effect from the long-term perspective. Simulation experiments demonstrate that the proposed L-ACO approach can deploy the instant tasks quickly and effectively in cloud data centers. It makes cloud data centers achieve a long-term load balancing of the whole network.

We continued the same process with the usage of HoneyBee algorithm. Experiment and analysis confirm the effectiveness of our schemes and design.

In future work, we are interested in how to make use of clustering and replication techniques to improve the proposed algorithms and we also plan to test them in public clouds. Besides, we intend to take into account the spot instance-based cloud services for workflow scheduling, which are an effective way to reduce costs when the deadline is loose. However, they introduce further challenges because of pricing dynamics and the interruption caused by bidding failures.

We are also interested in improving the migration technique in Honey Bee. Currently it's not as efficient in proposed LBA_HB since it checks VM value variation during task allocation in independent tasks. However in future migration technique can be implemented keeping in mind task allocation for a group of dependent tasks.

References

[1] DervisKaraboga· BahriyeBasturk, “A powerful and efficientalgorithm for numerical function optimization: artificial bee colony(ABC) algorithm”, Springer, J Glob Optim (2007) 39:459–471.

[2] Shridhar G Damanaland, G. Ram Mahana Reddy, “Optimal LoadBalancing in Cloud Computing By Efficient Utilization of VirtualMachines”, IEEE Sixth International Conference on CommunicationSystems and Networks (COMSNETS), 2014, pp. 1- 4.

[3] Soni. G, Kalra. M, “A Novel Approach for Load Balancing in CloudData Center”, IEEE International Conference on Advance ComputingConference (IACC), 2014, pp. 807-812.

[4] Santanu Dam, GopaMandal, KousikDasgupta, Paramartha Dutta,"An Ant Colony Based Load Balancing Strategy in CloudComputing", Vol. 28, 2014, pp. 403-413.

[5] Liuhua Chen, HaiyingShen and Karan Sapra, “RIAL: ResourceIntensity Aware Load Balancing in Clouds”, IEEE Conference onComputer Communications (INFOCOM 2014), pp. 1294-1302.

[6] GaochaoXu, Junjie Pang, and Xiaodong Fu, “A Load BalancingModel Based on Cloud Partitioning for the Public Cloud”, IEEE J.Tsinghua Science and Technology, 2013, pp. 34-39.

[7] M. Randles, D. Lamb, and A. Taleb-Bendiab, “A comparative study into distributed load balancing algorithms for cloud computing”, in Proc. IEEE 24th International Conference on Advanced Information Networking and Applications, Perth, Australia, 2010, pp. 551-556.

[8] Jing Yao, Ju-hou He, “Load Balancing Strategy of Cloud Computing based on Artificial Bee Algorithm”, IEEE 8th International Conference on Computing Technology and Information Management(ICCM), 2012, pp. 185-189.

[9] Pooja Samal, “Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing” (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 4 (3), 2013,pp. 416-419.

[10] Remesh Babu, K.R., Philip Samuel, "Virtual Machine Placement for Improved Quality in IaaS Cloud", IEEE Fourth International Conference on Advances in Computing and Communications(ICACC), 2014, pp. 190-194.

[11] Domanal, Shridhar G. Reddy, G. Ram Mohana, “Load Balancing in Cloud Computing Using Modified Throttled Algorithm” IEEE International Conference on Cloud Computing in Emerging Markets(CCEM), 2013, pp. 1-5.

[12] Ajit. M., Vidya. G., “VM Level Load Balancing in Cloud Environment”, IEEE Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), 2013, pp.1-5.

[13] H. Topcuoglu, S. Hariri, and M.-y. Wu, “Performance-effective and low-complexity task scheduling for heterogeneous computing,” IEEE Transactions on Parallel and Distributed Systems, vol. 13, no.3, pp. 260-274, 2002.

[14] H. Arabnejad, and J. G. Barbosa, “List scheduling algorithm for heterogeneous systems by an optimistic cost table,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 3, pp.682-694, 2014.

[15] S. Abrishami, M. Naghibzadeh, and D. H. Epema, “Cost-driven scheduling of grid workflows using partial critical paths,” IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 8, pp.1400-1414, 2012.

[16] R. N. Calheiros, and R. Buyya, “Meeting deadlines of scientific workflows in public clouds with tasks replication,” IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 7, pp.1787-1796, 2014.

[17] M. Malawski et al., “Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization,”Scientific Programming, vol. 2015, pp. 5, 2015.

[18] I. Pietri, and R. Sakellariou, “Cost-Efficient CPU Provisioning for Scientific Workflows on Clouds,” in International Conference on Grid Economics and Business Models, 2015, pp. 49-64.

