Job Scheduling Based on Virtual Abstractions in Cloud

B.Parkavi[#1], G.Malathy[*2]

[#]PG Scholar, [*]Research Scholar

Department of Computer science and Engineering,

K S R Institute for Engineering and Technology,

Tiruchengode, Namakkal-637215, India

***Abstract*-- Cloud computing is a provisioning of services in a timely, on-demand manner, to allow the scaling up and down of resources. Job scheduling is one of the major issues in the public cloud which concerns availability of resources in the datacenter. Data center need to achieve certain level of utilization of its nodes while maintaining level of responsiveness of parallel jobs. Existing scheduling schemes make use of backfilling strategies which pre-empt shortest jobs to execute when jobs at head of the queue have unavailable of resources. This results in starvation of larger jobs, reduced throughput and underutilization of resources. In this paper, job scheduling based on virtual abstraction scheme is proposed for efficient scheduling of jobs in k- cloud data center with multiple computing capacities which solves large-scale static scheduling problem in cloud.**

***Keywords*-- Abstraction scheduler, cloud computing, parallel workload, virtual machine**

I. INTRODUCTION

Cloud computing is a model for enabling convenient, on-demand 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 [1]. Client can access resources pooled in the cloud by requesting cloud service providers and based on client's request, resource is provisioned to the client by pay-per-usage demand. During resource provisioning, there may occur delay of response from cloud service provider since, cloud is a way of distributed computing, some other clients may request the same resource (or) the server is busy with its resource allocation. Hence there is a need for scheduling based on client's request and availability of resources in the datacenter. Resource scheduling problem is similar to Banker's algorithm which prevents deadlock by denying or postponing the request if it determines that accepting the request could put the system in an unsafe state. When a new process enters a system, it must declare the maximum number of instances of each resource type that may not exceed the total number of resources in the system.

Scheduling in distributed systems is spreading the load on processors and maximizing their utilization while minimizing the total task execution time. Fig 1 shows an outline of job scheduling in which the resources are allocated from datacenter to the client. Hence, in this paper, we proposed an Abstraction scheduler for scheduling the tasks regarding the requested job and the computational resources needed by it. Allocating the jobs to idle and appropriate data center node by reducing the execution time and improving parallel job responsiveness is the major role behind this proposed system.

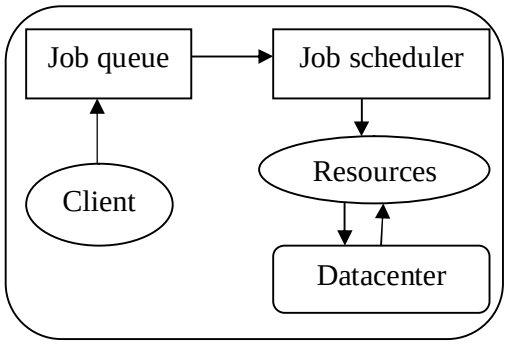

Fig 1: Job scheduling

Clients from various locations will be assigned in a job queue based on their needs. Job scheduler is responsible for allocation of required resources to the client which is rendered from datacenter [2]. Cloud service provider acts as intermediate between the datacenter broker and client. The availability of resources is checked dynamically and idle resources are allocated towards the client based on their needs. This scheduling of resources to the client is the major issue in cloud despite time and efficiency.

II. RELATED WORK

U. Schwiegelshohn and R. Yahyapour [3] has analysed the working of First-Come-First-Serve(FCFS) algorithm in which each job specifies the number of nodes required and the scheduler will processes those jobs in the order of their arrival. When there is a sufficient number of nodes to process the job then the scheduler dispatches the job to run on these nodes else it waits for the currently running job to finish. So it causes fragmentation of nodes and delay in getting resources.

D. Feitelson and M. Jettee [4] have proposed that gang scheduling improves node utilization and responsiveness over

parallel jobs. It allows sharing of resources among multiple parallel jobs in which the computing capacity of a node is divided into time slices. The allocation of time slices of different nodes to parallel processes is coordinated by OS support. It manages to make all the processes of a job progress together so that one process will not be in sleep state when another process needs to communicate with it. So it stretches the execution time of individual jobs.

