Performance based Prescheduling for MapReduce Heterogeneous Environments with minimal Speculation

Muhammad Adnan Ayub[1], Bilal Ahmad[1], Dr. Zahoor-ur-Rehman[*1], Muhammad Idrees[2], Zaheer Ahmed[1]

[1]Department of Computer Science, COMSATS Institute of Information Technology, Attock, Pakistan
[2]Department of Computer Science, University of Engineering and Technology, Narowal Campus

*Contacting author

Abstract: Currently for big data processing the de facto standard is MapReduce framework because of its attractive facilities and features. The most important feature is its automatic parallelizing a work or job into several tasks and evidently handles task execution on a large cluster of commodity machines. The growing heterogeneity in distributed environment may end up in a few straggling tasks, which delay job completion. To overcome the problem, speculative execution is offered to mitigate stragglers. Though, the speculative execution mechanism is a solution to the problem but existing system are inefficient as speculation process itself is an overhead to the MapReduce mechanism. This paper purposes a performance based pre-scheduling methodology for splitted tasks in MapReduce framework with minimal speculation. The approach is shortly termed as PMS (Pre-scheduling for Minimal Speculation). The controller system monitors and update the performance of the sub nodes utilizing their heartbeat feature. This information is further used to pre-schedule the upcoming tasks in an optimal and dynamic way. This would improve MapReduce performance, and further reduce the speculative execution cost.

Keywords: Minimal Speculative execution, Performance based scheduling, MapReduce, Heterogeneous Environment, Hadoop

1. Introduction

The most popular computer applications in the contemporary era are the ones providing services over the internet to millions of customers. In order to operate efficiently, these applications required handling of massive data. This modern trend has given rise to parallel processing. Since it is proposed by Google (1), the MapReduce framework has attained extensive adoption. It is then popularized by Apache and Yahoo. Similar to MapReduce system, for instance Dryad (2) and Hadoop (3) have been used in the past on a massive commodity apparatuses to work with applications involving intense data processing. MapReduce has the capacity to divide a job into several parallel tasks automatically and seamlessly handle this parallel job execution in a distributed but parallel setting, and it is one of the most attractive features of MapReduce. Slowest task determined the job completion time. The sluggish job termed as stragglers delay jobs and overall reduce performance due to heterogeneity of distributed settings. The necessities of scale-out systems and the development of virtualization have additionally worsened the heterogeneity. Therefore, handling the stragglers becomes precarious and of major interest. Researchers have developed a few straggler virtualization approaches. These techniques are normally clustered into two classes called speculative execution and blacklisting. Former is constructed in shape of the mapping segment or the reduction segment. The speculative tasks are scheduled by master for straggling tasks and are put inside the queue, when speculative execution is enabled. When execution slots are available, these speculative tasks will be launched. The scheduler also ensures that only single speculative task would be running for each original task. Either the speculative task finishes first or the original task finishes, in both cases, the original task is killed. Though, numerous corporations disable speculative execution on few jobs or tasks (4) because it lowers efficiency of the system. On the other hand, Blacklisting utilizes a user-offered

health check script to distinguish the eminence of the sub systems. It can be blacklisted if a sub-node is not performing properly, then these sub-nodes are not scheduled or assigned with any job. However, a severe or improper health check package will be adversely effected on distributed systems by reducing the numbers of available resources. In addition, stragglers can ascend on the non-blacklisted nodes at times as well, often in reason to some composite causes such as IO conflicts, hardware performances, and background services (5). The experimentation (6) on local groups was carried out and the efficiency of speculative execution was tested. The results show that, with stragglers, 42% of speculative task were destroyed with Hadoop's native speculative execution while 68% in case without stragglers. One improved algorithm of speculative execution in heterogeneous environments, over 70 % speculative tasks are lastly killed with LATE algorithm (4). The reason is that killed tasks utilize additional resources without taking part in output. Here in our proposed methodology sub systems' performance is collected, and basing on these performances task scheduling is carried out in term of completion time span, which will minimize the speculation. Furthermore, we monitor the performance of sub nodes in form of feedback from Sub nodes to handle requirement of speculation. The remaining work in the paper is structured as follows. Section 2 argues on related work of few researchers on Speculative execution, Performance based scheduling, MapReduce, Heterogeneous Environment, Hadoop. Sections 3 offer the methodology of proposed minimal speculative execution. Section 4 provides evaluation and comprehensive discussion on the performance of proposed methodology. The paper is concluded in Section 5.

