ISSN NO. 2320-5407

Journal homepage: http://www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH

RESEARCH ARTICLE

Deadline and Cost based MapReduce Job Scheduling in Heterogeneous Cloud using Dynamic Pricing

Ms. Harsha Daryani[1], Mr. Sanjay B. Thakare[2]

1. M. E. Second Year Student Department of Computer Engineering JSPM's Rajarshi Shahu College of Engineering, Tathawade Savitribai phule Pune University, India

2. Associate Professor Department of Computer Engineering JSPM's Rajarshi Shahu College of Engineering, Tathawade Savitribai phule Pune University, Pune Maharashtra, India

Manuscript Info

Manuscript History:

Received: 15 July 2015
Final Accepted: 16 August 2015
Published Online: September 2015

Key words:

Budget constraints; cloud service providers; dynamic pricing; heterogeneous cloud; task scheduling.

Corresponding Author

Ms. Harsha Daryani

Abstract

Cloud Computing has emerged as an eminent technology in internet world. Job scheduling is one of the most challenging issues in the cloud computing area. This paper is centered on scheduling algorithms for MapReduce jobs with respect to deadline and cost constraints on a range of provisioned heterogeneous machines in the cloud. Each job is partitioned into multiple tasks. In addition, the time for executing a task on clouds resources and price for using that resource is determined. We propose a dynamic pricing algorithm for dynamically changing the price required for task execution. It helps the cloud service provider(CSP) to recover its machine's cost and various other cost required for task execution based on the scenario when job is submitted. This algorithm will benefit CSP to maximizes its revenues. In addition to that, it provides inexpensive services to users as the cost of running jobs in the cloud is reduced by dynamic pricing. The Budget Distribution algorithm takes input the dynamic time-price table and distributes total budget among all the tasks, and remaining budget to slowest task to increase overall execution time. In this approach, we use Multiple Choice Knapsack (MCKS) deadline constrained algorithm that takes the dynamic pricing table as input and optimal cost efficient machine based on execution time required on that machine. An evaluation has been carried out which verifies the efficiency of proposed algorithm and also illustrates that the technique is cost-effective to both CSP and user.

INTRODUCTION

Cloud Computing is emerging as a new technique for large scale distributed computing driven by budget, in which a set of highly scalable, virtualized, heterogeneous, and configurable and reconfigurable computing resources can be rapidly acquired and released with minimum management effort. Users of cloud computing does not need to set up any kind of software and can acquire their data worldwide from any device (laptops, tablets, mobile phones, desktop PC) till an internet connection is available [10][11].

MapReduce has emerged as a parallel programming paradigm for processing data in large data centers. It can efficiently handle large compute cluster. Hadoop is an open source implementation of MapReduce parallel

programming framework and Google File System. It allows for distributed processing of large data sets across clusters of computers, each offering local computation and storage [13].

Pricing is an important factor for organizations providing services or products. The way price is set effects customer behavior, commitment to a provider, and the organizations success. Hence, building an appropriate pricing model will help attain higher revenues. Many pricing techniques are utilized [6][7]. For example, a typical pricing technique is to pay once for unlimited usage. On the other hand, this approach
is inflexible and does not take into account many other factors that effects pricing, such as maintenance costs for resource, and price fairness. Many cloud computing providers (e.g., Google App Engine and Amazon Web Services) provides pay-peruse fixed pricing which charges end users as per their overall resource usage. Pay for resources is another approach, in which users are charged based on the storage or bandwidth size allowed. Subscription is another pricing approach, in which the user subscribes with a certain CSP for a static price per unit for long periods of time. Dynamic pricing is another technique, in which the price charged to user for resource consumption changes dynamically. It directly affects both a provider's revenue and price charged to customer for resource usage.

