Doubling Runtime Estimations to Improve Performance of Backfill Algorithms in Cloud Metaschedular Considering Job Dependencies

Ankur Jindal and P. Sateesh Kumar

Indian Institute of Technology, Roorkee-247667, India
ankur2.iitr@gmail.com,
drpskfec@iitr.ernet.in

Abstract. Job scheduling is a very challenging issue in cloud computing. Traditional backfill algorithms such as Easy and conservative are extensively used as job scheduling algorithms. Backfill algorithms require the shorter job to come forward if sufficient resources for the execution of this job are available and run in parallel with the currently running jobs provided it does not delay the next queued jobs. This technique is highly dependent on runtime estimations of job execution. Moreover in real life scenario it has seen that submitted job's may or may not be independent to each other. In this paper we have proposed a technique that uses dynamic grouping method to consider job dependencies and doubling runtime estimation method in cloud metaschedular to improve performance of backfill algorithm. Results have shown that doubling runtime estimations can significantly improve performance of backfill scheduling algorithms provided that the runtime estimations are correct.

1 Introduction

Cloud computing [4] is a future technology that won't need to compute on local computers, but on centralized facilities operated by third-party compute and storage utilities. Job scheduling is one of the core and challenging issues in a Cloud Computing system. In general two schedulers [5] are available in cloud one is global or metaschedular and another is local scheduler. Local scheduler determines how the processes that reside on a single CPU are allocated and executed. Users submit their jobs to Metaschedular, it is metaschedular responsibility to use information about the system and allocate processes among the different clusters in cloud. In this paper, it is attempted to improve the performance of EASY(the Extensible Argonne Scheduling System) backfill Algorithm[2] by doubling runtime estimation method in cloud metaschedular to maximize the resource utilization and minimize the resource gap of idle resources. We also have taken care that submitted jobs may or may not be independent to each other. The evaluation of EASY algorithm before and after doubling runtime estimations is made. The rest of the paper is organized as follows: the next section discusses the related works. Section 3 presents an overview of cloud metaschedular architecture where EASY with doubling runtime estimations as well as dynamic grouping of jobs is made. Section 4 describes an algorithm which is combination of dynamic grouping and EASY algorithm with doubling of runtime estimations. Results of the performance and parameter study are reported in section 5.

Aswatha Kumar M. et al. (Eds.): Proceedings of ICAdC, AISC 174, pp. 621–628.
springerlink.com

2 Related Work

Recent years have seen many efforts focused on the efficient utilization of cloud resources by cloud metaschedular that lead satisfaction to both cloud service provider and service users. CloudSim [8] allows modeling and simulations of entities in parallel and distributed computing systems. Aneka[9] form enterprise grid and cloud platform provide following services as task scheduler service for the task programming model, thread scheduler services, for the thread programming model, storage service for file store for applications. Hadoop a popular open-source implementation of the Google's Map Reduce model is primarily developed by Yahoo. The work done by [6], [7] considers Hadoop scheduler can cause severe performance degradation in heterogeneous environments and provide a new scheduling algorithm, Longest Approximate Time to End (LATE) for concurrent jobs in heterogeneous environments. But LATE doesn't always improve the performance.

The work related to [1] considers self adaptive backfill policy for parallel systems using multi queue. And IBM in paper [3] proves the effectiveness of backfill algorithms for parallel systems. The work done by [10] focuses on optimizing the system throughput by maximizing the overall resource utilization and guaranteeing increased performance of the applications. Here an optimal solution for cloud job scheduling is made only better than the traditional First Come First Serve (FCFS), Round robin and failed to fill the resource gap completely. The work related to [2], consider the commonly used method of job scheduling FCFS, along with Backfilling method EASY and CONSERVATIVE algorithms where small jobs are moved ahead in the schedule can fill the resources gap that is generated by FCFS. However it has seen that resource gap is not fully covered using given runtime estimations.

