Scientific Journal Impact Factor (SJIF) : 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745
ISSN (PRINT) : 2393 - 8161

Adaptive Algorithm and a Genetic Algorithm for Minimizing Cloud Task Length with Prediction Error

Sharfan P.S[1], Shahina K.K[2]
[1,2]*Department of Computer Science, IIET, Nellikuzhi*

Abstract-In a cloud data centre model, server has to process user request efficiently. So task execution length optimization required. But optimizing the task length is difficult due to the involvement of constraints like user payment and divisible resource demand. In Adaptive prediction (AP) method, a local optimal allocation algorithm (LOAA) and a Dynamic optimal divisible resource allocation methods (DODRA) are used. An error prediction mechanism is implemented here through which optimal node allocation done. But still the system is inefficient due to requirement of time and it benefit to the clients only. In my proposed system a genetic partitioning (GP) method is implemented. In GP, after partitioning the cloud system, nodes are allotted for different partitions and job is chosen based on the partition in which node placed. Then by applying genetic algorithm based on these partitions, every task can be optimally allocated to the hosts. Here by comparing both existing and my proposed systems through CloudSim simulator we can analyze that genetic partitioning is efficient than AP method.

Keywords: ODRA, Local optimal allocation algorithm (LOAA), Partition method, genetic algorithm

I. INTRODUCTION

The system follows a popular server/client model to allocate and process cloud user requests. The cloud server collect dynamically available states of nodes and customizing virtual machines based on users various requirements or demands. A task execution can be split into three steps: Initially select a qualified physical node. Then the virtual machine resources are customized on demand by virtual machine monitor. Finally send computational results to users by the server [1]. Different task executions have different execution patterns because of multiple types of resource requirement. ODRA method used [2]. But it is only useful to users. For providers benefit we require a genetic algorithm with partitioned cloud. It require less time for allocation of best node. Instead of existing system's serial process here system will be more efficient so that it done parallel. It saves time. So both ideal time and response time can be reduced.

A partitioning of cloud system will provide a minimum Load balance and improves efficiency of public cloud environment. It can be used in very large complex systems.Resources can be CPU, memory, bandwidth etc. These have to be allocated to client on demand.

A suitable node is to be allocated for the job. While allocation, it requires efficient analysis of which node to be allocated. It must be profitable one for the clients. They must take less time and cost for the node allocation. Also it must be profitable to the server. Otherwise it is inefficient for the cloud system. Again the system must require less communication delay. For satisfying client node requirements our existing system proposed.

