Mathematical Statistician and Engineering Applications
ISSN:2094-0343
2326-986

Analysis of Earliest Deadline First Scheduler in Hadoop Mapreduce Environments

Devika Rankhambe [#1], Akash Dodake [*2], Mangal Wagh [#3], Kiran Ghate [#4]
Assistant Professor [#1], Assistant Professor [*2], Assistant Professor [#3], Assistant Professor [#4]
Apcoer Pune [#1], Apcoer Pune [*2], Apcoer Pune [#3], Apcoer Pune [#4]
Devika.rankhambe@abmspcoerpune.org [#1]

Article Info
Page Number: 749-758
Publication Issue:
Vol 71 No. 1 (2021)

Article History
Article Received: *02 February 2022*
Revised: *10 March 2022*
Accepted: *25 March 2022*
Publication: *15 April 2022*

Abstract
The MapReduce applications used for processing petabytes of data across the enterprise. Controlling the allocation of resources in shared MapReduce environments is a key challenge. Many users require job completion time guarantees. There is an increasing number of MapReduce applications associated with live business intelligence that require completion time guarantees (SLOs). There is a lack of performance models and workload analysis tools for automated performance management of such MapReduce jobs. None of the existing Hadoop schedulers support completion time guarantees (SLOs). A key challenge in shared MapReduce clusters is the ability to automatically tailor and control resource allocations to different applications for achieving their performance SLOs. We implemented a novel SLO-based scheduler in Hadoop by making use of ARIA framework that determines job ordering and the amount of resources to allocate for meeting the job deadlines. The new scheduler effectively meets the jobs' SLOs until the job demands exceed the cluster resources. This paper is all about EDF Scheduler Design, EDF Scheduling Algorithm, and Algorithmic Complexity of EDF Scheduling Scheme as well as SWOT analysis of EDF Scheduler.

Keywords: Hadoop; MapReduce; Resource Allocation; SLOs; Scheduling

1.0 Introduction

Early versions of Hadoop had a very simple approach to scheduling users jobs: they ran in order of submission, using a FIFO scheduler. Typically, each job would use the whole cluster, so jobs had to wait their turn. Although a shared cluster offers great potential for offering large resources to many users, the problem of sharing resources fairly between users requires a better scheduler. Production jobs need to complete in a timely manner, while allowing users who are making smaller ad hoc queries to get results back in a reasonable time. Later on, the ability to set a job's priority was added, via the mapred.job.priority property or the setJobPriority() method on JobClient (both of which take one of the values VERY_HIGH, HIGH, NORMAL, LOW, VERY_LOW).

MapReduce applications process PBs of data across enterprise. Key challenge: controlling the allocation of resources in shared MapReduce environments. Many users require job completion time guarantees. There is no support from existing schedulers FIFO, Fair Scheduler, Capacity Scheduler. In order to achieve Service Level Objectives (SLO), we need to answer, When will the job finish given certain resources? and How much resources should be allocated to complete the job within a given deadline?

Currently, there is no job scheduler for MapReduce environments that given a job completion deadline, could estimate and allocate the appropriate number of map and reduce slots to the job so that it meets the required deadline. In this work, we design a framework, called ARIA.

We implement a novel SLO-scheduler in Hadoop that determines job ordering and the amount of resources that need to be allocated for meeting the job's SLOs. The job ordering is based on the EDF policy (Earliest Deadline First). For resource allocation, the new scheduler relies on the designed performance model to suggest the appropriate number of map and reduce slots for meeting the job deadlines. The resource allocations are dynamically recomputed during the job's execution and adjusted if necessary.

