Journal of Computer Science 8 (10): 1615-1626, 2012
ISSN 1549-3636

Dynamic Scheduling for Cloud Reliability using Transportation Problem

[1]Senthil Kumar, S.K. and [2]P. Balasubramanie
[1]Department of Computer Science and Engineering,
Jansons Institute of Technology, Coimbatore, Tamil Nadu, India
[2]Department of Computer Science and
Engineering Kongu Engineering College, Erode, Tamil Nadu, India

Abstract: Problem statement: Cloud is purely a dynamic environment and the existing task scheduling algorithms are mostly static and considered various parameters like time, cost, make span, speed, scalability, throughput, resource utilization, scheduling success rate and so on. Available scheduling algorithms are mostly heuristic in nature and more complex, time consuming and does not consider reliability and availability of the cloud computing environment. Therefore there is a need to implement a scheduling algorithm that can improve the availability and reliability in cloud environment. **Approach:** We propose a new algorithm using modified linear programming problem transportation based task scheduling and resource allocation for decentralized dynamic cloud computing. The Main objective is to improve the reliability of cloud computing environment by considering the resources available and it's working status of each Cluster periodically and maximizes the profit for the cloud providers by minimizing the total cost for scheduling, allocation and execution cost and minimizing total turn-around, total waiting time and total execution time. Our proposed algorithm also utilizes task historical values such as past success rate, failure rate of task in each Cluster and previous execution time and total cost for various Clusters for each task from Task Info Container (TFC) for tasks scheduling resource allocation for near future. **Results:** Our approach TP Scheduling (Transpotation Problem based) responded for various tasks assigned by clients in poisson arrival pattern and achieved the improved reliability in dynamic decentralized cloud environment. **Conclusion:** With our proposed TP Scheduling algorithn we improve the Reliability of the decentralized dynamic cloud computing.

Key words: Transportation problem, cloud reliability, profit maximization, dynamic decentralized scheduling, Task Info Container (TFC)

INTRODUCTION

Cloud computing refers to Internet based development and utilization of computer technology and hence, cloud computing can be described as a model of Internet-based computing and a subscription-based service where you can obtain networked storage space and computer resources and so on. Cloud Computing, dynamically scalable (and mostly virtualized) resources are provided as a service over the Internet. With the promotion of the world's leading companies, cloud computing is attracting more and more attention for providing a flexible, on demand computing infrastructure for a number of applications.

The actual cloud computing definition (Badger *et al*., 2011; An and Neuman, 2011) by the national institute of standards and technology is: "Cloud computing is a model for enabling convenient, on-mand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction". The goal of cloud computing in general is to provide services to users with greater flexibility and availability as is often described as "taking everything as a service" (XaaS) (An and Neuman, 2011).

Demand (Goudarzi and Pedram, 2011) for computing power has been increasing due to the penetration of information technologies in our daily interactions with the world both at personal and public levels, encompassing business, commerce, education, manufacturing and communication services. At personal level, the wide scale presence of online banking, e-commerce, SaaS (Software as a Service), social networking and so on produce workloads of great

Corresponding Author: Senthil Kumar, S.K., Department of Computer Science and Engineering, Jansons Institute of Technology, Coimbatore, Tamil Nadu, India

diversity and enormous scale. At the same time computing and information processing requirements of various scientific researches (Hoffa *et al*., 2008), public organizations and private corporations have also been increasing rapidly. Examples include digital services and functions required by the various industrial sectors, ranging from manufacturing to housing, from transportation to banking. Such a dramatic increase in the computing demand requires a scalable and dependable IT infrastructure comprising of servers, storage, network bandwidth, physical infrastructure, electrical grid (Deelman *et al*., 2003), IT personnel and billions of dollars in capital expenditure and operational cost (Yang *et al*., 2008) to name a few

For consumers, it is illusion of infinite computing resources available on demand (Armbrust *et al*., 2008) and computing resources become immediate rather than persistent (Dillon *et al*., 2010) there are no up-front commitment and contract as they can use them to scale up whenever they want and release them once they finish scaling down. Moreover, resources provisioning appears to be infinite to them, the consumption can rapidly rise in order to meet peak requirement at any time. In practice, physical resources of clouds are limited and a performance bottleneck will eventually develop.

Scheduling (Senkul *et al*., 2002; Sakellariou and Zhao, 2004) is fundamental to the achievement of high performance in parallel and distributed systems. Scheduling problems, which are concerned with searching for optimal (or near-optimal) real-time (Srikanth *et al*., 2012) and predictive schedules subject to a number of constraints (Zhang *et al.*, 2009, Yu and Shi, 2008), are mostly NP-hard. In general, problem of determining whether there is an assignment of tasks to servers so that each task's demand may be satisfied by the available resources is NP-complete (Heger, 2010) (unlikely to be solvable in an amount of time that reflects a polynomial function). Even if resources are available to meet a certain demand, to correctly mapping the set of demands with a set of resources may be too complex to solve within an acceptable time-frame. In cloud computing, delivering services and resources on demand over a network requires addressing numerous technological issues, including automated provisioning, dynamic virtual server migration, or network security problems. Further, in a cloud environment, not all the resources (virtualized server systems) may actually be available to all customers, due to network latency, commercial agreements, or some security policy issues.

A cloud resource scheduler (Bautista *et al, 2012*, Dillon *et al*., 2010) should make full use of all kinds of resources on Internet such as computing, network bandwidth and storage resource. However, most of jobs that cloud computing needs to deal with are small granularity jobs (Liu *et al*., 2011), which means to need longer waiting-time, consume more resources and lead to lower flexibility and other drawbacks. Under cloud computing environment, in regard to multi-user and large amounts of small granularity concurrent job requirements, how to properly dispatch jobs to different slave nodes to avoid underutilization and how to deal with workload unbalance are the bottlenecks which importantly influence system performance.

The existing scheduling algorithms (Bala and Chana, 2011) consider various parameters like time, cost, make span, speed, scalability, throughput, resource utilization, scheduling success rate and so on. but, for a multiple workflows (Yang *et al*., 2008), metrics like reliability and availability (Bamiah and Brohi, 2011) should also be considered. Existing scheduling algorithms does not consider reliability and availability. Therefore there is a need to implement a workflow scheduling algorithm that can improve the availability and reliability in cloud environment.