Wiseman and D. Feitelson [5] has proposed that Paired gang scheduling tries to overcome the drawbacks of gang scheduling in which it utilizes the system resources well without causing interference between the processes of competing jobs. The processes will not have to wait much because a process which occupies the CPU most of the time will be matched with a process that occupies an I/O device most of the time, so they will not interfere with each other's work. On the other hand, the CPU and the I/O devices will not be idle while there are jobs which can be executed.

Y. Zhang, H. Franke, J. Moreira, and A. Sivasubramaniam [6] have proposed an effective scheduling strategy to improve response time, throughput, and utilization of resources in cloud. Gang-scheduling and backfilling are two optimization techniques that operate on orthogonal axes, space for backfilling and time for gang scheduling and the proposed technique is made by treating each of the virtual machines created by gang-scheduling as a target for backfilling. The difficulty arises in estimating the execution time for parallel jobs so migration is taken into account which improves the performance of gang-scheduling without the need for job execution time estimates.

Xiaocheng Liu, Chen Wang, Bing Zhou, Junliang Chen, Ting Yang, and Albert Y. Zomaya [7] has proposed CMCBF algorithm which overcomes the drawbacks of gang scheduling algorithm. It ensures a job to run in foreground VMs whenever the number of foreground VMs that are either idle or occupied by jobs arriving later than it satisfies its node requirement. It also allows jobs to run in background VMs simultaneously with those foreground VMs to improve node utilization. Shachee V Parikh and RichaSinha [8] has proposed a double level priority based task scheduling in which three different waiting-queues are considered such as low-priority queue, medium-priority queue and high-priority queue and the local scheduler maintains these queues. The scheduler needs to effectively schedule tasks in terms of both performance and energy consumption. For this, power-threshold of processor is monitored. When a processor reaches its power threshold, the task is assigned into another processor.

Hence an abstract representation of resource availability in the virtual machines of the concerned datacenters is known to effectively satisfy the client needs.

III. PROPOSED WORK

Virtual abstraction based job scheduling scheme is proposed for efficient scheduling of jobs in k- cloud datacenter with multiple computing capacity by using abstraction refinement. Scheduler based on abstraction first attempts to solve scheduling problem with abstract representations of job and computing resources. With small abstract representations, scheduling is done fast. If obtained schedule does not meet specified quality, then scheduler refines job and datacenter abstractions and again solves scheduling problem.

Model a job as dataflow of tasks and data center as set of computation nodes connected by communication links. Scheduling concerned assigning nodes and time intervals to tasks in a job [9]. Tasks start at a time only when all its preceding tasks are finished and task's inputs are available at the assigned node. Job abstractions represent pieces of computation (tasks) and data transferred between tasks (objects).Each task has an associated duration and each object has an associated size. Abstract job is obtained from concrete job by grouping together tasks to abstract tasks and ignoring data dependencies between tasks in each group.

A. Cloud and parallel workloads

Cloud computing provide cost-effective solution for running business applications through virtualization, scalable distributed computing, data management and pay-as-you-go pricing model. Data center handles applications with high-performance computing needs and runs parallel jobs most of the time [10]. Parallel workload requires a certain number of data center nodes to run which are fragmented by parallel jobs with different node number requirements. If number of available nodes cannot satisfy requirement of an incoming job then nodes remain idle.

Parallel programming involves computing, communication, and synchronization phase. Process in a parallel job frequently waits for data from other processes. During waiting utilization of the node is low. Batch scheduling algorithm for parallel jobs FCFS cause node fragmentation. Backfilling and Gang scheduling minimize node fragmentation but utilization degradation caused by parallelization.

B. Workload consolidation

Two workload consolidation experiments conducted to improve node utilization and examine impact to execution time of parallel jobs [11]. Collocate two VMs in each physical node with same priority and two VMs with different priorities,

- one is assigned a weight of 10,000 and
- other is assigned a weight of 1

High-priority VM is foreground VM andlow-priority one background VM [7]. Background VM only runs, when foreground VM is idle. When a foreground VM runs a job with CPU utilization higher than 96 percent, collocating a

VM to run in background does not benefit due to context switching incurs overhead and background VM has very small chance to get physical resource to run. When foreground VM runs a job with low CPU utilization, job running in collocated background VM get significant share of physical resources to run. When a background job departs, the scheduler scans queue according to the job arrival time and place a matching job to run in available background VMs.