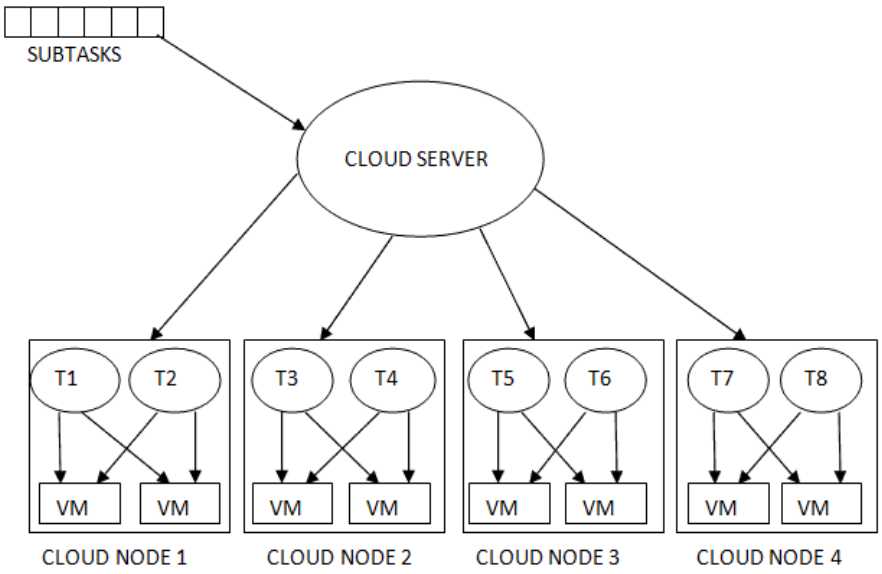

Fig 1: Client server model for task allocation in the cloud system

Fig 1 shows cloud system architecture having four nodes or clients and one cloud server. For processing a job cloud system require best host system and it is to be allotted with the help of server. Depending on various resource requirement these allocation where done. Clients submit their tasks to sever. There can be any number of requests at a time. Server will allocate them to the most suitable nodes. This is the basic model of cloud resource allocation process. Here the aim is to provide most optimal allocation of job to node. In this paper there is only one VM for each node and they will be allocated with corresponding cloudlet. Then task computation is done in that machine and the result where sent to the requested clients. Clients will receive the output. By using our DODRA algorithm, division of cloud and partition genetic algorithm implement an optimal allocation of task.

II . RELATED WORKS

A cloud resource allocation problem commonly deals with precise work load/host load. An optimization model for resource allocation problem assuming that workload information could be known in advance [2]. In our system workload information is not known in advance. We predict that and analyze it is right or wrong. Proposed a rebalancing method for distributed file systems in [4]. Here the data size of file system is easy to predict preciously. It is deal with single dimension. But in proposed system it deals with multiple dimensions. A multi dimensional SLA based resource allocation for multi-tier applications [5]. Here with the assumption of values resource power and user requests. Another system deals with resource allocation system. But it is between two clients or client and provider.

It is between two clients or client and provider. Their solutions have resource capacities are always large enough. Here limited resource capacity, leading to a huge challenge especially in the bound analysis with prediction errors. CloudScale [6] is a cloud system provides online demand prediction and handles prediction errors to achieve better resource allocation. In contract to this work, we theoretically derived the bound of the task execution length with errors. Moreover, we improved the algorithm by making it more efficient through genetic partitioning.

A load balancing model is aimed at the public cloud which has nodes with distributed computing resources in many different geographic locations [9]. This model divides the cloud into several cloud partitions. When the environment is very large and complex, use this model. The

system contains a main controller and the balancer for each cloud partition chooses the best load balancing strategy.

III. PROPOSED WORK

Local optimal allocation method provides an optimal allocation through predicting the resource requirements of a task in future. Based on time and cost requirements future prediction of requirements done. Time and cost found by calculating data size with respect to node speed. So future task's and subtask's time predicted for each node and an optimal node selected. In Optimal divisible resource allocation method focuses on how to make full use of the multi attribute resources facilitated by checking existing node has enough space for node allocation. It is found by reducing execution time from remaining time. So under users' specific resource demands, allocation can be done. By doing so, system can increase efficiency and optimality of assigning task.

Dynamic version for load balancing, algorithm can further extended to a dynamic version for making the AP system a runtime. Due to the dependency between the subtasks of a task, the resource availability state there is a requirement for cloud system to be dynamic one. For updating the resource availability states DODRA method is required. Also this can tune the resource allocation at runtime based on task's execution progress and update resource availability states. Same procedur are done on this AP method too. Only difference is that systems have to be resolved to a dynamic.

Expected Execution Time EET(E(i))=size(Ei) / speed(Ni)

Expected Execution CostEEC(E(i))=EET(E(i))* cost(N(i))

Remaining Time (E(i))= deadtime(Ei) - EET(i)

Here one thing to be checked is the taken task can have sub task. So to find the execution time and cost, have to consider the previous task's execution time also and it is to be added. But if the task is new one then only consider that tasks execution time because it is the parent node.