2. Materials

Hadoop's usage of MapReduce nearly looks like Google's File System (7). The single main system dealing with various sub-nodes. Further, the file used as input, which lives on a disseminated file system all through the group, is part into equal sized lumps repeated for adaptation to non-critical failure. Similarly, Hadoop isolates each MapReduce work into an arrangement of errands. Every lump of information is initially handled by a map errand, which yields a set produced by a client characterized map capacity. Then the map yields are part into cans considering key. When all maps have completed, diminish errands apply a decrease capacity to the rundown of map outcomes with every value of key. In Figure 1 pictorial representation of MapReduce calculation is given.

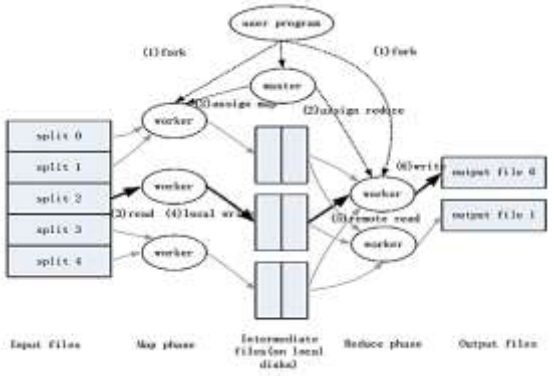

Figure 1 MapReduce Computation

As far as Hadoop is concern, it runs a few maps and decreases simultaneously on every sub, two of each as a matter of default – to cover calculation and I/O. Every sub node acknowledges the main when it has unfilled task openings. It is then the scheduler allocates it errands. Discussing the Mitigating Stragglers, two researchers Dean and Ghemawat initially depicted the straggler issue (1) and projected a broad instrument that plans reinforcement executions of the staying in-advancement errands when work is near speculation as theory. Speculative execution is, the point at which a hub has a vacant undertaking opening, at that time Hadoop picks a job for it from one of three classifications. Initially, any fizzled errands are given most noteworthy need. It is completed to distinguish when an undertaking falls flat over and over because of a bug and stop the employment. The second point is, non-running assignments are considered. Tasks with information neighborhood to the hub are picked first for maps. At last, Hadoop searches for an assignment to execute hypothetically.

To choose theoretical assignments, Hadoop screens undertaking progress utilizing an advancement score somewhere around 0 and 1. The advancement score is the part of information read for the map. The execution is isolated into three stages, each of which records for 1/3 of the score for a decrease task:

- When the assignment brings map output, the duplicate stage.
- The sort stage, when guide yields are sorted by key.
- The decrease stage, when a client characterized capacity is connected to the rundown of guide yields with every key.

In every stage, the score is the part of information handled. For instance, an undertaking part of the way through the duplicate stage has an advancement score of 1/2 x 1/3 = 1/6, while an errand part of the way through the decrease stage scores 1/3 + 1/3 + (1/2 x 1/3) = 5/6. Hadoop takes a gander at the normal advancement score of every class of errands (maps and diminishes) to characterize an edge for theoretical execution: When an assignment's advancement score is not exactly the normal for its classification short 0.2, and the undertaking has keep running for no less than one moment, it is set apart as a straggler. All assignments past the limit are considered "similarly moderate," and ties between them are broken by information area. The scheduler additionally guarantees that at most one theoretical duplicate of every assignment is running at once. In spite of the fact that a metric like advancement rate would gain more sense than total ground for recognizing stragglers, the limit in Hadoop works sensibly well in homogenous situations since assignments tend to begin and complete in "waves" at generally the same times and theory just begins when the last wave is running. At last, when running various employments, Hadoop utilizes a FIFO order where the most punctual submitted occupation is requested an errand to run, then the second, and so forth. There is likewise a need framework for placing occupations into higher-need lines.

LATE scheduler to alleviate stragglers in a heterogeneous setting is outlined by the Zaharia et al.(4). Moreover, the scheduler dependably hypothetically runs the errands that would complete most remote into what's to come. It depends upon the approximation of an errand's suspicions and completion time that undertakings utilizing the lengthiest rough time to end give the best chance to a theoretical duplicate to overwhelm the first assignment.

Chen et al. proposed a self-adaptable MapReduce (SAMR) planning calculation to change the time weight of every phase of guide and lessen errands, so as to gauge the remaining time of assignments all the more precisely, (8). Moreover, Sun et al. broadened the SAMR calculation by taking into the account other three aspects such as the work sort, the dataset size, and the hub setups (9).