C. Abstraction scheduler

Job abstractions represent pieces of computation (tasks) and data transferred between tasks (objects). Each task has an associated duration and each object has an associated size. Figure.3 shows system flow diagram. Abstract job is obtained from concrete job by grouping together tasks such as abstract tasks (called blocks) and ignoring data dependencies between tasks in each group [12]. Duration of a block is the maximal duration of represented concrete tasks. Instance of generic abstraction refinement scheduler provides own implementation of subroutines,

- initial abstraction
- schedule
- refine

Abstraction scheduler keeps track of free intervals on all computing nodes in data center and uses information to schedule blocks of tasks in job. Independent tasks in a parallel job are scheduled simultaneously which leads to idea of cost of maintaining set of free intervals on nodes in data center. It presented an inverted indices used in search algorithms which maintains a data structure and location of occurrences of data center nodes [13]. Success of an abstraction refinement depends on the quality of abstractions such that efficient captures of concrete instance without keeping track of much information.

D. K – Cloud Datacenter Scheduler

K-Tier data center scheduler starts with initial abstract job obtained from input job by using job duration abstraction. Initial abstract data center is obtained from input data structure by collapsing all computation nodes into a single node. Scheduler keeps job abstraction constant but refines data center abstraction as required. Memory allocator maintains a partition of the memory in order to find best suitable free memory block. Each refinement step splits some block into two new blocks. Partition is represented as a binary tree [12]. When an allocated memory is freed then compaction easily done by collapsing the tree. Best-fit allocation is used to schedule tasks from one job to nodes close to each other.Representation of data center changes with each allocation.

IV. SYSTEM ARCHITECTURE

Job scheduling is based on multiple computing resources and abstraction refinement. Once the parallel jobs get into the job queue, the abstraction scheduler will search for availability of needed process which is rendered to the parallel jobs despite CPU processing time. Fig2 shows the outline of system architecture. Initial abstract data center is obtained from input data structure by collapsing all computation nodes into a single node. Scheduler keeps job abstraction constant but refines data center abstraction as required. It will give the details of virtual machines regarding the process storage. In the k-cloud datacenter, datacenters are partitioned into clusters based on the location in the worldwide. It will fetch the resource location and if there is lack of process then it move from the nearest cluster to reduce the communication cost.

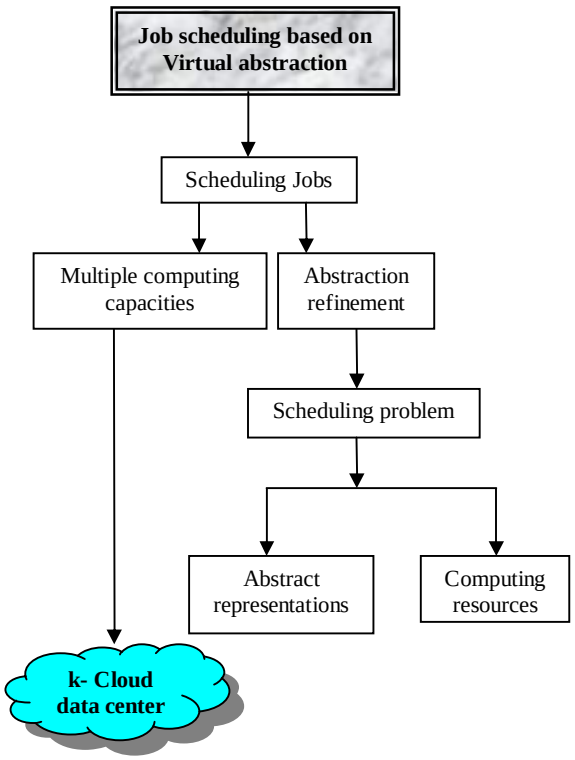

Fig 2: System Architecture

V. SIMULATION RESULT AND DISCUSSION

CloudSim-3.0 is used as a simulation tool with NetBeans 7.1 in this work. CloudSim is an extensible simulation toolkit that enables modeling and simulation n of Cloud computing systems and application provisioning environments. The CloudSim toolkit supports both system and behavior modeling of Cloud system components such as data centers, virtual machines (VMs) and resource provisioning policies. A dataset containing job list is given as input to the simulator and the job list contains user id, coordinate, process, bandwidth, required time and required instruction.