Algorithm Adaptive Prediction(AP)

Input: Nodes N1, N2…..Nn

Events E1, E2.....Em

Output: Task allocation

1. Get n nodes and m events
2. Get resources for the nodes and events
3. Perform deadline based sorting in tasks
4. Do matchmaking and get events Ei where i<=no:of tasks
5. For each Ei do
6. Perform prediction()
7. Perform selection()
8. Allocate task Ei on Nj
9. End for loop

- *Algorithm prediction ()*

1. For each node N do
2. check if primary node or intermediate node
3. Find EET(E(i)) ==size(Ei)/speed(Ni)

4. EEC(E(i))==(size(Ei)/speed(Ni))*cost(Ni)
5. ET(E(i)) == size(Ei)/speed(Ni)
6. RT(E(i)) == deadtime(Ei)-EET(i)
7. Return prediction parameters
8. End

- *Algorithm Selection()*

1. For a no: of rounds do
2. For each node N do
3. f(i)= W1*EET(i)+W2*EEC(i) // W1 and W2 are weight vectors
4. B[i]=min(f(i),B[i])
5. if(ET<RT)
6. Find min time=min(f(i),min_time)
7. min_cost=min(f(i),min_cost)
8. End for
9. End for
10. Return node

Algorithm DODRA explains with two inputs, nodes and events. Where initially the "n" number of nodes and "m" number of jobs we inputted. Initially we do deadline based sorting in tasks. Sorting is done by taking the largest length task then it is removed to a linked list and then taking second largest one and it is also removed and similarly all the task are taken . Then here do matchmaking where check for every task that it satisfies given nodes. If satisfied then it is taken in for allocation. Then do method of prediction then selection of node and then allocate. In prediction function, select each task with node and find minimum time and cost. Calculate prediction parameters Expected execution time, Expected execution cost, expected time and return time. Then selection of node done based on the ODRA method, that is minimum time and cost taken node is allotted for the specific task, if process have extra return time it is allocated. After this do genetic partition as discussed below.

The load balancing model using partition has numerous nodes that are distributed in many different geographic locations. In our partition genetic model divides the public cloud into several cloud partitions. Here let 4 partitions. When the environment is very large and complex we use divisions for load balancing. The cloud has a main controller that chooses the suitable partitions for arriving jobs. Here system uses a method for finding best partition through prediction. Like the balancer for each cloud partition for refreshing the system. Here update the system after the processes [11]. When a job arrives at the cloud system, the first step is to choose the right partition. The cloud partition state can be divided into three types:

- Idle state: If there is no node processed then it is changed to idle state. So load degree is zero. Its status is idle now. That node can be allocated by tasks.
- Normal state: Node is normal. It can process nodes in future. Here load degree is greater than maximum load degree and less than zero.
- Overload state: In this state it is full. No more nodes can be allocated. Load degree is greater than or equal to maximum load degree.

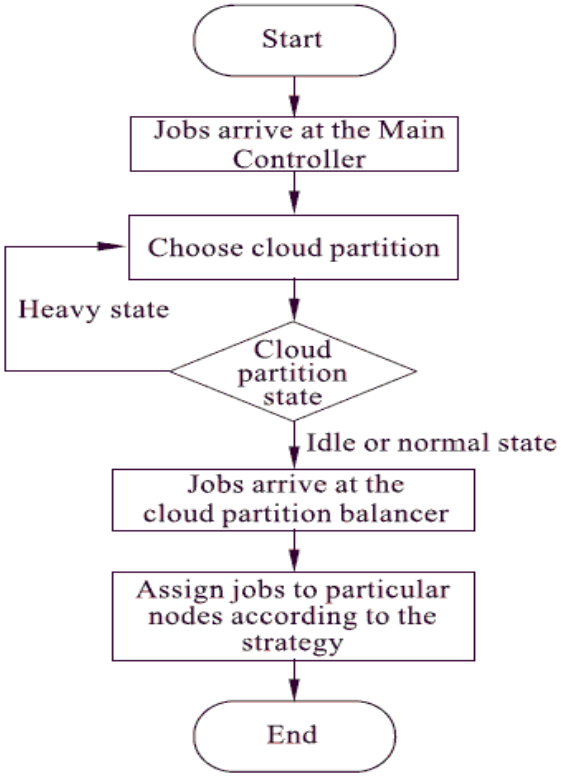

Fig 2: Job assignment strategy

Partitioning Algorithm

1. Start
2. while task do
3. searchBestPartition (task);
4. check if partitionState == idle or partitionState == normal then
5. Send task to Partition;
6. else
7. check for another Partition;
8. end if
9. end while
10. end