Related work: There are plenty of research is going for resource scheduling for improving various factors in cloud computing. Normally all research is based on the heuristic based algorithm where we requires lot iterations to achieve the optimal cost and minimizing the waiting time and turn-around time. No scheduling algorithm considers the important parameters such as reliability, Availability and improving the scalability. And also complex algorithm makes the cloud scheduler as more complex.

Gu *et al*. (2012) proposed a genetic algorithm based scheduling and considers the historical data and current states of VM, uses tree structure to do the coding in genetic algorithm, proposes the correspondent strategies of selection, hybridization and variation also puts some control on the method so that it has better astringency.

Chen and Zhang, 2009, Yang *et al*. (2008) proposed an Ant colony based task scheduling architecture to improve the scheduling behavior, better utilize (balance) the available resources, lower aggregate task execution time and hence, minimize cost.

Heger (2010) proposed an ANN based task scheduling architecture to improve the scheduling behavior, better utilize (balance) the available resources, lower aggregate task execution time and hence, minimize cost.

Henzinger *et al*. (2010) proposed a method known as “flexible provisioning of resources in a cloud environment, (Flex PRICE) where the cloud (provider) and the users build a symbiotic relationship. Instead of renting focuses on allocation of resources across tasks using a set of specific resources, the user

simply presents the job to be executed to the cloud. The cloud has an associated pricing model to quote prices of the user jobs executed.

Tayal (2011) proposed a centralized scheduler (master node) a choice by referring to a global view of the whole system with fuzzy setting based on GA parameters. Their idea was the adaptation of the GA operator's value (selection; crossover; mutation) during the run of the GA. The fuzzy control is applied if the condition of fuzzy adaptation is true. This Model described the information related to processors which includes slot information, data replication information and workload information of processors.

Senkul *et al*. (2002) presented a logical framework for scheduling work-flow under resource allocation constraints. The framework is based on Concurrent Constraint Transaction Logic (CCTR) and integrates Concurrent Trans-action Logic with Constraint Logic Programming. They presented an algorithm that took the initial work flows specification and a set of resource allocation constraints and returns a new work-flow and a resource assignment, such that every execution of that workflow is guaranteed to satisfy the constraints.

Clark *et al*. (2012) introduced an Intelligent Cloud Resource Al-location Service (ICRAS). ICRAS supports the consumer with (1) discovering all available resource con-figurations, (2) choosing the desired configuration, 3) negotiating a service agreement with the CSP, (4) monitoring the service agreement for violations and 5) assisting in the migration of services between CSPs.

Ramamritham *et al*. (1989) was among the first to propose the use of distributed algorithms to schedule tasks with time and resource restrictions. They give different algorithms for this purpose and a comparison of their performance. They claim that their solution is effective even in hard real-time environments. However, their approach requires each node to have full knowledge of the rest of the system, which naturally limits its scalability.

Zhong and Zhang (2010) proposed an optimized scheduling algorithm to achieve the optimization or sub-optimization for cloud scheduling. In this algorithm an Improved Genetic Algorithm (IGA) is used for the automated scheduling policy. It is used to increase the utilization rate of resources and speed.

Selvarani and Sadhasivam (2010) proposed an improved cost-based scheduling algorithm for making efficient mapping of tasks to available resources in cloud. This scheduling algorithm measures both resource cost and computation performance, it also Improves the computation/communication ratio.

An and Neuman (2011) proposed a scheduling algorithm which takes cost and time. The simulation has demonstrated that this algorithm can achieve lower cost than others while meeting the user designated deadline.

Liu *et al*. (2010) presented a novel compromised-time-cost scheduling algorithm which considers the characteristics of cloud computing to accommodate instance-intensive cost-constrained workflows by compromising execution time and cost with user input enabled on the fly.

Pandey *et al*. (2010) presented a Particle Swarm Optimization (PSO) based heuristic to schedule applications to cloud resources that takes into account both computation cost and data transmission cost. It is used for workflow application by varying its computation and communication costs. The experimental results show s that PSO can achieve cost savings and good distribution of workload onto resources.

Lin and Lu, (2011) proposed an SHEFT workflow scheduling algorithm to schedule a workflow elastically on a Cloud computing environment. The experimental results show that SHEFT not only outperforms several representative workflow scheduling algorithms in optimizing workflow execution time, but also enables resources to scale elastically at runtime.

Wu *et al*. (2011) proposed a market-oriented hierarchical scheduling strategy which consists of a service-level scheduling and a task-level scheduling. The service-level scheduling deals with the Task-to-Service assignment and the task-level scheduling deals with the optimization of the Task-to-VM assignment in local cloud data centers.

Xu *et al*. (2009) worked on multiple workflows and multiple QoS. They had a strategy implemented for multiple workflow management system with multiple QoS. The scheduling access rate is increased by using this strategy. This strategy minimizes the make span and cost of workflows for cloud computing platform.

Varalakshmi *et al*. (2011) proposed OWS algorithm for scheduling workflows in a cloud environment. The scheduling algorithm finds a solution that meets all user preferred QoS constraints. With this algorithm, a significant improvement in CPU utilization is achieved. Parsa and Entezari-Maleki (2009) proposed a new task scheduling algorithm RASA. It is composed of two traditional scheduling algorithms; Max-min and Min-in. RASA uses the advantages of Max-min and Min-min algorithms and covers their disadvantages. The experimental results show that RASA is outperforms the existing scheduling algorithms in large scale distributed systems (Xu *et al*., 2009).

Inconveniences with existing methods: In practice, cloud computing is highly dynamic and tasks are not always executed in the same style. For this type of problem, genetic algorithms have difficulty dealing with "deceptive" fitness functions (Melanie, 1998), those where the locations of improved points give misleading information about where the global optimum is likely to be found. ANN, ant colony (Yang *et al*., 2008) PSO (Liu *et al*., 2010) and honey bee algorithms are heuristic and need lot of considerable time to get trained and react on the situation. As the cloud, dynamic style, more new clients and new tasks introduces, same type of task may not be very much frequent.