3 Task Scheduling Problem

In general cloud users submit their jobs to cloud metaschedular. It is the cloud metaschedular which make decision to map jobs submitted by cloud users to cloud clusters. Figure1. Shows the scenario.

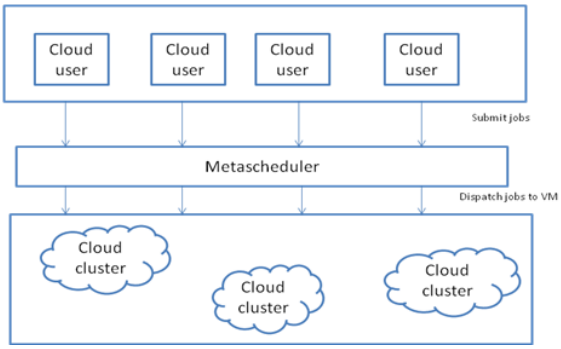

Fig. 1. Cloud metaschedular

The following steps are performed:

Step1. The cloud users submit their request for job completion to the metaschedular
Step2. As per the availability of free nodes in cloud cluster decision for scheduling is made. It is the metaschedular which is responsible for mapping jobs between cloud clusters and cloud users.
Step3. After the jobs are submitted to cloud cluster these are executed by local scheduler.

4 Task Scheduling Algorithm

We are representing our job workflow in the form of a DAG (Directed Acyclic Graph) G (V, E). V (Vertices) represents jobs and E (Directed edges) represents dependencies. We are considering a DAG because if there is a cycle present, we will stick in a situation of deadlock. Following steps are performed to make dynamic grouping.

4.1 Dynamic Grouping Method Algorithm

1. First find all the root nodes means jobs which are not dependent on any other job. Put these jobs in first group.
2. Increment group number and check all the nodes which are directly dependent on all or some jobs of the previous group. Put these jobs in this new group.
3. Check if there are any other nodes left, if yes go to step 2 otherwise step 4.
4. Apply EASY with doubling runtime estimation on each of these groups individually.

Fig. 2. Dynamic Grouping Method

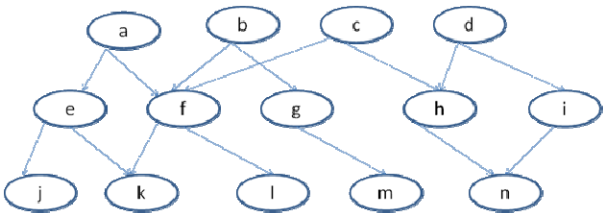

Fig. 3. DAG representing job workflow

Here according to this method we get three groups for DAG shown in Figure2 G1= {a,b,c,d}, G2= {e,f,g,h,i}, G3= {j,k,I,m,n}

Improved Easy Backfill Algorithm:

1. Double the given runtime estimation's of jobs
2. Find the shadow time and extra nodes
    a) Find when enough nodes will be available for the first queued job; this is the shadow time.
    b) If this job does not need all the available nodes, the ones left over are the extra nodes.
3. Find a backfill job
    a) Loop on the list of queued jobs in order of arrival
    b) For each check whether either of these conditions hold
        i. It requires no more than the currently free nodes ,and will terminate by the shadow time , or
        ii. It requires no more than the minimum of the currently free nodes and extra node
    c) The first such job can be used for backfilling

Fig. 4. EASY with doubling runtime estimations

Consider a scenario with 5 jobs in some individual group as shown in Fig 5. According to EASY backfill algorithm with actual runtime estimates the jobs will be executed as shown in Fig 6. The total execution time of all the 5 jobs come here is 950 units. But if we double the runtime estimates of the jobs total job execution time come here is750 units. The corresponding execution sequence is shown in Fig 7-11.These Figures are self explanatory.