Genetic algorithm is a research heuristic that contain the process of evaluation is used as the strategy for assigning node. GA can be applied to process controllers for their optimization using natural operators. Here a population of candidate solutions called individuals is taken for the processes. Selection, Mutation, Crossover, Best is the processes. Selection and the best processes are used depending on the fitness function. Then each candidate solution has a some of properties which can be mutated and altered [7]. By inputting a limit rounds and number of sequences, in the selection process the selected sequences are choose such that one of the two sequences having minimum fitness value or maximum rank will taken as the next sequence. Then 'crossover', in which combination of two sequences is taken. And again for 'mutation', it takes sequence by altering or modifying the sequences. Then 'best' in which depending on the fitness function sequences are selected for next round iteration.

Genetic algorithm problem

Input: R, Round
N, Sequence length
Output: Solution Sequence

Initial

- Randomly generate N solutions A[]
- Return A[ ]

Selection

- //here we generate 2N solutions B[]
- Copy first N sequences from A[ ] to B[ ]
- From i =0 to N randomly select two sequences S1 & S2 from A[ ] and set B[i+N]=Best of(S1,S2) based on fitness function
- Return B[ ]

Crossover

- //here we generate 2N solutions C[]
- Copy first N sequences from B[ ] to C[ ]
- From i =0 to N randomly select two sequences S1 & S2 from B[ ] and set C[i+N]=Combine (S1,S2) based on fitness function
- Return C[ ]

Mutation

- //here we generate 2N solutions D[]
- Copy first N sequences from C[ ] to D[ ]
- From i =0 to N randomly select two sequences S1 & S2 from C[ ] and set D[i+N]=Best of(S1,S2) based on fitness function
- Return D[ ]

Best

- // here we generate N solutions E[ ]
- From i =0 to N randomly select N sequences as best of sequences E() based on fitness function
- Return E[ ]

Let we have 10 rounds and 4 sequences input. After this five stages for round 1 do the same iteratively for the next 9 rounds. The output from best is given to the next round's initial. After these 10 iterations we get sequences with most optimal allocation of jobs to corresponding nodes.

Here we calculate Normal Execution time, Normal communication cost and Normal execution cost. Then to find Rank we add these three parameters for every sequence. Then take maximum of the ranked sequence for the next iteration. Fitness function is used in two stages, in Selection and in best. They form most optimized sequences. It can be used for next iteration or used

as the final result. Until a specific number of sequences the problem will iterate. Here say 10 time iteration required. After this iteration result will be optimal as we required. Finally we can analyse that our system with DODRA and partition using genetic algorithm both efficiency and compare. The ideal time and response time are checked. We got result that DODRA system requires more time as compared to the genetic partition system.

IV. RESULTS

By doing AP and Genetic Partitioning algorithm in cloudSim we can evaluate the performance in Fig 4 and Fig 5. By finding ideal time and response time for both existing and proposed system, we can conclude that proposed system is highly efficient than AP method. Initially we have a number of nodes and number of jobs. They are allocated through AP method and calculated ideal time and response time. Then we have partitioning of cloud in to 4 parts and nodes are assigned through geographical distance calculation.