Linear Programming (LP or linear optimization): Linear Programming (LP or linear optimization) is a mathematical method for determining a way to achieve the best outcome (such as maximum profit or lowest cost) in a given mathematical model for some list of requirements represented as linear relationships. Linear programming is a specific case of mathematical programming (mathematical optimization).

More formally, linear programming (Liu *et al*., 2010) is a technique for the optimization of a linear objective function, subject to linear equality and/or linear inequality constraints. It is feasible region is a convex polyhedron, which is a set defined as the intersection of finitely many half spaces, each of which is defined by a linear inequality. Its objective function is a real-valued affine function defined on this polyhedron.

The Transportation problem: There is a type of linear programming problem (Reeb and Leavengood, 2002, Liu *et al*., 2010) that may be solved using a simplified version of the simplex technique called transportation method. Because of its major application in solving problems involving several product sources and several destinations of products, this type of problem is frequently called the transportation problem. It gets its name from its application to problems involving transporting products from several sources to several destinations. The two common objectives of such problems are either (1) minimize the cost of shipping m units to n destinations or (2) maximize the profit of shipping m units to n destinations. Let us assume there are m sources supplying n destinations. Source capacities, destinations requirements and costs of material shipping from each source to each destination are given constantly.

Proposed system:

Overview of proposed system: In our proposed system, the clients can assign their tasks with priority value between 1 and 5 where 1 has the highest priority and charges more per unit of time and 5 has the lowest priority with least charges per unit of time. Sometimes the priority value can be automatically assigned to the client's task based on the Service Level Agreements (SLA) (Buyya *et al*., 2011) with client. Our System receives the tasks from clients with Flexible Quantum of buffer time. This buffer time to receive can be extended based on the inter-arrival time of tasks. After receiving set of tasks, that is transferred to scheduler. The scheduler gets all necessary information from other phases like workload predictor and Historical information from Task Info Container such as Expected Execution Time (EET), Expected Worst-case execution Time (EWT), Success Score within Expected Time (SSEET), Success Score within Worst-case Time (SSWT), Resources-Required (RR) and Cost for the task execution for each Cluster. With that information preprocessing is done to build the Transportation Problem Table (Table 3). The Column Minima (which gives the least cost for the execution in particular Cluster of resource from set of Cluster) method is used for efficient scheduling plan, which provide as much tasks scheduling as possible with minimum total cost for allocation. After the Scheduling, the Allocator generates a queue (execution Sequence order) of scheduled tasks based on EET and EWT time in ascending order for available resources at an every instance of time and allocates resources in the order. Resources availability can be periodically predicted with Resource predictor. Figure 1 shows the decentralized dynamic cloud scheduler.

With this system we can persevere and enhance the Reliability by considering the available fault-free resources for allocation of tasks and we also take the historical values for scheduling. So the task Execution failure because of resources is prevented. So the reliability can be preserved. Also our system considered the minimum cost and maximum profit to the cloud providers.

Proposed System Architecture: Our proposed system Architecture is shown in Fig. 2. It consists of 7 different phases to produce the scheduling and allocation of tasks with reliability. The concept behind the each phases of our system are

Task initiator: Our cloud environment is decentralized scheduling and task allocation, dynamic in nature, the clients can assign task to the cloud at any point of time. The tasks are assigned to the cloud is in poisson arrival pattern and tasks are independent with other tasks.

The task initiator has the following functions.

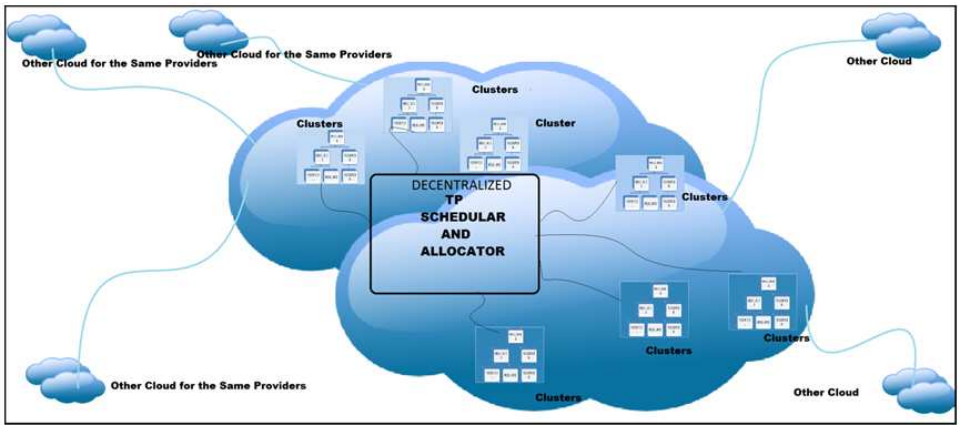

Fig. 1: Our decentralized dynamic cloud environment model

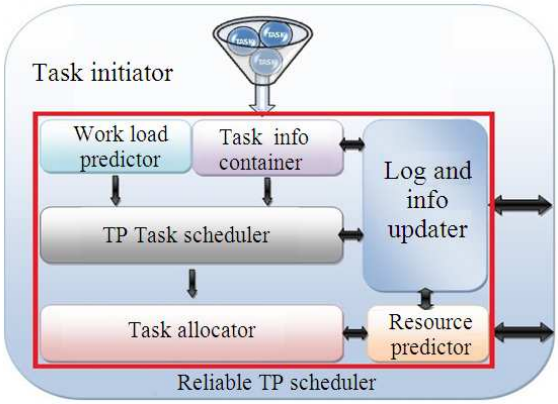

Fig. 2: The proposed system architecture

This phase maintains the necessary condition for the Linear Programming problem Transportation problem where Σ Sources = Σ Destinations by receiving the tasks from various clients' task assignment in which the sum of resources required should be equal to readily available Cluster resources at an instance of time.

Task initiator contains the Flexible Quantum Time Slice (FQTS) as tasks receiving buffer time. By default it has a fixed slice of time to receive the clients' tasks say 5 sec. If more tasks are assigned by clients and the inter-arrival time is too short say less than or equal to 200 ms then the time slice extended up to the inter-arrival time maintained to 200 millisecond. When FQTS completed and no tasks arrived with less than equal to 200 millisecond then tasks are transferred to TP scheduler.