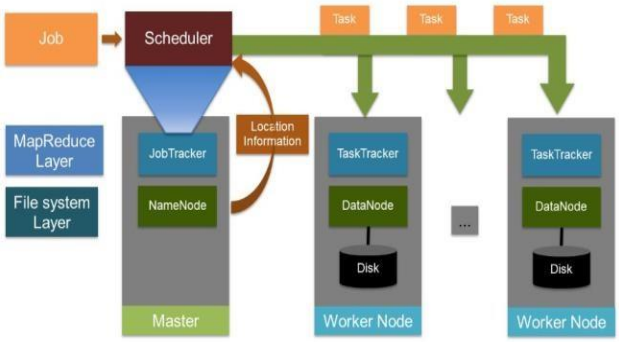


Fig. 1 Job Scheduling in Hadoop

Literature Review

In [5], M. Zaharia, D. Borthakur, J. Sen Sarma proposed Hadoop Fair Scheduler in order to maintain fairness between different users to allocate equal shares to each of the users running the MapReduce jobs and to maximize data locality

In the paper[3], To address the conflict between locality and fairness, authors propose a simple algorithm called delay scheduling.

In paper[4],Matei Zaharia proposed LATE scheduler which is highly robust to heterogeneity and can improve Hadoop response times. It uses estimated finish times to speculatively execute the tasks that hurt the response time the most. LATE performs significantly better than Hadoop's default speculative execution algorithm in real workloads on Amazon's Elastic Compute Cloud.

In Paper[5], T.Sandholm and K. Lai proposed Dynamic proportional share scheduler in Hadoop which allows users to offer for map and reduce slots by adjusting their spending over time and approach enables dynamically controlled resource allocation.It allows users to control their allocated capacity by adjusting their spending over time. Allows scheduler to make more efficient decisions about which jobs and users to prioritize and gives users the tool to optimize and customize their allocations to fit the importance and requirements of their jobs.

In papers[5,6],M. Isard, M. Budiu suggested Quincy scheduler to achieve fairness and data locality goals for Dryad environment. The authors design a novel technique that maps the fair scheduling problem to the classic problem of min-cost flow in a directed graph to generate a schedule. A powerful and flexible new framework for scheduling concurrent distributed jobs with fine-grain resource sharing. The scheduling problem is mapped to a graph data structure, where edge weights and capacities encode the competing demands of data locality, fairness, and starvation-freedom, and a standard solver computes the optimal online schedule according to a global cost model.

In Paper[7], J. Wolf extended HFS by proposing a special slot allocation schema that aims to optimize explicitly some given scheduling metrics (response time, stretch, makespan and Service Level Agreements (SLAs), among others) while ensuring the same minimum job slot guarantees as in HFS, and maximum job slot guarantees as well. FLEX relies on the speedup function of the job (for map and reduce stages) that produces the job execution time as a function of the allocated slots.

In Paper[8], K. Morton, M. Balazinska used ParaTimer for estimating the progress of parallel queries expressed as Pig scripts that can translate into directed acyclic graphs (DAGs) of MapReduce jobs. ParaTimer targets environments where declarative queries are translated into ensembles of MapReduce jobs.

Proposed Work

We have discussed different scheduling techniques in Hadoop available upto date.

i) Scheduler Design

Our goal is to propose a novel Deadline based scheduler for Mapreduce environments that supports a new API in which a job can be submitted with a desirable job completion deadline. The scheduler will then estimate and allocate the appropriate number of map and reduce slots to the job so that it meets the required deadline.To achieve this goal, we designed and implemented a framework, called ARIA, to address this problem.