The following are the factors that affect pricing in cloud environment:

1. Initial Costs: This is quantity of money that the service providers spend to purchase resources.
2. Cost of maintenance: This is the quantity of money that service providers spends to maintain cloud resources.
3. Demand: This is the demand from users for utilizing cloud resources. It is usually dynamic and unpredictable.
4. Supply: This indicates the available resources with cloud to fulfill demand. [5]

The task-level scheduling is performed for two criteria: budget and deadline [8][9][13]. The goal of the algorithms, Budget Distribution algorithm and Multiple Choice Knapsack
(MCKS) deadline constrained algorithm are:

I. For a given budget B, how to effectively select a machine from the candidate set of machines for each task so that the total time for executing the workflow comes out to be minimum without breaking the budget.

II. For a given deadline D, how to effectively select a machine from the candidate set of machines for each task so that the total cost for executing the workflow is comes out to be without breaking the deadline.

The scheduler will ensure that all tasks have adequate budget to finish execution. If there is any remaining budget of workflow, then iteratively this budget is distributed to a task whose current execution time ascertains to be the slowest. The process continues until no budget is left. The heterogeneous nature of cloud is depicted by having distinct machines that may have distinct configuration or performance parameters, and thus may have distinct service charges.

Motivation:

The cloud resources are often provisioned on-demand with a billing model, cloud based application are generally budget driven. Hence, the efficient use of resources to meet relevant performance requirements within budget is always a practical concern for Cloud Service Providers. A Cloud Service Providers (CSP) typical aim is to maximize its revenues with its pricing scheme, although its user's main aim is to acquire required service obtainable for a reasonable price. Hence, satisfying both parties requires an optimal pricing technique. The price charged is one of the most significant metrics that a CSP can control to promote the usage of its services.

Challenges:

1. Determining cost and deadline for each task.
2. Dynamically determining price charged to user.
3. Prices are charged per instance per time unit.

This paper is further organized as follows: in Section II we mention emergence of job scheduling and related work in job scheduling recently developed. Section III presents implementation details in which problem definition,

problem formulation, system architecture, algorithms, mathematical model, modules are explained. We follow with the results in section IV and conclude the paper in section V.

Related work

Following outlines the scheduling mechanisms developed in recent years.

Default FIFO Scheduler, uses a FIFO queue for functioning. After partitioning a job into multiple tasks, it is entered into a queue and assigned to TaskTracker nodes as they become available. Typically each job uses entire cluster, so jobs had to wait for their turn to arrive. Problem arises of sharing resources among users. Fair Scheduler [1] was developed by Facebook to handle access to their cluster. It overcomes the challenge of FIFO Scheduler by providing a fair share of the cluster capacity. In addition to that, users may load their jobs to pools, with each pool having a minimum number of TaskTracker slots. Thus cluster capacity is shared among jobs. Capacity Scheduler [2] was developed by Yahoo considers scenario of users where number of users is large. It performs fair allocation of computation resources among users rather than jobs. It does not consider resource availability.

Mark Yong, Nitin Garegrat, Shiwali Mohan [3], proposed two mechanisms for resource aware scheduling. First, Dynamic Free Slot Advertisement computes the available TaskTracker nodes dynamically by using resource metrics retrieved from each node. Second, Free Slot Priorities/Filtering configures maximum number of computation slots for each node at configuration time. Thomas Sandholm and Kevin Lai [4], this paper supports capacity distribution between concurrent users considering priorities of the users. It proposed the idea of sharing TaskTracker slots proportionally per unit time. It does not handle deadline requirements. Xiaocheng Liu et al.[12], is centered on parallel processing capabilities of cloud. This work is intended to attain specific level of utilization of its nodes as well as responsiveness of parallel jobs. Priority based have been proposed with two VM- tiers (low CPU priority and high CPU priority) to improve responsiveness of jobs.