Workload predictor: This phase provide the necessary information to the Tasks arrived into decentralized cloud environment for execution. Those information are such as (i) Expected Execution Time (EET) which gives the average case execution time for the task with given input parameters, (ii) Expected Worst case Execution Time (EWT) for the task with given input parameters. This phase also predicts an important attribute such as (iii) Resources-Required (RR) to complete the task in an efficient manner.

We have an assumption that the resources available in the Cluster of the cloud have same capacity and same capability but it has different quantity of resources available at instance of time. To minimize the prediction time, the Workload Predictor calculates EET, EWT and Resources-Required for only the tasks assigned for the first time to our cloud environment, if the task is already introduced, then those information such as EET, EWT and RR can be retrieved from the Task Info Container which was updated by the Log and Info Updater.

Task info container: The Task Info Container is storage with controller that keeps all the historical information about already assigned task. The Historical information is such as Task id, EET, EWT and Resource-Required those can be predicted at the first time by work load predictor and can be utilized for the near future execution. And other Historical data such as Success Percentage of a Task within EET called SSET, Success Percentage of a task within EWT called SWET and execution cost for task in each Cluster represented by $Cost_{ij}$ where i stand for Cluster and j represents the Task.

When the task is assign to the cloud, Task Info Container provides all the historical information to the TP scheduler for scheduling. For the new task these information's are newly generated and stored into the Task Info Container.

Log and info update: This phase keeps-on updating all activities within cloud environment into the log as well as updating of task information in the appropriate storage. The functions of this phase are:

- Provide the status of all Clusters and its resources by periodically collected from Cluster
- Update the success and failure percentage of each task executed with EET and EWT represented as SSET, SSWT
- Update the Cost information of a task in each execution for appropriate Cluster
- This Phase helps the Resource availability predictor by providing periodic status
- Update unallocated task details and reasons for that such as resource unavailability, resource failure during the execution, Task failure during the execution time and when the Task exceeds the EWT
- Provide all the above information to the TP scheduler on demand

TP task scheduler: The objectives of TP task scheduler are:

- Efficiently allocate more tasks within available Resources at an instance of time
- To maximize the reliability of the cloud environment and to maximize the profit for cloud provider
- To minimize the execution cost within Clusters and load of the resources of the Cluster
- To maximizing the success percentage of task either in EET or in EWT
- The TP scheduler has control over the Task Scheduling based on the following constrains
- Selection of Tasks which are all satisfying the necessary conditions on the TP at instance of the time
- Priority queue maintained for the Tasks by priority value given either by client for the Task or as in the Service Level Agreement (SLA) with the client.
- Scheduling the highest Success Percentage of the task either within EET or cumulative success percentage of EET and EWT (Percentage of successful Task completion updated by Log and Info Updater)
- The lowest cost for task-Cluster combination (predicted from historical values)

With the above constrains, this phase formulate LPP based TP table and generate schedule plan with the lowest execution cost and make allocation queue based on smallest to highest value of EET.

Task allocator: This phase receives the schedule plan for the tasks from the TP task scheduler. The allocator makes the queue or allocation order of tasks for each Cluster with appropriate Tasks by ascending order based on EET. The task allocator allocates all tasks or some possible task to the Cluster resources as per the scheduler plan generated by the scheduler. Sometimes the task allocator may not allocate the entire task to the Cluster because of the unavailability of resources. These tasks are kept in separate queue and allocate when resources are released from the task already allocated. One additional queue is maintained where all new tasks which do not have the necessary information such as SSET, SSWT cost for all clusters, will be allocated to the freely available resources in any clusters with near future.

Resource predictor: This phase periodically collects the status of each resources of the Cluster and keeps them update for helping the task allocator. Thus the task allocator can allocate the tasks as available in the queue. Also this phase monitors and collects the working status of the resources available in the Cluster, which can be used to avoid the resource failure before it occurs.

Mathematical formulation: The mathematical formulation of cloud resource scheduling and allocation using the modified Transportation problem consist of present and historical values. The cloud reliable scheduling and allocation using Transportation problem CRSATP can be defined as:

TP_Schedular = {T, C, S, D, TP, CO, EET, EWT, SSEET, SSWT, CH}

Where:

T = Set of TASKS from various Requesters defined as T_j with j = 1, 2 ,…, n

C = Set of Clusters of the cloud with various number of equal capability resources, defined as C_i with i = 1, 2 ,…, m

Table1: Initial problem formulation table

	T1		T2		T3		T4		
	TP I (Task Priority)		TP 2 (Task Priority)		TP 3 (Task Priority)		TP 4 (Task Priority)		
	EET1	EWT1	EET1	EWT1	EET1	EWT1	EET1	EWT1	Source
C1	COST 1, 1 SSEET (1,1)	SSWT (1,1) Allocation	COST 1,2 SSEET (1,2)	SSWT (1,2) Allocation	COST 1,3 SSEET (1,3)	SSWT (1,3) Allocation	COST 1,4 SSEET (1,4)	SSWT (1,4) Allocation	Available resources at instance of time S1
C2	COST 2, 1 SSEET (2,1)	SSWT (2,1) Allocation	COST 2,2 SSEET (2,2)	SSWT (2,2) Allocation	COST 2,3 SSEET (2,3)	SSWT (2,3) Allocation	COST 2,4 SSEET (2,4)	SSWT (2,4) Allocation	Available resources at instance of time S2
C3	COST 3,1 SSEET (3,1)	SSWT (3,1) Allocation	COST 3,2 SSEET (3,2)	SSWT (3,2) Allocation	COST 3,3 SSEET (3,3)	SSWT (3,3) Allocation	COST 3,4 SSEET (3,4)	SSWT (3,4) Allocation	Available resources at instance of time 3
C4	COST 4,1 SSEET (4,1)	SSWT (4,1) Allocation	COST 4,2 SSEET (4,2)	SSWT (4,2) Allocation	COST 4,3 SSEET (4,3)	SSWT (4,3) Allocation	COST 1,1 SSEET (4,4)	SSWT (4,4) Alloc ation	Available resources at instance of time S4
Desti nation	Resources required for Task 1		Resources required for Task 2		Resources required for Task 3		Resources required for Task 4		
CHA RGES	Charges / Unit of Time As per Priority or SLA		Charges / Unit of Time As per Priority or SLA		Charges / Unit of Time As per Priority or SLA		Charges / Unit of Time As per Priority or SLA		