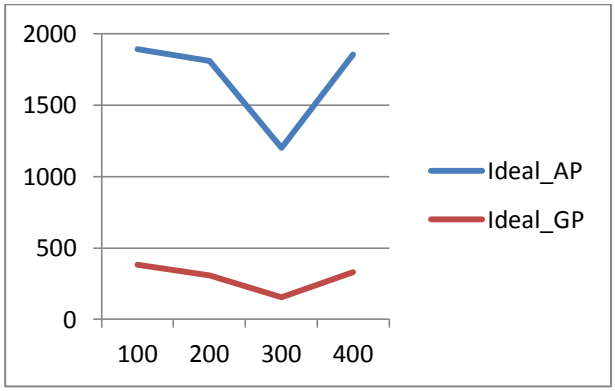

Fig 4. Performance diagram (Ideal Time)

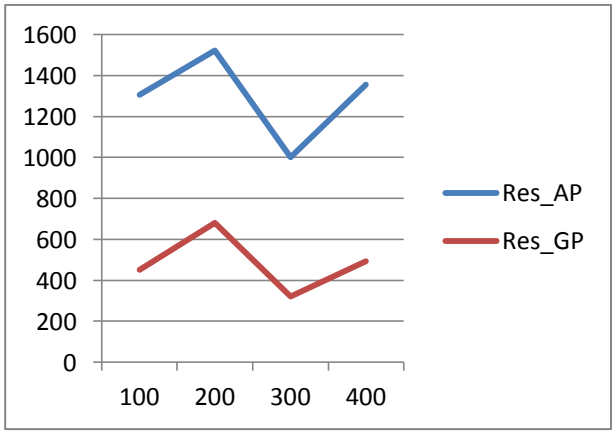

Fig 5. Performance diagram (Response Time)

IV. CONCLUSION

In this paper, deals a method for minimizing task execution length in a distributed cloud system through resource allocation. Main aim of the system is to minimize user's payment and also to make maximum profit to cloud provider. This is done by using the ideal time minimization. By using DODRA algorithm and genetic partitioning, it is proven that the output of this algorithm is optimal based on the CloudSim evaluation. A prediction method implemented in the DODRA and partition genetic provides a best result. Ideal time and response time performance shows a better comparison between existing and proposed systems. A large variation in the time will give a better efficiency to my proposed system.

REFERENCES

[1] F. Chang, J. Ren, and R. Viswanathan, "Optimal Resource Allocation in Clouds," Proc. Third IEEE Int'l Conf. Cloud Computing (Cloud '10), pp. 418-425, 2010.
[2] S. Di and C.-L. Wang, "Dynamic Optimization of Multi-Attribute Resource Allocation in Self-Organizing Clouds," IEEE Trans. Parallel and Distributed Systems, vol. 24.
[3] S. Di and C-L. Wang, "Minimization of Cloud Task Execution Length with Workload Prediction Errors," Proc. 20th High.
[4] H. Hsiao, H. Su, H. Shen, and Y. Chao, "Load Rebalancing for Distributed File Systems in Clouds," IEEE Trans. Parallel and Distributed Systems, vol. 24, no. 5, pp. 951-962.
[5] S. Di and C.-L. Wang, "Error-Tolerant Resource Allocation and Payment Minimization for Cloud System," IEEE Trans. Parallel and Distributed Systems, vol. 24, no. 6, pp.
[6] Ben-tal and A. Nemirovski, "Robust Convex Optimization,"
[7] D.Rajeswari, V.Jawahar, Senthil, R.Kanaga "Efficient Scheduling Using Multi-Objective Genetic Algorithm for Independent Task",2014 March.
[8] S. Chaisiri, B.S. Lee, and D. Niyato, "Optimization of Resource Provisioning Cost in Cloud Computing," IEEE Trans. Services Computing, vol. 5, no. 2, pp. 164-177.
[9] Gaochao Xu, Junjie Pang, and Xiaodong Fu "A Load Balancing Model Based on Cloud Partitioning for the Public Cloud" IEEE,
[10] Google App Engine (google code): http://code.google.com/ appengine/, 2013.