Dzmitry Kliazovich1 et al.[13], this work emphases the role of communication fabric and presents a scheduling solution, named e-STAB, which takes into account traffic requirements of cloud applications providing energy efficient job allocation and traffic load balancing in data center networks. Abhishek Verma, Ludmila Cherkasova, and Roy H. Camp- bell [14], goal of this paper is to minimize the completion time of a set of jobs. It also focuses on improving cluster utilization. It handles a set of production workload that consists of jobs with no dependencies. Z. Guo and G. Fox.[15], proposed resource stealing to utilize idle resources. The paper focuses improving resource utilization and decreasing job execution time. It also explains the concepts of MapReduce in detail. Yang Wang and Wei Shi [16], proposed task level scheduling algorithms with constraints of budget and deadline. The proposed mechanism uses static pricing which could not be fair to every user because every user does not have the same needs.

Implementation Details

A. Problem Definition

We design a system for scheduling of task within budget and deadline constraints using dynamic pricing for determining the time required for execution of task on the best machine from candidate set and price for using that machine.

The scheduling algorithm for task level with regard to budget and deadline constraints for a batch of jobs on a set of provisioned heterogeneous machines in cloud environment where price remains unaltered after it has been determined (static pricing). It introduces the problem of being impartial to users and with service providers. As the service providers might reserve the resources for longer time than the customers has utilized. On the user side, they may overpay for the resources reserved if they does not use them extensively. To address this problem, we are proposing a mechanism for dynamically changing the price and time. Time parameter indicates the time to run a task on a particular machine within the given deadline of the heterogeneous cloud. Price indicates the cost of using that machine which should be within allotted budget. According to the current scenario of machines in the cloud, task level scheduling will be performed.

Problem Formulation

A batch of jobs is modeled as a multi-stage fork and join workflow which consists of 'x' stages, each stage 's' will be having a collection of independent tasks, denoted as $J_s = \{J_0, J_1, J_2, J_3, \ldots, J_n\}$, where n+1 is the size of stage

s. In the cloud, each task can be related with a set of machines. For task J_i, $0 \leq s \leq x$, $0 \leq i \leq n$ the time to run task on a machine and its corresponding prices are shown in Time-Price Matrix where t^m , $1 \leq m \leq r$ gives time to run a task J_i on machine M^m where p^m gives the corresponding price for using that machine, and r is the total number of machines that can execute task J_i. We assume that the prices have been sorted in decreasing order and times in increasing order. TableI indicates the conventions used. For each task, the time for scheduling it is given by TaskTime and price for using virtual machine is given by TaskPrice.

TaskPrice = Total Cost / (100 * Task Size) (1)

TaskTime = Processing Speed / Task Size (2)

Time-Price Matrix for task J_i

$$\begin{bmatrix} t^1 & t^2 & \dots & t^r \\ p^1 & p^2 & \dots & p^r \end{bmatrix}$$

TABLE I. Memorization Parameter Table

Notation	Meaning
x	Number of stages
J_s	Task set in stage s
J_i	Task i in stage J
n	Number of tasks in stage s
W	Total number of tasks in the workflow
t^m	Time to run task J_i on machine M^m
p^m	Cost for using machine M^m
r	Total number of machines that can run
B_i	Budget used by task J_i
B_s	Budget for stage s
B_w	Total budget for workflow
D_s	Deadline for stage s
D_w	Total deadline for workflow
$T_i (B_i)$	Shortest time to finish J_i for given B_i
M^m	Assigned machine

Budget Constraints

Given budget B_i for task J_i , shortest time to finish it is denoted as $T_i (B_i)$, defined as :

$T_i (B_i) = t^m$, $p^{m+1} < B_i < p^{m-1}$ (3)

The time to finish a stage s with budget B_s denoted as $T_s (B_s)$ and is defined as time taken to finish last task in the stage within given budget:

$T_s (B_s) = \max \sum_{i \epsilon [0,n]} T_i (B_i)$ (4)

Since workflow is fork and join, a stage cannot start until its previous stage had finished. Hence, the total makespan within budget B_w to finish the workflow is defined as the sum of all stages times. We need to minimize this time within budget B_w :