Table 2: Cost score calculation table

	SS Value	TP = 1	TP = 2	TP = 3	TP = 4	TP = 5
SSEET > 90	0	0.442	0.884	1.326	1.768	2.21
SSET + SSWT > 90	1	1.442	2.884	4.326	5.768	7.21
SSET + SSWT BETWEEN 80 - 90	2	2.442	4.884	7.326	9.768	12.21
SSET + SSWT BETWEEN 70 - 80	3	3.442	6.884	10.326	13.768	17.21
SSET + SSWT BETWEEN 60 - 70	4	4.442	8.884	13.326	17.768	22.21
SSET + SSWT BETWEEN 50 - 60	5	5.442	10.884	16.326	21.768	27.21
SSET + SSWT BELOW 50	10	10.442	20.884	31.326	41.768	52.21
NEW EXECUTION (SAME AS ABOVE)	10	10.442	20.884	31.326	41.768	52.21

Table 3: Actual transportation problem table

	T1		T2		T3		T4		
	EET1	EWT1	EET2	EWT2	EET3	EWT3	EET4	EWT4	Source
C1	Cost score (1,1) Allocation or zero		Cost score (1,2) Allocation or zero		Cost score (1,3) Allocation or zero		Cost score (1,4) Allocation or zero		Available no. of resources at instance of time S1
C2	Cost score (2,1) Allocation or zero		Cost score (2,2) Allocation or zero		Cost score (2,3) Allocation or zero		Cost score (2,4) Allocation or zero		Available no. of resources at instance of time S2
C3	Cost score (3,1) Allocation or zero		Cost score (3,2) Allocation or zero		Cost score (3,3) Allocation or zero		Cost score (3,4) Allocation or zero		Available no. of resources at instance of time S3
C4	Cost score (4,1) Allocation or zero		Cost score (4,2) Allocation or zero		Cost score (4,3) Allocation or zero		Cost score (4,4) Allocation or zero		Available no. of resources at instance of time S4
Destination	Resources Required for Task 1		Resources Required for Task 2		Resources Required for Task 3		Resources Required for Task 4		Resources Required for Task 1
Charges	Charges / / Unit of Time As per Priority or SLA		Charges / Unit of time As per Priority or SLA		Charges / Unit of Time Unit of Time As per		Charges / Unit of Time As per Priority or SLA		Charges / Unit of Time As per Priority or SLA

S = Set of sources ie. Resources of equal capability available for each Cluster, defined as S_i with i= 1,2,..m (S_i is available resources at an instance of time for Cluster C_i)

D = Set of destinations ie. Number of Resources of equal capability required to complete the Task, defined as D_j with j= 1, 2,n

TP = Set of TASK Priority of a TASK specified by either Requester or as per SLA with Requester, defined as TP_j with j = 1,2,…n

CO = Set Of Processing COST for TASK processing each Cluster, defined as $CO_{ij,}$ where CO_{ij} is the cost at $cluster_i$ for $TASK_j$ for one resource used in unit of time

EET = Set of Expected Execution Time by workload Predictor for a TASK (Average case Time for N input) irrespective to the CLUSTER, defined as EET_j, where Expected Execution Time for $TASK_{j.}$

Eewt = Set of expected execution worst case time by workload predictor for N input to a TASK irrespective to the cluster, defined as $eewt_j$, where Expected Execution Time for $TASK_{j.}$

SSEET = Set of success score for EET (Percentage of Success in EET time at $Cluster_i$ for $TASK_{j)}$ by Historical Value to a TASK respective to the cluster, defined as $SSEET_{ij}$, where success score EETIJ at $Cluster_i$ for $TASK_j$.

SSWT = Set of success score for EWT(Percentage of Success within EWT Time for $TASK_i$ at $Cluster_j$) by Historical Value to a TASK respective to the CLUSTER, defined as $SSWT_{ij}$, where $SSWT_{ij}$ at $Cluster_i$ for $TASK_j$.

CH = Set of Charges for processing the TASK in any cluster for any number of resources for a Unit of Time, Defined as CH_j

Objective function: The main objective to this scheduling and allocation function is to minimize the total cost $\text{Min} \sum_{i=1}^{i-m} \sum_{j=1}^{,j-n} Xij\, Cij$, failure percentage of execution of a Task $\text{Min} \sum_{i=1}^{i-m} \sum_{j=1}^{,j-m} Xij\, Fij$ and to maximize the profit and success percentage $\text{Max} \sum_{i=1}^{i-m} \sum_{j=1}^{,j-n} Pij$. The initial table formulation is as given in Table 1. This table contains the tasks, Clusters sources, destinations and predicted execution time, success score calculated from the previous history vales and charges to each tasks completion. To formulate the following formula is used to find the single cost score to fit into the actual TP table, shown in Table 3:

Cost score = $((CO_{ij} / 1000) + SS\ Value_{ij}) * TP_j$

The equivalent SS value for the SSET and SSWT is shown in Table 2, for example if the CO_{ij} = 442 then cost score is available in the Table 2

Algorithm for our proposed TP scheduling:

Step 1: Formulate the initial transportation table with available and received attributes and tasks as Table 1

Step 2: Find the cost score $_{ij}$, cost score = (((COij/1000) + SS Value $_{ij}$) * TP_j SS value = 0 if SSET>90, 1 if SSET+SSWT>90, 2 IF SSET+SSWT Between 80-90, 3 for 70-60, 4 for 60-50, 5 for 50-60 and FOR<50 and new execution 10. refer Table 2.

Step 3: Formulate the actual transportation problem with above data as Table 3.

Step 4: Repeat step 5-7 for all tasks available.

