OPTIMIZED SCHEDULING OF TASKS USING HEURISTIC APPROACH WITH COST-EFFICIENCY IN CLOUD DATA CENTERS

G.Ramya[1], Dr P. Keerthika[2], Dr P. Suresh[3] and Ms M. Sivaranjani[4].

[1]PG Scholar, Department of CSE, Kongu Engineering College, Perundurai-638052, Erode, Tamil Nadu, India

[2]Assistant Professor (Sr.G), Department of CSE, Kongu Engineering College, Perundura-638052,Erode, Tamil Nadu

[3]Assistant Professor (Sr.G), Department of IT, Kongu Engineering College, Perundura-638052, Erode, Tamil Nadu

[4] Assistant Proferssor, Department of CSE, Kongu Engineering College, Perundurai-638052, Erode, Tamil Nadu

ABSTARCT:

Cloud computing is a promising approach to execute large programs, As this class of programs may be decomposed into multiple sequence of tasks that can be executed on multiple virtual Machines, the execution of the tasks can be viewed as a Directed Acyclic graph (DAG). In DAG, nodes are the tasks and edges are the precedence constraints between tasks. Users of cloud pay for what their programs actually consume according to the pricing models of the cloud providers. Earlier task scheduling algorithms are mainly focused on minimizing the makespan, but not the mechanisms to reduce the monetary cost incurred in the settings of cloud. The proposed scheduling algorithm mainly focuses on Cost-efficiency and it uses two heuristic methods. The First method dynamically maps task to the most cost-efficient VMs based on the concept of Pareto dominance. The second method, a complement to the first method, reduces the monetary costs of non-critical tasks.The simulation results show that our algorithm can substantially reduce monetary cost while producing makespan as good as the best known task-scheduling algorithm can provide.

Keywords—minimizing makespan,reducing monetary cost, virtual machine, Task scheduling.

I. Introduction

Cloud computing is Internet connected mode of supercomputing. It is a type of shared infrastructure, whichsimply puts the huge system pools together by using various means: distributed, virtualization etc. It gives users a variety of storage, networking and computing resources in the cloud computing environment via Internet, users put a lot of information and accesses a lot of computing power with the help of its own computer. The goal of cloud computing is to manage computing power, storage, various kind of platforms and services which is assigned to the external users on demand through the internet.

The scheduling of tasks in cloud means choosing the best suitable resource available for execution of tasks or to allocate computer machines to tasks in such a manner that the completion time is minimized as possible. In scheduling algorithms list of tasks is created by giving priority to each and every tasks where setting of priority to different taskscan be based on various parameters. Tasks are then chooses according to their priorities and assigned to available processorsand computer machines which satisfy a objective function.Task scheduling problem can be viewed as weighted directed acyclic graph (DAG). It is a directed graph with no directed cycles, formed by a collection of vertices and directed edges, each edge connecting one vertex to another. The vertex represents a task and its weight represents the size of task computation. Arc represents the communication among the two tasks and weight represents the communication cost. The directed edge shows the dependency between the two tasks. The main goal of the task scheduling is to schedule the tasks on processors to minimize make-span. Make-span is defined as the completion time of the last task relative to the start time of the first task.

II. RELATED WORK

M.Wieczorek et al.,[15],examined three scheduling algorithms to evaluate their performance for scheduling scientific workflows in the cloud environment. The scheduling algorithms comprise a genetic algorithm similar to one presented in the well-known HEFT algorithm and a "myopic" algorithm. That the HEFT algorithm is more effective and less time consuming than the genetic algorithms. HEFT also performs substantially better than a simple myopic algorithm. HEFT algorithm appears to perform best for unbalanced workflows.

R.Buyya et al.,[2], presented a cost-based workflow scheduling algorithm for time-critical workflow applications. The processing time and execution cost are two typical QoS constraints for executing workflows on "pay-per-use" services. The users normally would like to get the execution done at lowest possible cost within their required timeframe. That allows the workflow management system to minimize the execution cost while delivering results within a certain deadline. A Markov Decision Process approach to schedule sequential workflow task execution, such that it can find the optimal path among services to execute tasks and transfer input/output data.