TABLE 1: Job List

Id	***Coordinate***	***Process***	***BW***	***Required time***	***Required instruction***
1	31-42-45-N	Google Nexus	454	48	454
2	35-56-37-S	Toshiba Canvas	387	71	321
3	45-65-48-N	SanDisk 8GB	423	13	113
4	75-50-48-N	Seagate backup	93	55	57
5	57-47-39-E	Lava iris	123	15	87
6	57-87-45-N	K7 security	394	22	97
7	36-45-57-S	USB shield	169	30	54
8	35-47-67-S	Titanic movie	444	66	95
9	57-42-81-N	hip hop songs	200	45	88
10	57-81-64-N	Word Power	615	77	34

TABLE 2: Process selection

User id	*Process id*	***Coordinate***	***Process***	***BW***	***Required time***
1	1	31-42-45-N	Google Nexus	454	48
2	4	35-56-37-S	Seagate backup	93	55
5	4	45-65-48-N	Seagate backup	93	55
3	9	75-50-48-N	hip hop songs	200	45
3	3	57-47-39-E	SanDisk 8GB	423	13
2	7	57-87-45-N	USB shield	169	30
7	7	36-45-57-S	USB shield	169	30
7	2	35-47-67-S	Toshiba Canvas	387	71

TABLE 3: CMCBF Scheduling

VMs	***Process id***	***Coordinate***	***Process***	***BW***	***User id***
VM1	1	31-42-45-N	Google Nexus	454	1
VM1	9	75-50-48-N	hip hop songs	200	3
VM3	3	57-47-39-E	SanDisk 8GB	423	3
VM4	4	35-56-37-S	Seagate backup	93	2,5
VM3	7	36-45-57-S	USB shield	169	2,7
VM3	2	35-47-67-S	Toshiba Canvas	387	7

Table 1 shows the dataset for job list containing process and its properties. Table 2 shows process id created for each user's request. A user may request for more than one process to execute his/her application. Table 3 shows CMCBF scheduling, where the VMs are shared among multiple users which in turn reduce the process waiting time and under-utilization of servers to a considerable amount.

TABLE 4: Abstraction Scheduler

VM id	***Process id***	***Coordinate***	***Process***	***BW***	***User id***
VM1	1 9	31-42-45-N 75-50-48-N	Google Nexus hip hop songs	454 200	1 3
VM3	3 7 2	57-47-39-E 36-45-57-S 35-47-67-S	SanDisk 8GB USB shield Toshiba Canvas	423 169 387	3 2,7 7
VM4	4	35-56-37-S	Seagate backup	93	2,5

TABLE 5: K-Cloud Datacenter Scheduler

Partition id	*VM id*	***Process id***	***Coordinate***	***Process moved to location***	***User id***
1	VM1	1 9	31-42-45-N 75-50-48-N	Nil	1 3
4	VM3	3 7 2	57-47-39-E 36-45-57-S 35-47-67-S	Nil Nil 58-67-28-N	3 2,7 7
2	VM4	4	35-56-37-S	Nil	2,5

Here processes 1 and 9 shared VM1; similarly processes 3, 7 and 2 are shared by VM3 which improves datacenter node utilization to a higher level. Table 4 shows the abstraction scheduler which depicts the process availability on corresponding virtual machines. Now the cloud is partitioned into clusters based on location coordinates. If there is a starvation of process then it is moved to the nearest cluster which will be time consuming and cost consuming location from the user's location as it is shown in table 5.

The performance comparison of proposed work with existing methods and later on comparison of different approaches is made using different performance metrics such as number of data centers, number of jobs, scheduler time intervals, scheduler memory rate, number of Process, and CPU cycles for abstraction scheduler. It is clear from the graph shown below in fig 3, the proposed work have higher performance ratio compared to the existing method.