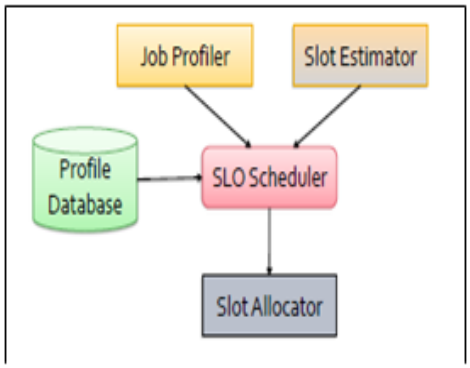


Fig. 1 Scheduler Design

ii)Job Profiler

It collects the job profile information for the currently running or finished jobs. We use the Hadoop counters, which are sentfrom the workers to the master along with each heartbeat to build the profile. This profile information can also be gleaned from the logs in the HDFS output directory or on the job master after the job is completed. The job profile is then stored persistently in the profile database. We create a compact job profile that is consists of performance invariants that reflects all the phases of a given job viz, : map, shuffle, sort, and reduce phases, which are independent of the amount of resources assigned to the job over time. This information is obtained from the counters at the job master during the job's execution or parsed from the logs. The map stage consists of a number of map tasks. To compactly characterize the task duration distribution and other invariant properties, we extract the following metrics: (Mmin , Mavg , Mmax , AvgSizeinputM , SelectivityM)
where,

- Mmin – the minimum map task duration. Mmin serves as an estimate for the beginning of the shuffle phase since it starts when the first map task completes.
- Mavg – the average duration of map tasks to summa-rize the duration of a map wave.
- Mmax – the maximum duration of map tasks. It is used as a worst time estimate for a map wave completion.
- AvgSizeinputM - the average amount of input data per map task. We use it to estimate the number of map tasks to be spawned for processing a new dataset.
- SelectivityM – the ratio of the map output size to the map input size. It is used to estimate the amount of intermediate data produced by the map stage.
- The job profile in the shuffle phase is characterized by two pairs of measurements: (Sh1avg , Sh1max , Shtypavg , Shtypmax).
- The reduce phase begins only after the shuffle phase is complete. The profile of the reduce phase is represented by:(Ravg , Rmax , SelectivityR) : the average and maximum of the reduce tasks durations and the reduce selectivity, denoted as SelectivityR , which is defined as the ratio of the reduce output size to its input.

iii) Profile Database:

We use a MySQL database to store the past profiles of the jobs. The profiles are identified by the user and job name which can be specified by the application.

iv) Slot Estimator:

Given the past profile of the job and the deadline, the slot estimator calculates the minimum number of map and reduce slots that need to be allocated to the job in order to meet its SLO. It uses the Lagranges Multiplier method to find the minima on the allocation curve.

$$T_J^{low} = A_J^{low} \cdot \frac{N_M^J}{S_M^J} + B_J^{low} \cdot \frac{N_R^J}{S_R^J} + C_J^{low}$$

where,

AlowJ = Mavg, BlowJ = (Shtypavg + Ravg), and ClowJ =Sh1avg − Shtypavg.

The equation provides an explicit expression of a job completion time as a function of map and reduce slots allocated to a job J for processing its map and reduce tasks, i.e., as a function of (NJM,NJR) and (SJM, SJR).The equations for TupJ and TavgJ can be written similarly.

v) Slot Allocator: Using the slots calculated from the slot estimator, the slot allocator assigns tasks to jobs such that the job is always below the allocated thresholds by keeping track of the number of running map and reduce tasks. In case there are spare slots, they can be allocated based on the additional policy. There could be different classes of jobs, jobs with and without deadlines. The jobs with deadlines will have higher priorities for cluster resources than jobs without deadlines. Once jobs with deadlines are allocated their required minimums for meeting the SLOs, the remaining slots can be distributed to the other job classes.

vi) SLO-Scheduler: This is the central component that co-ordinates events between all the other components. Hadoop provides a support for a pluggable scheduler. The scheduler makes global decisions of ordering the jobs and allocating the slots across the jobs. The scheduler listens for events like job submissions, worker heartbeats, etc. When a heartbeat containing the number of free slots is received from the workers, the scheduler returns a list of tasks to be assigned to it.

The SLO-scheduler has to answer two inter-related questions, which job should the slots be allocated and how many slots should be allocated to the job? The scheduler executes the Earliest Deadline First algorithm (EDF) for job ordering to maximize the utility function of all the users. The second question is answered using the Lagrange computation. The detailed slot allocation schema is shown in the following Algorithm.

.**vii) EDF Scheduling Algorithm**

Step 1: Start

Step 2: Create Job Profile database by considering past running history of each job. Step 3: Submit a job application 'J' with SLO- deadline.