K.Kurowski et al., [10]focused on two different models of Grid resource management problems: one of the problem is grid scheduling problems with no time characteristics available, and the other is scheduling of jobs in presence of time characteristics achieved by using some prediction techniques, and resource reservation mechanisms. That is modeled as multi-criteria decision support problems. Multi-criteria methods may increase a total satisfaction of stakeholders taking part in Grid resource management.

R.Buyya et al.,[2]focused on interconnecting clouds for dynamically creating an environment which identifies various computing paradigms promising to deliver the vision of computing utilities , provides the architecture for creating market-oriented Clouds by leveraging technologies such as VMs, provides thoughts on market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain SLA-oriented resource allocation; presents some representative Cloud platforms especially those developed in industries along with the current work towards realising market-oriented resource allocation of Clouds by leveraging the third generation Aneka enterprise Grid technology, reveals the early thoughts on interconnecting Clouds for dynamically creating an atmospheric computing environment along with pointers to future community research and concludes with the need for convergence of competing IT paradigms.

Nephele et al.,[12] to integrate frameworks for parallel data processing in their product portfolio, making it easy for customers to access these services and to deploy their programs. It is the first data processing framework to explicitly exploit the dynamic resource allocation offered by today's compute clouds for both, task scheduling and execution. It allows assigning the particular tasks of a processing job to different types of virtual machines and takes care of their instantiation and termination during the job execution. Based on this new framework, we perform evaluations on a compute cloud system and compare the results to the existing data processing frameworks.

Pregal et al., [11] focused onmany practical computing problems concern large graphs. Standard examples include the Web graph and various social networks. The scale of these graphs, in some cases billions of vertices, trillions of edges poses challenges to their efficient processing. In this paper they proposed a computational model suitable for this task. Programs are expressed as a sequence of iterations, in each of which a vertex can receive messages sent in the previous iteration, send messages to other vertices, and modify its own state and that of its outgoing edges or mutate graph topology. This vertex centric approach is flexible enough to express a broad set of algorithms. The model has been designed for efficient, scalable and fault-tolerant implementation on clusters of thousands of commodity computers, and its implied synchronicity makes reasoning about programs easier. Distribution related details are hidden behind an abstract API. The result is a framework for processing large graphs that is expressive and easy to program.

III. METHODOLOGY

A. APPLICATION MODEL

Let G=(V,E) be a DAG, where V is the set of v tasks to be executedand E is the set of e edges representing the precedence constraints between tasks.Assume that G has a task without any predecessors, call it an entry task and denote it by v_{entry}.The weight of a node v_i,denoted by dt_{vi}, represents the computation load for task v_i. Cloud computing environment consists of a set of m fully connected heterogeneous VMs, denoted by M.Let ca_{mj} denote the CPU cycles allocated to VM m_j. Each task can be executed on a different VM, and it is denoted by $t(v_i,m_j)$and the execution time of the task v_i on VM m_j is given as,

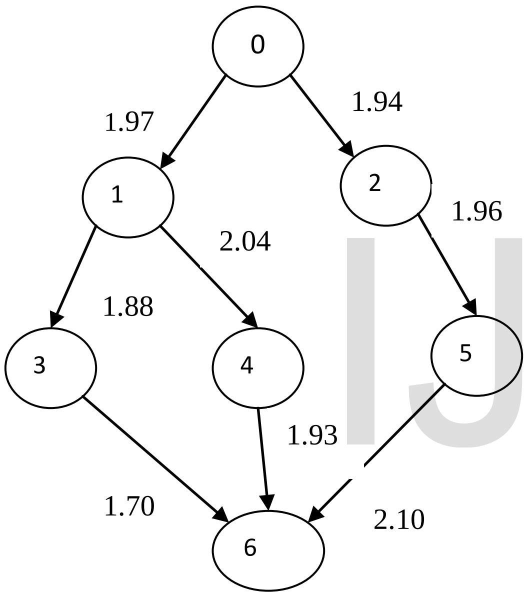

Figure 1. Task Graph

$$t(v_i,m_j) = \frac{dt_{v_i}}{ca_{m_j}} [1]$$

The average execution time of the task $v_i$is given in Equation 2 as,

$$\overline{T_{V_i}} = \sum_{j=1}^{m} \frac{t(v_{i,},m_j)}{m} \quad [2]$$

The weight of edge (v_i, v_k), denoted by $ct_{vi,vk}$represents the communication time between task $v_i$and task v_k..