$T_w (B_w) = \min \sum_{s \epsilon [0,x]} T_i (B_i)$ (5)

Deadline Constraints

The minimum cost to finish stage s, given deadline D_j denoted as $C_s (D_s)$:

$C_s (D_s) = \sum_{i \in [0,n]} C_i (D_s)$ (6)

where $C_i (D_s)$ is the minimum cost to finish task J_i in stage s within such that $t^1 \leq D_s \leq t^r$

The total cost for the workflow within deadline D_w :

$C_w (D_w) = \min \sum_{s \in [0,x]} C_s (D_s)$ (7)

System Architecture

Fig. 1 shows the system architecture. Users access the cloud through web browser and submit their jobs into cloud. Cloud administrator estimates the time and price required for task execution and dynamically maintain it. Budget and deadline distribution solver finds appropriate node for executing task. If the task can be completed within budget or deadline it is scheduled, else it is not. It estimates the scheduling length and cost for complete workflow.

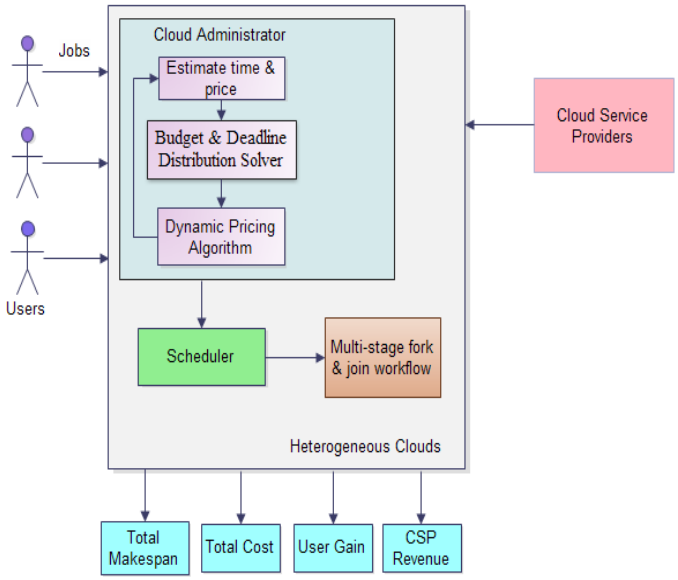

Fig. 1. System Architecture

Algorithms

Assumption: Communication between map/reduce tasks is manipulated by the MapReduce framework via the underlying network file systems.

System Constraints: For time-price table, time should be sorted in ascending order whereas price should be sorted in descending order.

Algorithm 1: Budget Distribution Algorithm

Input: JobSet; total Budget; initial execution time, budget distribution, assigned machine

Output: Minimum time for executing workflow

1: Maintain idle list of machines
2: **for** task ∈ stage s **do**
3: Find best machine from candidate machines which is idle.
4: **end for**
5: Calculate remaining budget for workflow
6: Initialize profile variables (T_i, B_i, M^m)
7: Identify slowest and second slowest task

8: Assign these pairs to a set L
9: Determine which task in L should be allocated budget

Algorithm 2: MCKS Deadline Constrained Algorithm
Input: JobSet; time-price table

Output: Minimum cost for executing workflow
1: **for** stage s ∈ workflow w **do**
2: Construct class s consists of tuple (D_{si}, C_{si})
$1 \leq i \leq \sum_{r \in [1,n]} p^r$
3: **end for**
4: **for** each task $J_i \in$ stage s **do**
5: Assign execution time on the candidate machine into a Set S
6: **end for**
7: Sort S in increasing order
8: **for** each element $t_i \in S$ **do**
9: $D_{si} \leftarrow t_i$
10: **end for**
11: Compute C_{si} (D_{si}) for each task
12: Choose exactly one tuple from each class to minimize the total cost.