Step 5: Find the minimum EET/EEWT mark the column

Step 6: Find the minimum cost score for the column marked and allocate the resources as required by the task

Step 7: Make the EET marked and cost score ij into infinitive value.

Step 8: Find the order of execution by finding maximum charges/unit of time as the first task and so on.

Step 9: Allocate all resources and update the tasks history after execution is over

Step10: If required resources are more than resources of all Cluster then eliminate the tasks which has more resources required.

MATERIALS AND METHODS

The genetic algorithm based scheduling the Tasks and resource allocation is implemented. The fitness function was selected to find the total cost for task allocation in Cluster and population is taken as 100 tasks and mutation by changing the allocation vector one task on Cluster with random value. The crossover function was implemented by generating new combination of tasks to Cluster allocation vector. The genetic Algorithm is executed with 100 tasks, mutation rate by 1% and Crossover rate by 96.5%.

The Genetic Algorithm generates the maximum of 400 iterations to produce the near optimum value comparing with our algorithm which generates best optimum cost with minimum number of resources. Two different experiments are conducted. One with one set of 100 tasks another with 10 different set of each 100 tasks to prove that our system can efficiently allocate the dynamic tasks.

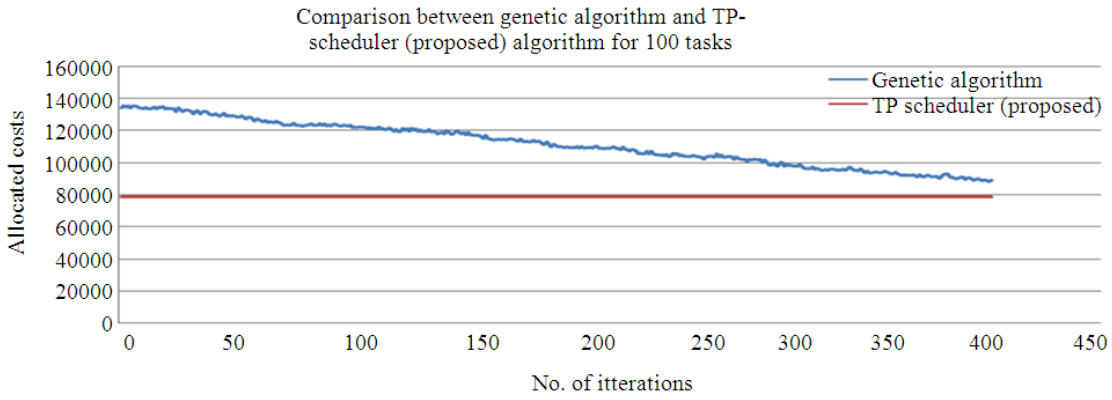

Fig. 3: Comparison between Genetic Algorithms with our Proposed TP scheduling algorithm

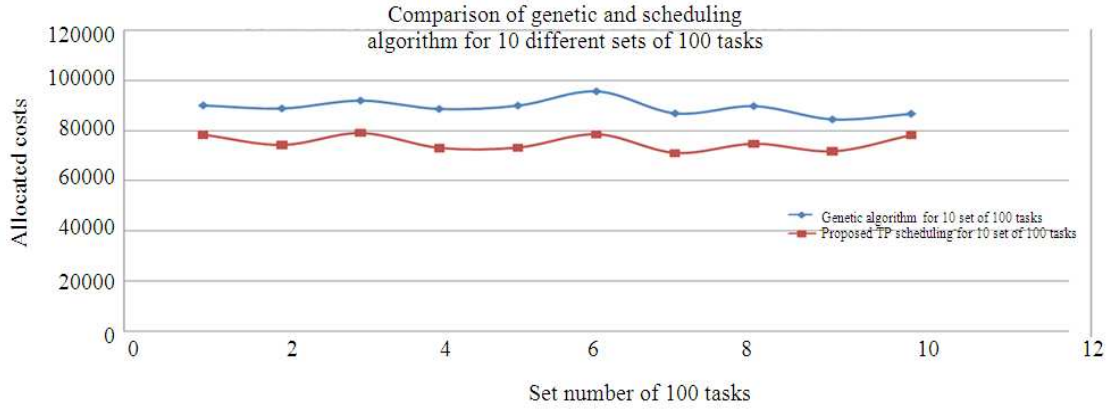

Fig. 4: Experimenting with10 different sets of 100 Tasks between genetic and our proposed

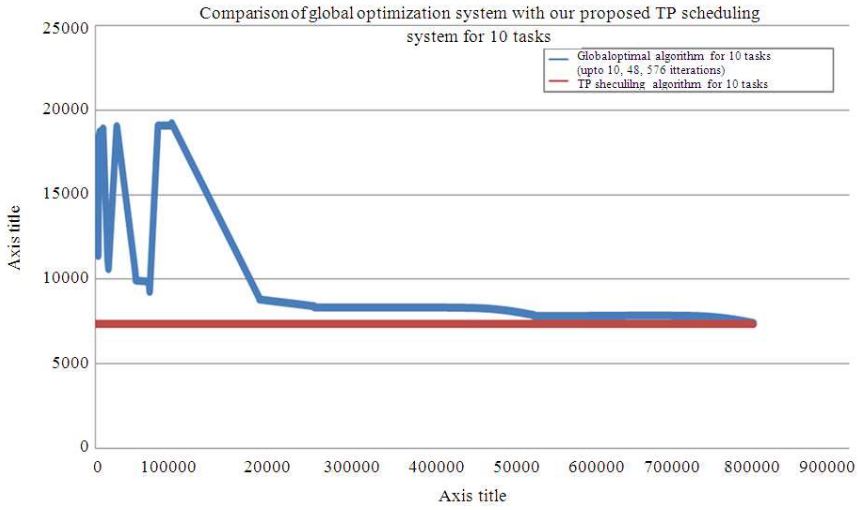

Fig. 5 Comparison of global optimization system with our proposed TP scheduling system

Naturally the genetic algorithm could not produce the optimum results for both experiments and also it takes more execution time to produce the results. All the results are imported in mat lab as xpls file and graphs 1 and 2 are generated (Fig. 3 and 4).