Table 3.1 Execution time and priority of tasks

TASK	VM1	VM2	VM3	PRIORITY
0	9.26	11.01	13.79	51.95
1	9.2	10.5	13.9	38.47
2	10.05	12.12	13.15	38.66
3	9.54	10.41	15.4	24.81
4	10.33	11.51	14.08	25.23
5	9.36	11.1	14.03	24.93
6	9.39	10.73	13.88	11.33

Thepriority of the task $v_i$is computed by traversing the DAG upward,starting from the exit task, and is defined in Equation 3.

$$P_{v_i} = \bar{T}_{v_i} + \max_{v_k € succ(v_i)} \left(ct_{v_{i,v_k}} + P_{v_k} \right) \quad [3]$$

Where succ $(v_i)$is the set of successors of task v_i. The value of P_Vk is the priority of immediate successors of task v_i..The priority is computed by traversing the task graph upward, the priority of exit task is defined in Equation 4,

$$P_{v_{exit}} = \bar{T}_{v_{exit}} [4]$$

B. CLOUD RESOURCES MODEL

Cloud providers offer different types of VMs for different types of workload. These VMs have different processing capacities and pricing models.In particular, we use a linear pricing model and an exponential pricing model. In the linear pricing model, the cost of using VM is linearly correlated with the number of CPU cycles. The total monetary cost is computed by

$$c = \sum_{j € select} c(v_i, m_j) \quad [5]$$

IV. PROPOSED WORK

Cost efficient task scheduling algorithm consists of two scheduling heuristics.The First heuristic uses the concept of Pareto dominance to generate a cost efficient task schedule based on the execution time of the tasks and the monetary charges of VMs. The second heuristic complements to the first heuristic and attempts to minimize the monetary costs of non-critical tasks.

The following minimization problem with an objective function for each node $v_i \in$ V as a convex combination of running time and monetary cost:

$$Minimize: \alpha * T(i,j) + (1-\alpha) * c(i,j)\ for\ all\ m_j \epsilon M \quad [6]$$

Subject to: $T(i,j) = \frac{t(v_j,m_j)-t_{min}}{t_{max}-t_{min}}$ [7]

$$c(i.j) = \frac{c(v_i,m_j)-c_{min}}{c_{max}-c_{min}} [8]$$

$$\alpha \in [0,1] \quad [9]$$

Where

α -cost efficient factor that represents the userpreference for the execution time and monetary cost.

t(i,j) and c(i,j)- represents cost-efficiency ratios of time and costs.

$t_{min(max)}$- represents the minimum maximum execution time

$c_{min(max)}$-represents the minimum maximum monetary cost.

4.1 PARETO OPTIMALSCHEDULING HEURISTIC(POSH)

POSH is a heuristic approachto dispatch tasks in a DAG to the cost-conscious VMs based on Pareto dominance.

POSH involves the following three phases:

(1)Weighting Phase: In this phase, weights are assigned to the nodes and edges in the workflow. The weights assigned to nodes are calculated based on the predicted execution time of tasks and the weights assigned to edges are calculated based on predicted time of data transferred between the VMs.

(2)Prioritizing Phase: In this phase,a sorted list of tasks is created in the order how they should be executed. The priority of each task is to be set with the upward priority value, which is equal to the weight of the node plus the execution time of the successors. The task list is generated by sorting the tasks by the descending order of priority.

(3)Mapping Phase: In this phase, the tasks are assigned to the resources based on Pareto dominance. Consecutive tasks are mapped to the resources based on the priority queue. For each task, choose the VM that favours scheduling tasks with low monetary cost to run it. This is done by the pre-defined objective function.

SLACK TIME SCHEDULING HEURISTIC (STSH)

To reschedule non-critical tasks for reducing monetary costs, compute the slack time for the non-critical tasks. The slack time for a non-critical task Ti can be calculated by

$$T_{slack}(vi) = \min_{v_j \in pred(v_i)} \left(LFT(v_j)\right) - EST(v_i) \quad [10]$$

STSH calculates the slack time for each non-critical task and then reschedules it to the idle VM with the minimum monetary cost.

V. RESULTS AND DISCUSSION

The performance of Hybrid algorithm and the proposed HEFT algorithm with respect to the makespan and monetary cost are evaluated using cloudsim. Results obtained with the use of the same machine configurations.