Jobs	Number of Pe's required	Expected Runtime
J1	5	400
J2	9	100
J3	2	200
J4	4	300
J5	7	150

Fig. 5. Jobs with expected runtime

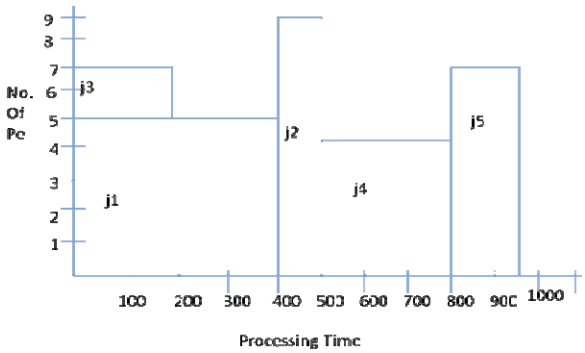

Fig. 6. Job execution according to EASY backfill with actual runtimes

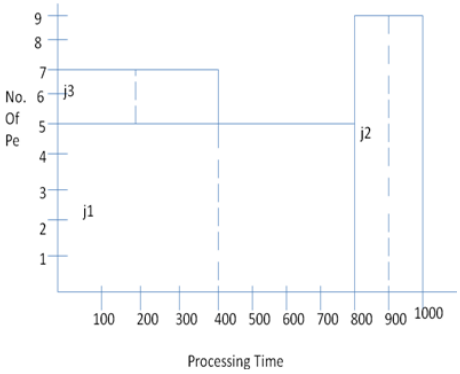

Fig. 7. Job execution according to EASY backfill with doubling j1 and j3 starts execution

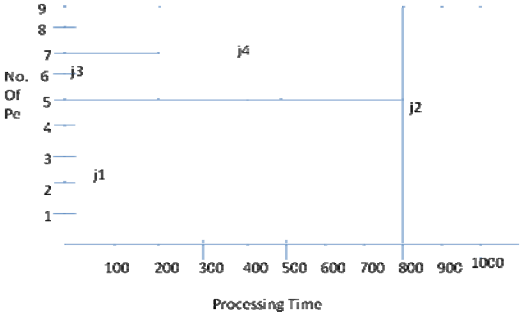

Fig. 8. Job execution according to EASY backfill with doubling after j3 completed execution and j4 and j1 are executing

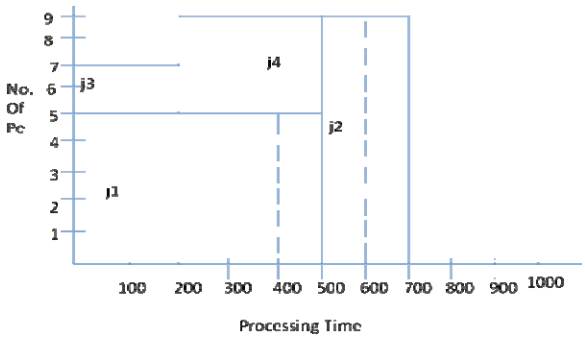

Fig. 9. Job execution according to EASY backfill with doubling after j4 and j1 completed execution and j2 is executing

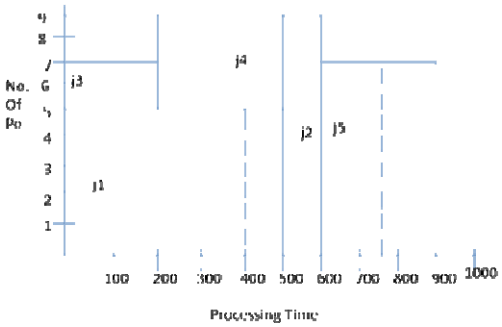

Fig. 10. Job execution according to EASY backfill with doubling after j2 completed execution and j5 is executing