Step 4: Collect profile of a job 'J' from the database.

Step 5: Estimate amount of map and reduce slots (m_j,r_j)using Lagranges multiplier method. Step 6: After receiving heartbeat from node 'n' sort jobs in order of deadline present in the database. Step 7: If free map/reduce slot 's' is present on node 'n'. If Yes then goto step 8 else goto step 15. Step 8: Verify if 〖Runningmaps〗_j < m_j and s is a map slot

Step 9: If job J has unlaunched map task t with data on node n.

Step 10: If YES then launch a map task 't' with local data on node 'n'. Step 11: If NO then launch map task 't'with any other node 'n'.

Step 12: If 〖Finishedmaps〗_j > 0 and 's' is a reduce slot and 〖Runningreduces〗_j<rj

Step 13: If YES launch task 't' on node 'n'.

Step 14: Goto Step 8.

Step 15. If each task T_j is finished by node 'n' present. If YES then goto step 17 else goto step 20. Step 16: Recompute slots (m_j,r_j) based on the current time, current progress and deadline of job J.

Step 17: Store job execution history during map, reduce, shuffle, sort durations for each tasks in profile database. Step 18: Update profile database after job gets completed.

Step 19. Stop.

Description of EDF Scheduling Algorithm

The EDF Scheduling Algorithm consists of two parts : 1) when a job is added, and 2) when a heartbeat is received from a worker. Whenever a job is added, we fetch its profile from the database and compute the minimum number of map and reduce slots required to complete the job within its specified deadline using the Lagrange's multiplier method.

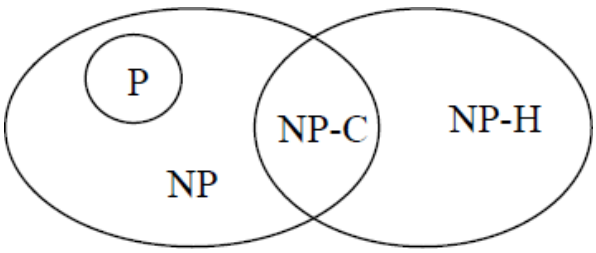
Workers periodically send a heartbeat to the master reporting their health, the progress of their running tasks and the number of free map and reduce slots. In response, the master returns a list of tasks to be assigned to the worker. The master tracks the number of running and finished map and reduce tasks for each job. For each free slot and each job, if the number of running maps is lesser than the number of map slots we want to assign it, a new task is launched. The preference is given to tasks that have data local to the worker node. Finally, if at least one map has finished, reduce tasks are launched as required. In some cases, the amount of slots available for allocation is less than required minima for job J and then J is allocated only a fraction of required resources. As time progresses, the resource allocations are recomputed during the job's execution and adjusted if necessary. Whenever a worker reports a completed task, we decrement NJM or NJR in the SLO- based model and re-compute the minimum number of slots.

04. Algorithmic Analysis of EDF Scheduler

i) Algorithmic Complexity of EDF Scheduler

The data in the selected datasets is handled using parallel approach. The complete dataset is divided amongst mapper. In this way parallel computation using MapReduce takes place. Thus, the complexity of is O (n) and for Sorting of 'n' Jobs from the queue as per the deadline will take O(nlogn) complexity. In worst case, sorting will take O(n^2) complexity.

NP Completeness of EDF Scheduling



Where,
NP-C : NP Complete
NP-H : NP Hard

Only a decision problem can be NPC. Optimization problems may be NPH. Optimization problems may reduce to corresponding decision problem. Optimization problems cannot be NP Complete where decision problem can. If algorithm has n products which go on increasing, then it will be NPH. But it is restricted to fixed n, it will convert to NPC.