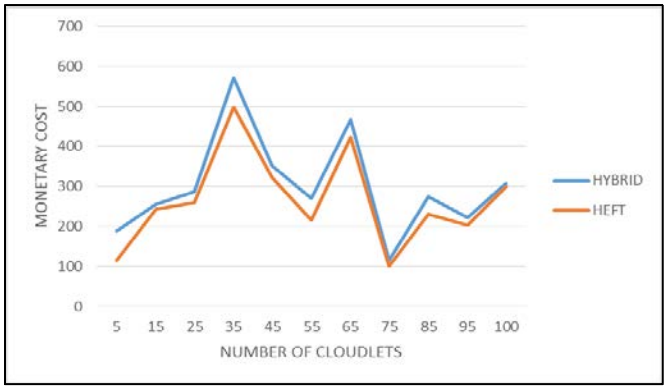

Figure. 2 Monetary cost Result

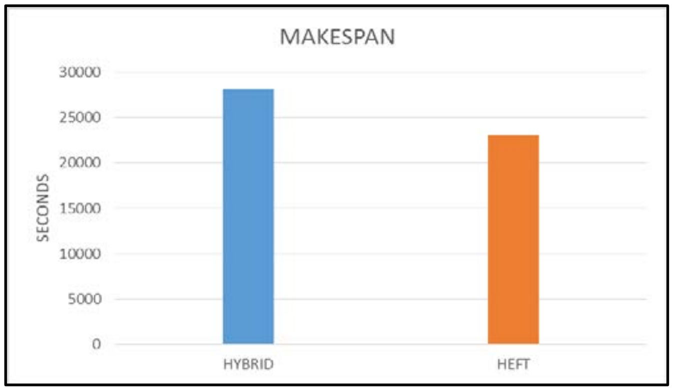

Figure. 3 Makespan Result

We normalize the makespan and monetary cost and call them as Schedule Length ratio (SLR) and monetary cost ratio(MCR) as follows:

$$SLR = \frac{makespan}{\sum_{v_i \in cp} min_{m_j \in M}\{t(i,j)\}} [11]$$

$$\text{MCR} = \frac{C}{\sum_{vi \in cp} min_{mj \in M}\{C(i,j)\}} [12]$$

The input cost-efficient factor α to measure the makespan and the monetary cost. Fig 2 shows the monetary cost result of a random graph with 100 tasks, fig 3 shows the makespan result. There is a tradeoff between the makespan and monetary cost.

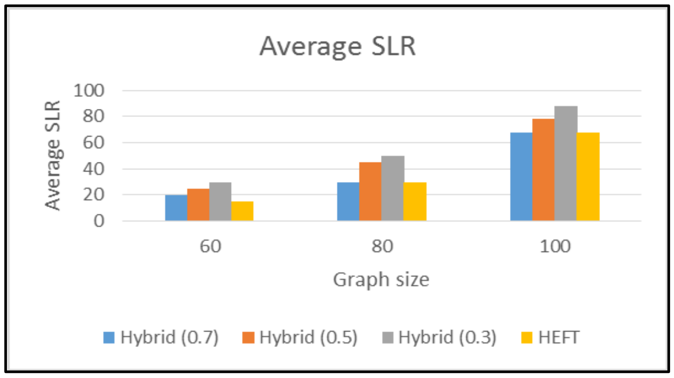

Figure. 4 Average SLR

When the cost-efficient factor α = 0.7, Hybrid incurs 34.98 percent less monetary cost than HEFT on average, with almost the same makespan. This implies that Hybrid performs well in a cloud setting.

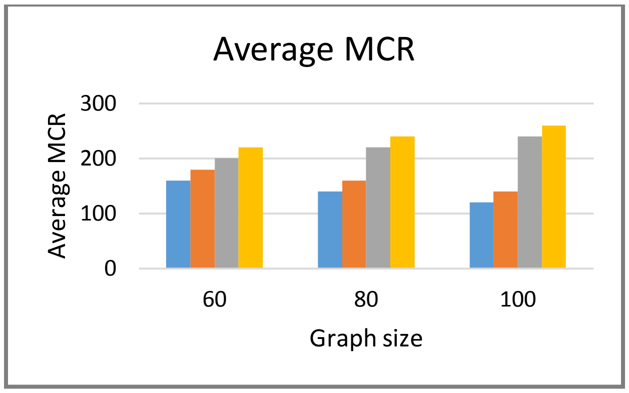

Figure. 5 Average MCR

When α = 0.5, Hybrid incurs 38.25 percent less monetary cost than HEFT on average, with about 13.18 percent makespan extension.
When α = 0.3, Hybrid incurs 41.26 percent less monetary cost than HEFT on average, but with 32.65 percentage more makespan.