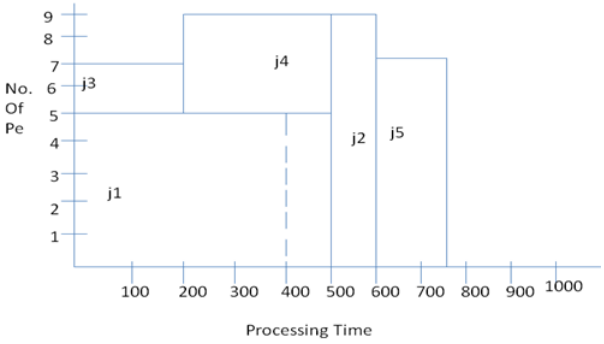

Fig. 11. Job execution according to EASY backfill with doubling after all jobs completed execution

5 Simulation and Results

In this section, the experimental evaluation for the cloud metaschedular is discussed. The Cloudsim toolkit is used to simulate the algorithm with various experimental setups. The default classes in Cloudsim toolkit are extended to implement the proposed policy and other parallel job scheduling strategies. The experimental setup include by varying jobs runtime, speed of processing elements, size of cloudlets and also policies. It can be analyzed by experimental results as shown in Figure 12 that job execution with doubling runtime estimation is faster than backfill algorithms with actual runtime estimations.

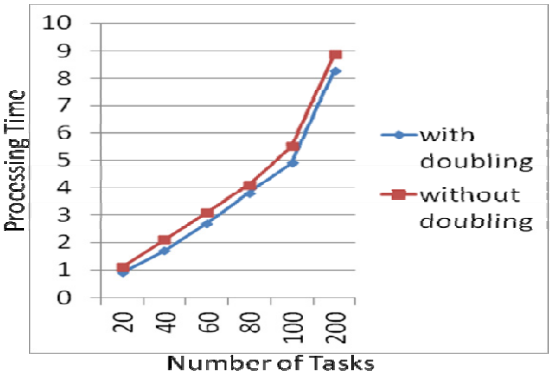

Fig. 12. Performance Backfill Algorithms before and after doubling

6 Conclusion

Doubling approximates an SJF like scheduling by repeatedly preventing the first queued job from being started. Thus, doubling trades off fairness for performance and should be viewed as a property of the scheduler, not the predictor. We have shown doubling technique on EASY Backfill algorithm, but it can be applied to any backfill algorithm.

References

1. Lawson, B.G., Smirni, E., Puiu, D.: Self-adapting backfilling scheduling for parallel systems
2. Feitelson, D.G., Weil, A.M.: Utilization and predictability in scheduling the IBM SP2 with backfilling. In: Proceedings of the First Merged International and Symposium on Parallel and Distributed Processing, Parallel Processing Symposium, IPPS/SPDP 1998, pp. 542–546 (1998)
3. Tsafrir, D., Etsion, Y., Feitelson, D.G.: Backfilling using system-generated predictions rather than user runtime estimates. IEEE Transactions on Parallel and Distributed Systems 18(6), 789–803 (2007)

4. Foster, I., et al.: Cloud Computing and Grid Computing 360-Degree Compared. In: Grid Computing Environments Workshop, pp. 1–10 (2008)
5. Peixoto, M.L.M., et al.: A Metascheduler architecture to provide QoS on the cloud computing. In: 2010 IEEE 17th International Conference on Telecommunications (ICT), pp. 650–657 (2010)
6. Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R., Stoica, I.: Improving map reduce performance in heterogenous environment. In: OSDI (2008)
7. Isard, M., et al.: Quincy: fair scheduling for distributed comptuing clusters. Microsoft Research, SOSP (2008)
8. Buyya, R., et al.: Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. In: International Conference on High Performance Computing & Simulation, HPCS 2009, pp. 1–11 (2009)
9. Buyya, R.: Aneka next generation. net grid/cloud computing company (2009)
10. Sadhasivam, S., Jeya Rani, R., Nagaveni, N., Vasanth Ram, R.: Design and implementation of two level scheduler for cloud computing environment. In: International Conference on Advance in Recent Technologies in Communication and Computing (2009)