Next we compare our proposed TP scheduling system with Global Optimization System Implemented with total Permutation and Combination method in JAVA, This System can take only of 10 tasks for 4 Clusters and produced the best optimum cost after execution of 10,48,576 different iterations to complete. The graph 3 (Fig. 5.) shows the comparison of TP scheduling with Global Optimization for 10 tasks.

RESULTS

From the graph 1(Fig. 3) we understand that the blue horizontal line (TP Scheduling System) is producing the optimal allocation cost in first iteration itself for the given 100 tasks, but the black line (genetic Scheduling and allocation system) takes 400 itterations it converged to find the near optimal result. It also takes long time to complete the task scheduling. With the graph 2 (Fig. 4) we understand that the black line (TP Scheduling) produces the optimal allocation cost for all 10 sets of each 100 tasks, whereas the red line (Genetic Scheduling) produces near optimal value for all sets with long execution time.

From the graph 3 (Fig. 5) we understand that the TP scheduling produces (red horizontal line) shows that TP scheduling optimal cost in first iteration whereas Global optimization system (red dotted line) produces the optimal cost at 20,000 th iteration (in this data set) and goes upto 10,48,576 total iterations to complete.

From this we can understand TP Scheduler produces better scheduling and resource allocation and minimizes the Cost and maximizes the profit. And the objective of reliability also preserved and enhanced.

DISCUSSION

Our proposed system is implemented as simulation environment using the Core JAVA with System Configuration of Core 2 Duo with T6600 and 2.20 GHz processor and with 2 GB RAM, in which we have 5 clients and 4 Clusters. Each will have random number of resources and 100 tasks are generated with random inter-arrival time. Out of 100 tasks some of the tasks were generated with random priority, EET, EWT, SSET, SSWT, Cost for 4 Clusters. Some of them are generated as new tasks so that the system will generate the Cost and Success scores. With that the TP scheduler is called for scheduling and allocation of tasks. The system generates the allocation and maintains and updates the historical information in Task Info Container. It produces the reliable and best optimum cost in first iteration itself.

With the same simulated environment, our system is compared with two other systems also developed with Core JAVA. (1) Genetic Scheduling Algorithm with same 4 tasks, 100 tasks and with necessary information. (2) Global Optimization by total permutation and combination method for 10 tasks, 4 Cluster and with necessary information.

CONCLUSION

Thus our proposed TP scheduling algorithm for task scheduling and resource allocation in decentralized and dynamic cloud computing environment, efficiently schedule and allocate the tasks. The main objective of this algorithm, to enhance the reliability and maximization the profit by minimizing the allocation and execution cost and minimizing the complexity of cloud controller is achieved.

The reliability is achieved by the following ways. First it considered the actual availability of the resources which are all physically and logically good condition and based on that it schedules the tasks. Second preferences given to the task which are all have most successful by historical values and up-to-date cost values is considered for finding the minimal cost. Third it maximizes allocation of all assigned tasks as earlier as possible. So it serves almost all assigned tasks. This system has Task initiator which removes the bottleneck problem by control the task incoming flow.

Now we have proposed the method for independent tasks with equal capability resources of Clusters and assuming no advanced reservation in Task Assinment. In Future the we are planning to improve the reliability and availability for Task Scheduling and resource allocation for some complex constraints which are not considered now such as resource specialization, critical resources, tasks dependent to predecessor task, time bounded prescheduled tasks and advanced reservation.

REFERENCES

An, L. and C. Neuman, 2011. A Survey on Cloud Computing System Implementation. University of Southern California, Los Angeles, California, U.S.

Armbrust, M., A. Fox, R. Griffith, D. Anthony and Joseph *et al*., 2008. Above the Clouds: A Berkeley View of Cloud Computing. University of California, Berkeley.

Badger, L., T. Grance, R. Patt-Corner and J. Voas, 2011. Draft cloud computing synopsis and recommendations, U.S. Department of Commerce.

Bala, A. and I. Chana, 2011. A survey of various workflow scheduling algorithms in cloud environment. Proceedings of the 2nd National Conference on Information and Communication Technology, (ICT' 11), IJCA, India, pp: 26-30.

Bamiah, M.A. and S.N. Brohi, 2011. Seven deadly threats and vulnerabilities in cloud computing. Inter. J. Adv. Eng. Sci. Technol., 9: 087-090.

Bautista, L., A. Abran and A. April, 2012. Design of a performance measurement framework for cloud computing. J. Software Eng. Appli., 5: 69-75. DOI: 10.4236/jsea.2012.52011

Buyya, R., S.K. Garg and R.N. Calheiros, 2011. SLA-Oriented Resource Provisioning for Cloud Computing: Challenges, Architecture and Solutions. Proceedings of the International Conference on Cloud and Service Computing, IEEE, Australia, pp: 1-10.

Chen, W. and J. Zhang, 2009. An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. IEEE Transactions on Systems, Man and Cybernetics-Part C. Appli. Rev., 39: 29-43. DOI: 10.1109/TSMCC.2008.2001722

Clark, K., M. Warnier and F.M.T. Brazier, 2012. An intelligent cloud resource allocation service agent-based automated cloud resource allocation using micro-agreements. Delft University of Technology.

Deelman, E., J. Blythe, Y. Gil, C. Kesselman and G. Mehta *et al*., 2003. Mapping abstract complex workflows onto grid environments. J. Grid Comput., 1: 25-39. DOI: 10.1023/A:1024000426962

Dillon, T., C. Wu and E. Chang, 2010. Cloud Computing: Issues and Challenges. Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, Apr. 20-23, IEEE Xplore Press, Perth, WA, pp: 20-23. DOI: 10.1109/AINA.2010.187

Goudarzi, H. and M. Pedram, 2011. Maximizing profit in cloud computing system via resource allocation. Proceedings of the 31st International Conference on Distributed Computing Systems Workshops, Jun. 20-24, IEEE Xplore Press, USA., pp: 1-6. DOI: 10.1109/ICDCSW.2011.52

Gu, J., J. Hu, T. Zhao and G. Sun, 2012. A new resource scheduling strategy based on genetic algorithm in cloud computing environment. J. Comput., 7: 42-52. DOI: 10.4304/jcp.7.1.42-52

Heger, D.A., 2010. Optimized resource allocation and task scheduling challenges in cloud computing environments.