VI. CONCLUSION AND FUTURE WORK

In cloud computing, task scheduling helps to choose the best suitable resources available for execution of tasks. Most conventional task scheduling algorithms do not consider monetary costs, so they cannot be directly applies in a cloud setting. Hence, a new task scheduling algorithm is devised for running large programs in the cloud. The proposed algorithm computes scheduling plans to produce efficient makespan and to reduce the monetary cost.

As a future work, load balancing can also be considered for allocating tasks to VMs and new optimization techniques can be used for efficient scheduling and penalties can be incorporated for who violates the consumer-provider contracts.

REFERENCES

[1] Armbrust M, Fox A, Griffith R, Joseph A, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, (2009), Above the clouds: 'a berkeley view of cloud computing', Tech. rep, Technical Report UCB/EECS, EECS Department, University of California, Berkeley.

[2] Buyya R, Yeo C, Venugopal S, Broberg J, Brandic I, (2009),'Cloud computing and emerging it platforms: Vision hype and reality for delivering

computing as the 5th utility', Future Generation computer systems 25 (6) 599–616.

[3] Bajaj R, Agrawal D, (2004), 'Improving scheduling of tasks in a heterogeneous environment', IEEE Transactions on Parallel and Distributed Systems 15 (2) 107-118.

[4] Bozdag D, Ozguner F, Catalyurek U, (2009), 'Compaction of schedules and a two-stage approach for duplication-based dag scheduling', IEEE Transactions on Parallel and Distributed Systems 20 (6) 857–871.

[5] Braun T, Siegel H, Beck N, Boloni L, Maheswaran M, Reuther A, Robertson J, Theys M, Yao B, Hensgen D, (2001), 'A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems ', Journal of Parallel and Distributed computing 61 (6) 810–837.

[6] Hou E, Ansari N, Ren H, (1994),' A genetic algorithm for multiprocessor scheduling', IEEE Transactions on Parallel and Distributed Systems 5 (2) 113–120.

[7] Isard M, Budiu M, Yu Y, Birrell A, Fetterly D, Dryad,(2007): 'Distributed data-parallel programs from sequential building blocks', ACM SIGOPS Operating Systems Review 41 (3) 59–72.

[8] Kwok Y, Ahmad I, (1996), 'Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors', IEEE Transactions on Parallel and Distributed Systems 7 (5) 506–521.

[9] Kwok Y, Ahma I, (1999) 'Static scheduling algorithms for allocating directed task graphs to multiprocessors', ACM Computing Surveys (CSUR) 31 (4)406–471

[10] Kurowski K, Nabrzyski J, Oleksiak A, Weglarz J , (2006) , 'Grid multicriteria job scheduling with resource reservation andprediction mechanisms, Perscpectives in Modern Project Scheduling , pp. 345-373.

[11] Li J, Su S, Cheng X, Huang Q, Zhang Z, (2011), ' Cost-conscious scheduling for large graph processing in the cloud', IEEE 13th International Conference on High Performance Computing and Communications (HPCC), pp. 808–813.

[12] Alewicz G, Austern M, Bik A, Dehnert J, Horn I, Leiser N, zajkowski, Pregel,(2010) : 'a system for large-scale graph processing', in Proceedings of the International Conference on Management of Data, ACM, pp.135–146.

[13] Nephele, Austern M, Kao O, (2009),' Efficient parallel data processing in the cloud', Many –Task Computing on Grids and Supercomputers, ACM,p.8.

[14] Topcuoglu H, Hariri S, Wu M, (2002), 'Performance-effective and low complexity task scheduling for heterogeneous computing', IEEE Transactions on Parallel and Distributed Systems 13 (3) 260–274

[15] Ullman J, (1997), 'Np-complete scheduling problems ', Journal ofComputer and System Sciences 10 (3) 384–393.

[16] Wieczorek M, Prodan R, Hoheisel A , (2009), 'Towards a general model of multi-criteria workflow Scheduling on the grid, Future Generation Computer Systems 25(3) 237-256.

.