This problem will come under NPC as well as NPH. Its NPC because its related to predicting approximate makespan bound time and coming to certain decision which is a probabilistic good algorithm and can be solved in non-polynomial time. It becomes NPH if worker node fails and master doesn't receive heartbeat then master can go in infinite loop to search for node which has empty map & reduce slots to assign task. Also, if a number of jobs in a queue increase then due to contention and due to increase in waiting time, it can go to NP-Complete state.

SWOT Analysis of EDF Scheduler

i) Strengths

- This scheduler is useful in controlling the allocation of resources in a shared MapReduce environments which is a key challenge.
- Many users require job completion time guarantees, Existing schedulers do not support SLOs, this project will be useful inachieving SLOs.
- It performs jobs ordering in Earliest Deadline First(EDF) fashion.
- This scheduler computes required resource allocation for a job from its historic profile and a given deadline T
- The scheduler is automatic.
- It preserves data locality.
- This scheduler is robust against runtime variability
- It profiles the job while it is running.
- It does dynamic adjustments of the allocation of resources.
- The new scheduler effectively meets the jobs' SLOs until the job demands exceed the cluster resources.
- Comparative study of different Hadoop Schedulers is done.

ii) Weaknesses

- If we build profile by executing on smaller datasets, then duration of map tasks not impacted.
- If we build profile by executing on Larger datasets, then greater number of map tasks
- Each map task processes fixed amount of data
- If number of reduce tasks kept constant, intermediate data processed per reduce task increases for longer durations. Reducestage = shuffle + reduce phase
- Shuffle duration depends on network performance
- Reduce duration depends on reduce function and disk write speed
- Proper division of the computation is required for optimization of different set of input dataset sizes.
- Different amount of resources can lead to different executions viz,Job execution can be very different depending on the amountof allocated resources.

iii) Opportunities

- Designed to meet Job's SLOs and to give Service Completion time guarantees.
- It is useful for users who require service guarantees.
- Many enterprises, financial institutions and government organizations are experiencing a paradigm shift towards large scale data intensive computing. Analyzing large amount (petabytes) of unstructured data is a high priority task for many companies. So, for large scale data processing Hadoop-Mapreduce is used most popularly.
- It is increasingly being used across the enterprise for advanced data analytics and enabling new applications associated with data retention, regulatory compliance, e-discovery and litigation issues.
- In MapReduce environments, many production jobs are run periodically on new data. For example, Facebook, Yahoo!, and eBay process terabytes of data and event logs per day on their Hadoop clusters for spam detection, business intelligence and different types of optimizations.
- For the production jobs that are routinely executed on the new datasets, we can build on-line job profiles that later are used for resource allocation and performance management by the job scheduler.

  Benchmarking Hadoop, optimizing cluster parameter settings, design job schedulers with different performance objectives and constructing intelligent workload management in shared Hadoop clusters create an exciting list of challenges and opportunities for the performance analysis and modeling in MapReduce environments.
- Much faster in performing computations using MapReduce.
- Helps in realizing the full potential of the MapReduce.

iii) Threats

- If commodity hardware fails then more failures to system may happen.
- It's performance depends on Time of failure and whether resources replenishable or not.
- In Worker failure, there may be faulty hard disk or process may crash.
- Time of failure also matters in recovery process.
- If failure happens in map stage, recompute all completed or in-progress map tasks of the failed node.
- If failure happens in Reduce stage, recompute all in-progress reduce tasks of the failed node and the shuffle phase of thesereduce tasks too.
- If number of jobs increase in the job queue then waiting time may increase.
- The accuracy of new performance models might depend on the resource contention, especially, the network contention in theproduction Hadoop cluster.
- Typically, service providers tend to over provision network resources to avoid undesirable side effects of network contention.

Conclusion

In this paper, we have used a novel performance modeling framework ARIA (Automatic Resource Inference and Allocation) to achieve performance goals and Deadlines, need for job completion time guarantees , for applications that use a MapReduce cluster for resource sharing in enterprise setting. The paper has introduced bounds-based performance model which is quite accurate. Proposed MapReduce job profiling is compact and comprised of

performance invariants, required by the SLO-scheduler. Robust prediction of required resources for achieving given SLOs. EDF scheduling is a main policy for real-time processing. The implemented EDF scheduling algorithm improves the job completion time, response time in Hadoop environment. In this paper, we performed Algorithmic Complexity Analysis of EDF Scheduler. We have found Strengths, Weakness, Opportunities & Threats, in the EDF Scheduler.