Henzinger, T.A., A.V. Singh, V. Singh, T. Wies and D. Zufferey, 2010. FlexPRICE: Flexible provisioning of resources in a Cloud Environment. Proceedings of the 3rd International Conference Cloud Computing, Jul. 5-10, IEEE Xplore Press, USA., pp: 83-90. DOI: 10.1109/CLOUD.2010.71

Hoffa, C., G. Mehta, T. Freeman, E. Deelman and K. Keahey *et al*., 2008. On the use of cloud computing for scientific workflows. Proceedings of the IEEE 4th International Conference on eScience, Dec. 7-12, IEEE Xplore Press, Indianapolis, pp: 640-645. DOI: 10.1109/eScience.2008.167

Lin, C. and S. Lu, 2011. Scheduling scientific workflows elastically for cloud computing. Proceedings of the IEEE 4th International Conference on Cloud Computing, Jul. 4-9, IEEE Xplore Press, Washington, DC, pp: 746-747. DOI: 10.1109/CLOUD.2011.110

Liu, K., Y. Yang, J. Chen, X. Liu, D. Yuan and H. Jin, 2010. A compromised-time-cost scheduling algorithm in swindew-c for instance-intensive cost-constrained workflows on a cloud computing platform. Inter. J. High Perfor. Comput. Appli., 24: 445-456. DOI: 10.1177/1094342010369114

Liu, X.P., Y. Mei, S. Sivathanu, Y. Koh and C. Pu, 2011. Who is Your neighbor: Net I/O performance interference in virtualized clouds. IEEE Tran. Serv. Comput., DOI: 10.1109/TSC.2012.2

Melanie, M., 1998. An Introduction to Genetic Algorithms. 1st Edn., MIT Press, Cambridge, ISBN-10: 0262133164, pp: 221.

Pandey, S. L. Wu, S.M. Guru and R. Buyya, 2010. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, Apr. 20-23, IEEE Xplore Press, Perth, WA., pp: 400-407. DOI: 10.1109/AINA.2010.31

Parsa, S. and R. Entezari-Maleki, 2009. RASA: A new task scheduling algorithm in grid environment. World Applied Sci. J. Spec. Comput. IT., 7: 152-160.

Ramamritham, K., J. Stankovic and W. Zhao, 1989. Distributed scheduling of tasks with deadlines and resource requirements. IEEE Transactions Comput., 38: 1110-1123. DOI: 10.1109/12.30866

Reeb, J. and S. Leavengood, 2002. Transportation problem: A special case for linear programming problemsr. Operations Res.

Sakellariou, R. and H. Zhao, 2004. A hybrid heuristic for DAG scheduling on heterogeneous systems. Proceedings of the 18th International Parallel and Distributed Processing Symposium, 2004. Proceedings, Apr. 26-30, IEEE Xplore Press, DOI: 10.1109/IPDPS.2004.1303065

Selvarani, M.S. and D.G.S. Sadhasivam, 2010. Improved cost-based algorithm for task scheduling in cloud computing. Proceedings of the IEEE International Conference Computational Intelligence and Computing Research, Dec. 28-29, IEEE Xplore Press, Coimbatore, pp: 1-5. DOI: 10.1109/ICCIC.2010.5705847

Senkul, P., M. Kifer, I.H. Toroslu, 2002. A logical framework for scheduling workflows under resource allocation constraints. Proceedings of the 28th International Conference on Very Large Data Bases, (VLDB' 02) ACM, Inc., pp: 694-705.

Srikanth, G.U., A.P. Shanthi, V.U. Maheswari and A. Siromoney, 2012. A survey on real time task scheduling. Eur. J. Sci. Res., 69: 33-41.

Tayal, S., 2011. Tasks scheduling optimization for the cloud computing system. Inter. J. Adv. Eng. Sci. Technol. 5: 111-115.

Varalakshmi, P., A. Ramaswamy, A. Balasubramanian and P. Vijaykumar, 2011. An optimal workflow based scheduling and resource allocation in cloud. Depar. Inform. Technol. Anna Univ. Chen. Ind., 190: 411-420. DOI: 10.1007/978-3-642-22709-7-41

Wu, Z., X. Liu, Z. Ni1, D. Yuan and Y. Yang, 2011. A market-oriented hierarchical scheduling strategy in cloud workflow systems. J. Supercomputing, DOI: 10.1007/s11227-011-0578-4

Xu, M. L. Cui, H. Wang and Y. Bi, 2009. A multiple qos constrained scheduling strategy of multiple workflows for cloud computing. Proceedings of the IEEE International Symposium on Parallel and Distributed Processing with Applications, Aug. 10-12, IEEE Xplore Press, Chengdu, pp: 629-634. DOI: 10.1109/ISPA.2009.95

Yang, Y., K. Liu, J. Chen, X. Liu, D. Yuan and H. Jin, 2008. An algorithm in swindew-c for scheduling transaction-intensive cost-constrained cloud workflows. Proceedings of the 4th IEEE International Conference on e-Science, Dec. 7-12, IEEE Xplore Press, Indianapolis, pp: 374-375. DOI: 10.1109/eScience.2008.93

Yu, Z. and W. Shi, 2008. A planner-guided scheduling strategy for multiple workflow applications. Proceedings of the International Conference on Parallel Processing-Workshops, Sept. 8-12, IEEE Xplore Press, Portland, pp: 1-8. DOI: 10.1109/ICPP-W.2008.10

Zhang, Y., A. Mandal, C. Koelbel and K. Cooper, 2009. Combined fault tolerance and scheduling techniques for workflow applications on computational grids. Proceedings of the 9th IEEE/ACM International Symposium on Clustering and Grids. May 18-21, IEEE Xplore Press, Shanghai, pp: 244-251. DOI: 10.1109/CCGRID.2009.59

Zhong, H., K. and X. Zhang, 2010. An approach to optimized resource scheduling algorithm for open-source cloud systems. Proceedings of the 5th Annual China Grid Conference, Jul. 16-18, IEEE Xplore Press, Guangzhou, PP: 124-129. 10.1109/ChinaGrid.2010.37