[19] W. N. Chen, and J. Zhang, “An Ant Colony Optimization Approach to a Grid Workflow Scheduling Problem With Various QoS Requirements,” IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 39, no. 1, pp.29-43, 2009.

[20] S.-T. Lo et al., “Multiprocessor system scheduling with precedence and resource constraints using an enhanced ant colony system,”Expert Systems with Applications, vol. 34, no. 3, pp. 2071-2081,2008.

[21] M. A. Tawfeek et al., “Cloud task scheduling based on ant colony optimization,” in Computer Engineering & Systems (ICCES), 20138th International Conference on, 2013, pp. 64-69.

[22] W. Zheng, and R. Sakellariou, “Budget-Deadline Constrained Workflow Planning for Admission Control,” Journal of Grid Computing, vol. 11, no. 4, pp. 633-651, 2013.

[23] H. Arabnejad, and J. G. Barbosa, “A budget constrained scheduling algorithm for workflow applications,” Journal of Grid Computing, vol. 12, no. 4, pp. 665-679, 2014.

[24] C. Q. Wu et al., “End-to-End Delay Minimization for Scientific Workflows in Clouds under Budget Constraint,” IEEE Transactions on Cloud Computing, vol. 3, no. 2, pp. 169-181, 2015.

[25] F. Wu et al., “PCP-B2: Partial critical path budget balanced scheduling algorithms for scientific workflow applications,” Future Generation Computer Systems, vol. 60, pp. 22-34, 2016.

[26] R. Sakellariou et al., "Scheduling Workflows with Budget Constraints," Integrated Research in GRID Computing: Core GRID Integration Workshop, pp. 189-202, Boston, MA: Springer US, 2007.

[27] Z. Zhu et al., “Evolutionary multi-objective workflow scheduling in cloud,” IEEE Transactions on Parallel and Distributed Systems, vol.27, no. 5, pp. 1344-1357, 2016.

[28] J. J. Durillo, H. M. Fard, and R. Prodan, “Moheft: A multi-objective list-based method for workflow scheduling,” in Cloud Computing Technology and Science (Cloud Com), 2012 IEEE 4th International Conference on, 2012, pp. 185-192.

[29] J. J. Durillo et al., “Bi-objective Workflow Scheduling in Production Clouds: Early Simulation Results and Outlook,” in Proceedings of the First International Workshop on Sustainable Ultrascale Computing Systems, Porto, 2014, pp.

[30] J. J. Durillo, R. Prodan, and J. G. Barbosa, “Pareto trade off scheduling of workflows on federated commercial Clouds,” Simulation Modelling Practice and Theory, vol. 58, pp. 95-111,2015.

[31] M. Malawski et al., “Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds,” Future Generation Computer Systems, vol. 48, pp. 1-18, 2015.

[32] "Amazon Elastic Compute Cloud," 2016;http://aws.amazon.com/ec2/.

[33] M. Zotkiewicz et al., “Minimum Dependencies Energy-Efficient Scheduling in Data Centers,” IEEE Transactions on Parallel and Distributed Systems, vol. PP, no. 99, pp. 1-1, 2016.

[34] M. Dorigo, “Ant Colony Optimization: The MIT Press, 2004”.

[35] T. H. Cormen et al., “Introduction to Algorithms, Third Edition,” pp.1297–1305, 2009.

[36] E. Mezura-Montes, and C. A. CoelloCoello, “Constraint-handling innature-inspired numerical optimization: Past, present and future,”Swarm and Evolutionary Computation, vol. 1, no. 4, pp. 173-194,2011.

[37] T. Takahama, and S. Sakai, “Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation,” in IEEE Congress on Evolutionary Computation, 2010,pp. 1-9.

[38] T. Stützle, and H. H. Hoos, “MAX–MIN Ant System,” Future Generation Computer Systems, vol. 16, no. 8, pp. 889-914, 2000.

[39] G. Juve et al., “Characterizing and profiling scientific workflows,” Future Generation Computer Systems, vol. 29, no. 3, pp. 682-692,2013.

[40] T. Stützle et al., “Parameter adaptation in ant colony optimization,” Autonomous Search, pp. 191-215, 2010.

[41] Walaa Hashem, Heba Nashaat and Rawya Rizk, "Honey Bee Based Load Balancing in Cloud Computing," KSII Transactions on Internet and Information Systems, vol. 11, no. 12, pp. 5694-5711, 2017. DOI: 10.3837/tiis.2017.12.001

[42] Jasraj Meena, Shubham Jain, “Workflow Scheduling Algorithms in Cloud Computing: An Analysis, Analogy and Provocations,” Conference on Cloud Computing, 2018.