Acknowledgement

I am profoundly grateful to Innovation Club members, Prof. Ashok Ranade & Prof. Ashok Saraf for their expert guidance and continuous encouragement throughout to see that this paper rights its target since its commencement to the completion & Preparation of Paper.

References

[1] J.Dean and S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, In Proc. Communications of the ACM, vol. 51, no. 1, 2008.

[2] Apache Hadoop. http://hadoop.apache.org.

[3] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling In Proc..EuroSys '10: Proceedings of the 5th European conferenceon Computer systems, pages 265–278, New York, NY, USA, 2010. ACM.

[4] Hadoop Distributed File System http://hadoop.apache.org/hdfs

[5] Hadoop's Fair Scheduler http://hadoop.apache.org/common/docs/r0.20.2/fair_scheduler.html

[6] Hadoop's Capacity Scheduler http://hadoop.apache.org/docs/r0.20.2/capacity_scheduler.html

[7] M.Zaharia, A.Konwinski, A.Joseph, Y.zatz, and I.Stoica. Improving MapReduce Performance in Heterogeneous Environments InProc. OSDI'08: 8th USENIX Symposium on Operating Systems Design and Implementation, October 2008

[8] B.Thirumala Rao, L.S.S.Reddy. Scheduling Data Intensive Workloads through Virtualization on MapReduce based Clouds. In Proc. Department of Computer Science and Engineering, Lakireddy Bali Reddy College of Engineering, Mylavaram

[9] Aysan Rasooli , Douglas G. Down. An adaptive Scheduling Algorithm for Dynamic heterogeneous Hadoop Systems In Proc. Department of Computing and Software McMaster University.

[10] ChenHe, Ying Lu, David Swanson Matchmaking: A New MapReduce Scheduling Technique. In Proc Department of Computer Science and Engineering,University of Nebraska-Lincoln Lincoln, U.S.

[11] Brian Cho and Philbert Lin.Responsive Priority Scheduling in Large-scale Hadoop Clusters. In Proc. University of Illinois at Urbana-Champaign.

[12] Linh T.X. Phan, Zhuoyao Zhang, Boon Thau Loo, Insup Lee. Real-time MapReduce Scheduling, in Proc. University of Pennsylvania

[13] Sherje, N. P., Agrawal, S. A., Umbarkar, A. M., Kharche, P. P., & Dhabliya, D. (2021). Machinability study and optimization of CNC drilling process parameters for HSLA steel with coated and uncoated drill bit. Materials Today: Proceedings,

doi:10.1016/j.matpr.2020.12.1070

[14] Shukla, A., Almal, S., Gupta, A., Jain, R., Mishra, R., & Dhabliya, D. (2022). DL based system for on-board image classification in real time, applied to disaster mitigation. Paper presented at the PDGC 2022 - 2022 7th International Conference on Parallel, Distributed and Grid Computing, 663-668. doi:10.1109/PDGC56933.2022.10053139 Retrieved from www.scopus.com

[15] Kamal Kc, Kemafor Anyanwu.Scheduling Hadoop Jobs to Meet Deadlines. In Proc.Department of Computer Science North Carolina State University.

[16] Linh T.X. Phan Zhuoyao Zhang Qi Zheng Boon Thau Loo Insup Lee.An Empirical Analysis of Scheduling Techniques for Real-time Cloud-based Data Processing.In Proc. University of Pennsylvania

[17] homas Sandholm and Kevin Lai.Dynamic Proportional Share Scheduling in Hadoop In Proc JSSPP'10: 15th Workshop on JobScheduling Strategies for Parallel Processing, April,2010