Algorithm 3: Dynamic Pricing Algorithm
Input: Job, MachineSet(Initial_Cost, Maintenance_Cost, Power_Cost, Processing_Speed), Task_Size
t_p = Time for planning and allocating machine

Output: PriceSet and TimeSet for task

1: Estimate price-demand function Z(m, tp , w)
2: **if** (Capacity > Z) **then**
3: TaskPrice = TaskPrice - (Capacity / Z)
4: Allot capacity to incoming demand
5: **else**
6: Select $\max_{p \in Pw,m}$ p
where p = TaskPrice, $P_{w,m}$ = PriceSet for tasks which are demanding resource
7: TaskPrice = TaskPrice + (CapacityLeft / Z)
8: **end if**
9: Estimate revenue of CSP = $\max_{p \in Pw,m}$ p. Z(m, tp , w)
10: Estimate user gain.

Modules

The system is divided into multiple sub-systems which performs its functions as mentioned below.

The first module of the system is User side. User submits their jobs to cloud environment. The complexity of cloud system is abstracted from user. The second module is dynamic pricing module which makes the time-price table dynamic by updating its values considering different parameters of the machine on which the task can be scheduled. The third module of the system is Budget and Deadline Distribution Solver (BDS). The BDS takes as an input a batch of jobs that are arranged as a multi-stage fork and join workflow by the scheduler at execution time. Every task of the job is associated with a time-price table As a result; the BDS can be configured with various parameters, including time-price tables, the job set and the total number of jobs in the workflow. The goal is to make the total scheduling length of the workflow minimum without breaking the budget and the total monetary cost of the

workflow minimum without missing the deadline. The result of these sub-systems is aggregated to obtain the final result i.e. total makespan and total cost.

Experimental Results

The system is being developed using eclipse with MapReduce for Hadoop operations. PHP/MySQL with Apache for load injection in the cloud. The system does not require any specific hardware to run. The experiments have been conducted on machine having following configurations: Intel i5 quadcore processor with 2.67GHz CPU, 4GB RAM. We have taken google4me cloud dataset consisting of a Job dataset, Machine Set. Job Set is obtained from user which is split into Task Set in the cloud. Task Set is maintained by cloud administrator which contains multiple attributes. Machine Set contains various details about machines in the cloud such as costs, configuration etc.

TABLE II : Sample DataSet for jobs and tasks entered into the cloud.

Job ID	Job Name	Task ID	Task Name	Task Size (in kb)	Time	Static Price	Dynamic Price
1	Read Files	1	Read PDF	34.76	0.337	0.71	0.4
		2	Read TXT	36.98	0.41	0.61	0.2
2	Browse Web	1	Url	102.12	1.29	1.9	2.15
		2	Search Text	130.98	0.67	1.72	2.38
		3	Search Images	212.49	2.64	0.94	1.44