Similarly, a research by Ananthanarayanan et al. (10) introduced the framework that initially monitors the tasks and kills stragglers taking into account the reasons and assets. Later in (5) gave a procedure to alleviate stragglers in little occupations by dispatching various clones of each errand when a vocation begins. These systems can be separated into boycotting and theoretical execution. Different methodologies handle straggling undertakings by parceling and adjusting the assignments' workloads. Case in point, Ramakrishnan et al. (11) presented a dynamic testing method and a static burden adjusting framework to relieve reducer information skew

In the research work of Gufler et al. they proposed a few methods of observing and catching information conveyance at mapper side to moderate skewed reducers (12, 13). Similarly other researchers Kwon et al. considered a few procedures of burden uneven characters (14) and outlined their study in (6) to deal stragglers, for example, SkewTune(15) and SkewReduce(16). Reciprocal to these works, proposed research manage insignificant theory by minimizing stragglers which is like the circumstance mindful mappers (17), which use helpful guide assignments to settle on streamlining choice.

Moreover, in this segment, we depict the component utilized by Hadoop to disseminate work over a group. We distinguish suppositions made by the scheduler that hurt its execution.

3. Proposed Methodology

To overcome the limitations discussed in the literature review we proposed an idea. In our idea, we audit the local theoretical execution utilized by Hadoop. The following section discussed the proposed idea in detail.

A) Speculative Execution in Hadoop

At the point when a client presents work, the JobClient conjures the execution of InputFormat to separate the information into numerous coherent InputSplits. Each InputSplit depicts where the info information originates from and which errand it has a place with. The mentioned InputSplits are composed to HDFS and utilized afterward by propelled errands. The JobTracker conveys errands to the TaskTrackers and screens the lifecycle of occupation execution after that.

For example, taking a map task as an illustration, the phases of task execution are depicted as takes after:

(1) The Read phase: a planned assignment gets the comparing InputSplit from HDFS, peruses the information determined by the InputSplit, and parses the inputs into key-esteem sets, (k1, v1), as guide inputs.

(2) The Compute phase: in this stage, a client characterized map capacity is conjured to devour every record (a key value combine) and creates new key-esteem sets, (k2, v2). The recently created middle of the road information is reserved in a roundabout memory cushion.

(3) The Sort and spill phase: when the use of the memory cradle achieves an edge, the middle information is non-concurrently sorted by segments alongside moderate keys and spilled to neighborhood plate. It must be noted that, the spilling string tries its best not to obstruct the guide stage. If a combiner is indicated, a consolidate capacity is to run before spilling.

(4) The Merge phase: The cushioned transitional information is additionally sorted and consolidated alternatively after all the information is handled. At that point, whole spilled records and memory information are converged into a last yield document and a file record.

At the point when a Task Tracker has a void guide space, Hadoop's Task Scheduler doles out it an undertaking which will keep running in an isolated Child JVM. In the event that the errand is a theoretical and executes the aforementioned phases with the same asset as the first assignment. Amid the running of the speculative task, the first knows nothing about its rival. First errand is slaughtered by the Job Tracker if the theoretical undertaking completes initially and the other way around.

For a decrease assignment, the running is isolated into four stages such as copy, sort, diminish, and compose. Unique in relation to a guide errand, a diminish assignment peruses information from remote mappers rather than document framework. In the lessen stage, a client characterized decrease capacity is conjured as far as an uncommon instance of a record, a key gathering. For whatever length of time that theoretical execution is empowered, both the first diminish assignment and the theoretical one compose interim results to the record framework. Just the person who finishes first can effectively present the outcomes.

The Job Tracker screens the advancement of every assignment utilizing an advancement score somewhere around 0 and 1. Normal advancement score of every classification of assignments is utilized for speculative execution as the limit: it is considered as a straggler if an errand's advancement score is not exactly the normal less 0.2. LATE (4) gauges the opportunity to fulfillment and conjectures the undertakings that have the longest inexact time to end as an enhanced calculation. The mentioned two schedulers additionally guarantee that at most one theoretical duplicate of every errand is running at once.

B) Minimal Speculative Execution in PMS

In our proposed methodology, pre-scheduling is done in the starting phase before committing the jobs to the Sub-System. That pre-scheduling is based on the collected performance data of the Sub-Systems. PMS system algorithm uses greedy approach for assigning the tasks to the Sub-System, System assign the task to the Sub-System by consideration of its performance and minimum time to complete the task. With this approach, we foresee and utilized the system with maximal performance without runtime speculation.