VI. CONCLUSION AND FUTURE ENHANCEMENT

Since the proposed system is a virtual abstraction based scheduling of process to the client in a parallel processing. The K-Cloud partitioning of datacenter node for the allocation of jobs and computing resources which schedules effectively the parallel workloads in multiple cloud datacenters is solely based on the user's requests in a proactive manner rather than a reactive manner. So the performance has truly higher in the proposed model compared to the existing model on the basis of average response time, datacenter node utilization and job responsiveness.

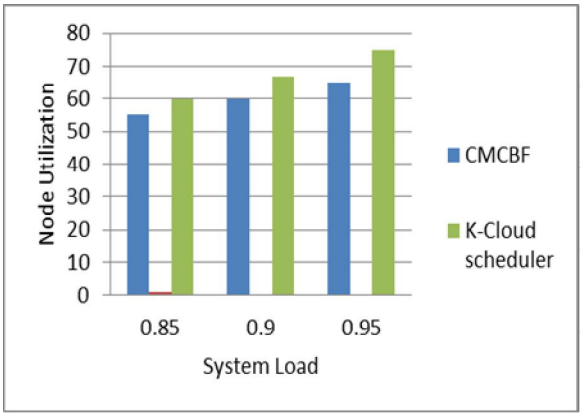

Fig 3: Node utilization

In our future work, we will exploit mechanisms that can effectively schedule the process among the intra-cluster of datacenter resources which may further improve the node utilization and responsiveness for parallel workload in the cloud.

REFERENCES

[1] Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J. and Brandic, I. ''Cloud computing and emerging IT platforms: vision, hype and reality for delivering computing as the 5th utility'', *Future Gener. Comput. Syst., 25(6)*, pp. 599–616 (2009).

[2] Peter Mell, Timothy Grance, "The NIST Definition of Cloud Computing", NIST (National Institute of Standards and Technology) Special Publication 800-145.

[3] U. Schwiegelshohn and R. Yahyapour, "Analysis of First-Come-First-Serve Parallel Job Scheduling," Proc. Ninth Ann. ACM-SIAM Symp. Discrete Algorithms, pp. 629-638, 1998.

[4] D. Feitelson and M. Jettee, "Improved Utilization and Responsiveness with Gang Scheduling," Proc. Workshop Job Scheduling Strategies for Parallel Processing, pp. 238-261, 1997.

[5] Y. Wiseman and D. Feitelson, "Paired Gang Scheduling," IEEE Trans. Parallel and Distributed Systems, vol. 14, no. 6, pp. 581-592, June 2003.

[6] Y. Zhang, H. Franke, J. Moreira, and A. Sivasubramaniam, "An Integrated Approach to Parallel Scheduling Using Gang-Scheduling, Backfilling, and Migration," IEEE Trans. Parallel and Distributed Systems, vol. 14, no. 3, pp. 236-247, Mar. 2003.

[7] Xiaocheng Liu, Chen Wang, Bing Zhou, Junliang Chen, "Priority-Based Consolidation of Parallel Workloads in the Cloud", IEEE Transactions on Parallel and Distributed Systems, Vol. 24, No. 9, September 2013.

[8] Shachee V Parikh, RichaSinha, "Double Level Priority Based Task Scheduling with Energy Awareness in Cloud Computing", International Journal of Engineering and Technology, 2011.

[9] D. Jackson, Q. Snell, and M. Clement, "Core Algorithms of the Maui Scheduler," Proc. Workshop Job Scheduling Strategies for Parallel Processing, pp. 87-102, 2001.

[10] J. Jann, P. Pattnaik, H. Franke, F. Wang, J. Skovira, and J. Riordan, "Modeling of Workload in Mpps," Proc. Workshop Job Scheduling Strategies for Parallel Processing, pp. 95-116, 1997.

[11] R. Fujimoto, A. Malik, and A. Park, "Parallel and Distributed Simulation in the Cloud," Int'l Simulation Magazine, Soc. for Modeling and Simulation, vol. 1, no. 3, 2010.

[12] T. A. Henzinger, V. Singh, T.Wies, and D. Zufferey, "Scheduling large jobs by abstraction refinement", in EuroSYS, pages 329–342, 2011.

[13] Thomas A. Henzinger, Anmol V. Singh, Vasu Singh, Thomas Wies, and Damien Zufferey, "Flex-PRICE: Flexible provisioning of resources in a cloud environment", IEEE International Conference on Cloud Computing, 2010.