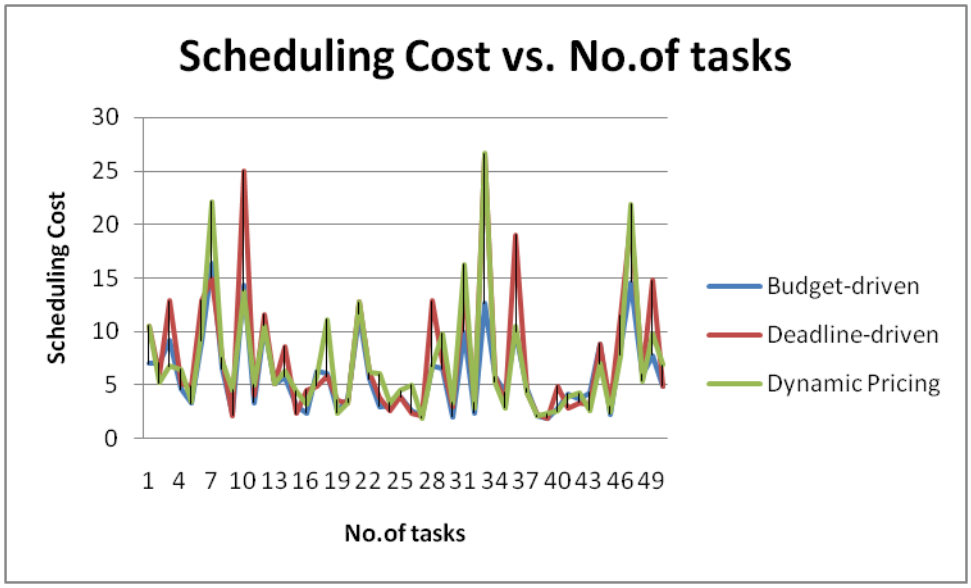

Fig 3: Comparison between Budget-driven, Deadline-driven and Dynamic Pricing Scheduling technique in terms of Cost.

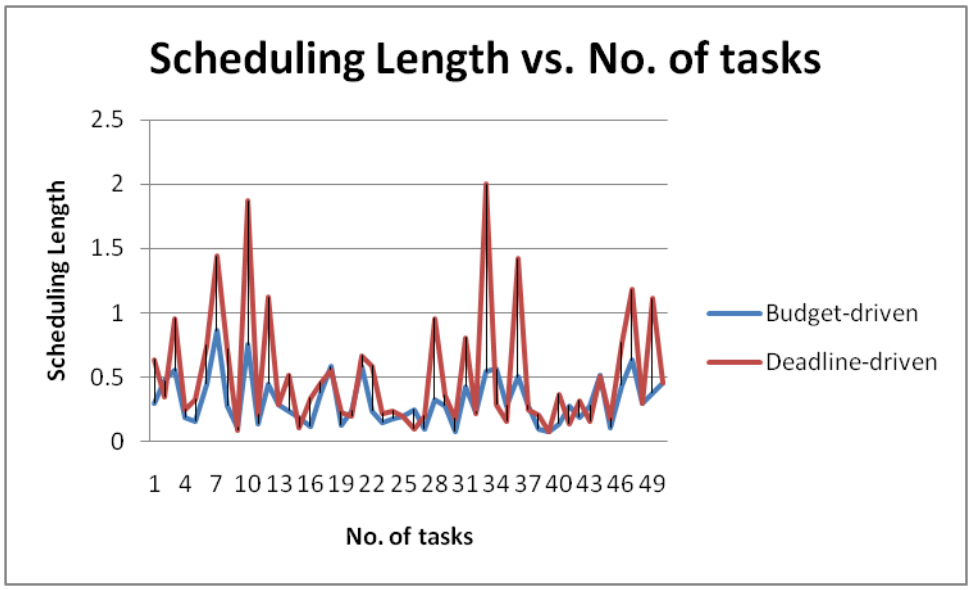

Fig 4: Scheduling Length of the algorithms vs. No. of tasks

Figure 3 shows the comparison between state-of-the-art algorithm and proposed algorithm. The plots are the scheduling cost verses no. of tasks. This graph supports the claim that dynamic pricing algorithm, the proposed algorithm provides gains to user as well as CSP in terms of cost. Figure 4 shows the scheduling length for various algorithms. Scheduling Length gives the total time required for scheduling the task. This time is the estimated minimum time. The plots are the scheduling length verses the no. of tasks. The blue line is Budget-driven algorithm and the red line is Deadline-driven algorithm and green line is Dynamic Pricing algorithm.

The scheduling results obtained by the deadline and cost constrained scheduling technique, describes the relation between the tasks and their scheduling length and scheduling cost. This results shows that dynamic pricing provides the real time scenario of market conditions. Lower price is set to attract customers, whereas higher price is set to attract more revenue The success definition of this scheduling technique lies in dynamically controlling the unpredictable demand of customers.