Proposed approach also incorporated with signals (heartbeat messages) from the Sub-System after defined interval of time to check the performance of sub-systems. That updated performance data can be further used update the assigning of upcoming task to avoid runtime speculation overhead

4. PMS Evaluation and Discussion

To justify the results of our PMS methodology, we compare the LATE, SAMR scheduling algorithm with PMS for MapReduce. For comparison, an example of speculative execution is considered with three Sub-Systems in a heterogeneous environment. Details discussion of the speculative execution on the given instance for LATE, SAMR, and PMS is as follows: -

A) LATE

In LATE scheduling mechanism, the idea of speculation is to back up the straggler task with largest estimated finish time also known as longest approximate time to end. Before speculation selection we calculate the progress rate (i) and on that progress rate estimate time left (ii) from the formulas as follow: -

$$Progrss\ Rate = \frac{Progress\ Score}{Execution\ Time} \ldots (i)$$

$$Estimated\ Time\ Left = \frac{1 - Progress\ Score}{Progress\ Rate} \ldots (ii)$$

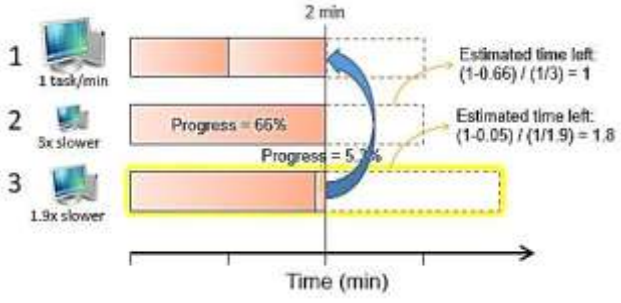

Figure 2 LATE Scheduling Mechanism

Here in our example of LATE scheduler, when the fastest sub-system finishes assigned tasks, the progress of sub-system with remaining task will checked and calculate estimated time to end and only select the backup task with largest time to complete as depicted in Fig-2 LATE scheduler is efficient in terms of minimal loss of completed task but comparative to our proposed mechanism PMS which is with minimal speculation assignment of task is based on performance of the sub-systems.

B) SAMR

Self-Adaptive MapReduce scheduler work the same way as LATE scheduler work, but the difference is its dynamically scheduling process, which changes the assignment in the runtime on considering the execution performance of the sub-system. SAMR is also have the dedicated system for task tracking which periodically checking the task execution information as a feedback from the working sub-systems show in Fig-3. Comparing to our purposed methodology PMS is considering the performance of Sub-System which is gather on startup and assignment of task is based on that performance information, whereas proposed mechanism is similar in a way of gathering the feed from the Sub-Systems.

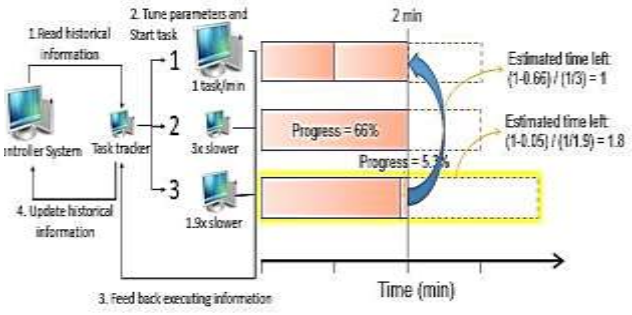

Figure 3 SAMR Scheduling Mechanism

C) PMS

In our proposed mechanism of pre-scheduling process is defined on the first stage after taking the performance of sub-systems into account.

In proposed approach pre-scheduling is based on the collected performance data of the Sub-Systems. PMS system algorithm uses greedy approach for assigning the tasks to the Sub-System, System assign the task to the Sub-System by consideration of its performance and minimum time to complete the task. By the said

approach we foresee and utilized the system with maximal performance without runtime speculation. So, efficiency of PMS is the elimination of Speculation overhead cost and sub-system those are not assigned with jobs will be further utilized for other jobs.

The proposed approach also incorporated with signals (heartbeat messages) from the Sub-System after defined interval of time to check the performance of sub-systems shown in Fig-4. That updated performance data can be further used update the assigning of upcoming task to avoid runtime speculation overhead. PMS has an advantage over SAMR as it is a Task Tracker free system, and feedback is collected only when the performance of Sub-System decreases or increases from the pre-stored performance of Sub-System.

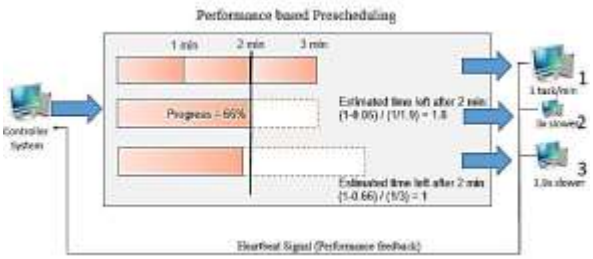

Figure 4 PMS Scheduling Mechanism

5. Conclusion

This research offers an approach of pre-scheduling with minimal speculative execution time in MapReduce, the real-world problem of node heterogeneity motivated us to propose this idea. The research recognized shortcoming with both the LATE approach and SAMR scheduler. A simple, robust pre-scheduling algorithm, PMS is designed, which uses probable completion times to speculatively execute the tasks that delay the response time up to the maximum extent. But it differs from LATE as it pre-schedule the task for minimal speculation and enhances the overall performance of the system. This research claims that, PMS performs significantly better than LATE and SAMR in term of performance and time.

References

1. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Communications of the *ACM*. 2008;51(1):107-13.
2. Isard M, Budiu M, Yu Y, Birrell A, Fetterly D, editors. Dryad: distributed data-parallel programs from sequential building blocks. ACM SIGOPS operating systems review; 2007: *ACM*.
3. Chandio AA, Tziritas N, Xu C-Z. Big-data processing techniques and their challenges in transport domain. *ZTE Communications*. 2015;1(010).
4. Zaharia M, Konwinski A, Joseph AD, Katz RH, Stoica I, editors. Improving MapReduce performance in heterogeneous environments. Osdi; 2008.
5. Ananthanarayanan G, Ghodsi A, Shenker S, Stoica I, editors. Effective Straggler Mitigation: Attack of the Clones. *NSDI*; 2013.
6. Wang Y, Lu W, Lou R, Wei B. Improving MapReduce performance with partial speculative execution. *Journal of Grid Computing*. 2015;13(4):587-604.
7. Saritha S. Google file system. 2010.
8. Chen Q, Zhang D, Guo M, Deng Q, Guo S, editors. Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. Computer and Information Technology (CIT), 2010 *IEEE 10th International Conference* on; 2010: IEEE.
9. Sun X, He C, Lu Y, editors. ESAMR: An enhanced self-adaptive mapreduce scheduling algorithm. Parallel and Distributed Systems (ICPADS), 2012 *IEEE 18th International Conference* on; 2012: IEEE.
10. Ananthanarayanan G, Kandula S, Greenberg AG, Stoica I, Lu Y, Saha B, et al., editors. Reining in the Outliers in Map-Reduce Clusters using Mantri. *OSDI*; 2010.
11. Ramakrishnan SR, Swart G, Urmanov A, editors. Balancing reducer skew in MapReduce workloads using progressive sampling. *Proceedings of the Third ACM Symposium on Cloud Computing*; 2012: ACM.
12. Gufler B, Augsten N, Reiser A, Kemper A. Handling Data Skew in MapReduce. Closer. 2011;11:574-83.

13. Gufler B, Augsten N, Reiser A, Kemper A, editors. Load balancing in mapreduce based on scalable cardinality estimates. Data Engineering (ICDE), 2012 *IEEE 28th International Conference* on; 2012: IEEE.
14. Kwon Y, Balazinska M, Howe B, Rolia J. A study of skew in mapreduce applications. *Open Cirrus Summit*. 2011;11.
15. Kwon Y, Balazinska M, Howe B, Rolia J, editors. Skewtune: mitigating skew in mapreduce applications. Proceedings of the 2012 *ACM SIGMOD International Conference on Management of Data;* 2012: ACM.
16. Kwon Y, Balazinska M, Howe B, Rolia J, editors. Skew-resistant parallel processing of feature-extracting scientific user-defined functions. *Proceedings of the 1st ACM symposium on Cloud computing*; 2010: ACM.
17. Vernica R, Balmin A, Beyer KS, Ercegovac V, editors. Adaptive MapReduce using situation-aware mappers. *Proceedings of the 15th International Conference on Extending Database Technology*; 2012: ACM.