Conclusion and future scope

Large number of jobs arrive into the cloud, to handle these jobs, scheduling provides a smarter way of placing these jobs on cloud resources. In this paper, we came up with a novel approach for dynamically changing the price charged for user's job execution which is beneficial to both user and CSP, after that applying cost and deadline constrained scheduling. Microsoft Azure views dynamic pricing as a feasible option that will be highly adopted in addition to static pricing. Google AdWorks and Facebook Ads uses bidding/auctions strategy which is efficient but doesnot considers market conditions (demand and supply). This implicates the necessity of dynamic pricing. The advanced features in the scheduling (speculative task scheduling, redundant computing for fault tolerance) with respect to the budget constraints can be considered in future work. Enforcing the full system in a real cloud computing environment (Eg. Amazon) can also be tackled as future work. The market competition between clouds can also be taken as future enhancement.

References

[1] Hadoop's Fair Scheduler, http://hadoop.apache.org/docs/r1.2.1/fair_scheduler.html

[2] Hadoop's Capacity Scheduler, http://hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html

[3] Mark Yong, Nitin Garegrat, Shiwali Mohan: "Towards a Resource Aware Scheduler in Hadoop" in Proc. ICWS, 2009, pp:102-109.

[4] Thomas Sandholm and Kevin Lai, "Dynamic Proportional Share Scheduling in Hadoop", E. Frachtenberg and U. Schwiegelshohn (Eds.): JSSPP 2010, LNCS 6253, pp. 110–131, 2010. c_Springer-Verlag Berlin Heidelberg 2010.

[5] Arun Anandasivam, Marc Premm, "BID PRICE CONTROL AND DYNAMIC PRICING IN CLOUDS", Association for Information Systems AIS Electronic Library (AISeL) ECIS 2009 Proceedings European Conference on Information Systems (ECIS).

[6] Saravanan. K and Sri Vigna Hema. V, " Dynamic Pricing Model for a Cloud Cache Environment", International Journal of Engineering Trends and Technology (IJETT) - Volume4Issue4- April 2013.

[7] May Al-Roomi, Shaikha Al-Ebrahim, Sabika Buqrais and Imtiaz Ahmad, " Cloud Computing Pricing Models: A Survey", International Journal of Grid and Distributed Computing Vol.6, No.5 (2013), pp.93-106 http://dx.doi.org/10.14257/ijgdc.2013.6.5.09.

[8] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," Sci. Program., vol. 14, no. 3-4, pp. 217–230, Dec. 2006.

[9] Dr. Amit Agarwal, Saloni Jain, "Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment", International Journal of Computer Trends and Technology (IJCTT) – volume 9 number 7– Mar 2014.

[10] Anthony T. Velte, Toby J. Velte, Ph.D., Robert Elsenpeter , Cloud Computing: A Practical Approach, 2010 The McGraw-Hill Companies.

[11] Peter Mell , Timothy Grance ,The NIST Definition of Cloud Computing, National Institute of Standard and Technology U.S. Dept. of Commerce.

[12] Xiaocheng Liu, Chen Wang, Bing Bing Zhou, Junliang Chen, Ting Yang, and Albert Y. Zomaya, "Priority-Based Consolidation of Parallel Workloads in the Cloud", IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 24, NO. 9, SEPTEMBER 2013.

[13] Dzmitry Kliazovich, Sisay T. Arzo, Fabrizio Granelli, Pascal Bouvry and Samee Ullah Khan, "e-STAB: Energy-Efficient Scheduling for Cloud Computing Applications with Traffic Load Balancing", 2013 IEEE International Conference on Green Computing and Communi- cations and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing.

[14] Abhishek Verma, Ludmila Cherkasova, and Roy H. Campbell, "Orchestrating an Ensemble of MapReduce Jobs for Minimizing Their Makespan", IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 10, NO. 5, SEPTEMBER/OCTOBER 2013.

[15] Yang Wang and Wei Shi, "Budget-Driven Scheduling Algorithms for Batches of MapReduce Jobs in Heterogeneous Clouds", IEEE TRANSACTIONS ON CLOUD COMPUTING, VOL. 2, NO. 3, JULY-SEPTEMBER 2014.