IDL - International Digital Library Of
Technology & Research
Volume 1, Issue 2, Mar 2017

Available at: www.dbpublications.org

International e-Journal For Technology And Research-2017

Map Reduce Workloads: A Dynamic Job Ordering and Slot Configurations

Ms. Mamatha M R[1], Dr. K Thippeswamy [2]
Dept. of Computer Science
[1] MTech, Student– VTU PG Center, Mysuru, India
[2] Guide & Head of Department– VTU PG Center, Mysuru, India

SURVEY SURVEY

1 ABSTRACT: MapReduce is a popular parallel computing paradigm for large-scale data processing in clusters and data centers. A MapReduce workload generally contains a set of jobs, each of which consists of multiple map tasks followed by multiple reduce tasks. Due to 1) that map tasks can only run in map slots and reduce tasks can only run in reduce slots, and 2) the general execution constraints that map tasks are executed before reduce tasks, different job execution orders and map/reduce slot configurations for a MapReduce workload have significantly different performance and system utilization. This survey proposes two classes of algorithms to minimize the make span and the total completion time for an offline MapReduce workload. Our first class of algorithms focuses on the job ordering optimization for a MapReduce workload under a given map/reduce slot configuration. In contrast, our second class of algorithms considers the scenario that we can perform optimization for map/reduce slot configuration for a MapReduce workload. We perform simulations as well as experiments on Amazon EC2 and show that our proposed algorithms produce results that are up to 15 - 80 percent better than currently unoptimized Hadoop, leading to significant reductions in running time in practice.

2 INTRODUCTION: In this survey introduce a MapReduce job, it consists of a set of map and reduce tasks, where reduce tasks are performed after the map tasks. Hadoop, an open source implementation of MapReduce, has been deployed in large clusters containing thousands of machines by companies such as Amazon and Facebook. In those cluster and data center environments, MapReduce and Hadoop are used to support batch processing for jobs submitted from multiple users (i.e., MapReduce workloads). Despite many research efforts devoted to improve the performance of a single MapReduce job, there is relatively little attention paid to the system performance of MapReduce workloads. Therefore, this survey tries to improve the performance of MapReduce workloads.

In this survey, we target at one subset of production MapReduce workloads that consist of a set of independent jobs (e.g., each of jobs processes distinct data sets with no dependency between each other) with different approaches. For dependent jobs (i.e., MapReduce workflow), one MapReduce can only start only when its previous dependent jobs finish the computation subject to the input-output data dependency. In contrast, for independent jobs, there is an overlap computation between two jobs, i.e., when the current job completes its map-phase computation and starts its reduce-phase computation, the next job can begin to perform its map-phase computation in a pipeline processing mode by possessing the released map slots from its previous job. This survey we plan to implement

1. We plan to propose a bi-criteria heuristic algorithm to optimize make span and total completion time simultaneously.

2. Propose slot configuration algorithms for make span and total completion time.

We also show that there is a proportional feature for them, which is very important and can be used to address the time efficiency problem of proposed enumeration algorithms for a large size of total slots.

International e-Journal For Technology And Research-2017

3 SURVEY

3.1 Dynamic Job Ordering and Slot Configurations for MapReduce Workloads "MAPREDUCE is a widely used computing model for large scale data processing in cloud computing. A MapReduce job consists of a set of map and reduce tasks, where reduce tasks are performed after the map tasks. Hadoop, an open source implementation of MapReduce, has been deployed in large clusters containing thousands of machines by companies such as Amazon and Facebook. In those cluster and data center environments, MapReduce and Hadoop are used to support batch processing for jobs submitted from multiple users (i.e., MapReduce workloads). Despite many research efforts devoted to improve the performance of a single MapReduce job there is relatively little attention paid to the system performance of MapReduce workloads. Therefore, this survey tries to improve the performance of MapReduce workloads. Makespan and total completion time (TCT) are two key performance metrics. Generally, makespan is defined as the time period since the start of the first job until the completion of the last job for a set of jobs. It considers the computation time of jobs and is often used to measure the performance and utilization efficiency of a system. In contrast, total completion time is referred to as the sum of completed time periods for all jobs since the start of the first job. It is a generalized makespan with queuing time (i.e., waiting time) included. We can use it to measure the satisfaction to the system from a single job's perspective through dividingthe total completion time by the number of jobs (i.e., average completion time). Therefore, in this survey, we aim to optimize these two metrics.

We consider the production MapReduce workloads whose jobs run periodically for processing new data. The default FIFO scheduler is often adopted in order to minimize the overall execution time. The analysis is generally performed offline to optimize the execution for such production workloads. There are a surge amount of optimization approaches on that. For example, Rasmussen et al. and Jiang et al. consider the low-level I/O efficient optimization. Agrawal et al. and Nykiel et al. share the operation by eliminating redundant data access and computation.We evaluate our algorithms using both testbed workloads and synthetic Facebook workloads.

Experiments show that,
1). for the make span, the job ordering optimization algorithm achieve an approximately 14-36 percent improvement for testbed workloads, and 10-20 percent improvement for Facebook workloads. In contrast, with the map/reduce slot configuration algorithm, there are about 50-60 percent improvement for test bed workloads and 54-80 percent make span improvement for Facebook workloads;
2). for the total completion time, there are nearly 5 improvement with the bi-criteria job ordering algorithms and 4 improvement with the bi-criteria map/reduce slot configuration algorithm, for Facebook workloads that contain many small-size jobs."

3.2 A Workload Model for MapReduce "MapReduce is a programming model for parallel computing developed by Google. The need to perform analyses on large amounts of data is not specific to Cloud service providers, but the MapReduce programming model was developed with this specific audience in mind, apart from Google it is known to be used by for example Yahoo!, MySpace, Facebook, and Twitter. Facebook uses MapReduce for, among other, business intelligence, spam detection, and advertisement optimization. The name MapReduce originates from the higher-order1 map and reduces functions originally, found in functional programming languages. A MapReduce program is in fact the combination of a map and a reduce function, the map function is applied on the input data and the reduce function is applied on the output of the map function.

1. A MapReduce job consists of a map function, a reduce function, and input data (on a distributed file system).

2. First, the input data are partitioned into smaller chunks of data.

3. Then, for each chunk of input data, a "map task" runs which applies the map function to the chunk of input data. The resulting output of each map task is a collection of key-value pairs.

4. The output of all map tasks is shuffled, that is, for each distinct key in the map output; a collection is created containing all corresponding values from the map output.

5. Then, for each key-collection resulting from the shuffle phase, a "reduce task" runs which applies the reduce function to the collection of values. The resulting output is a single key-value pair.

6. The collection of all key-value pairs resulting from the reduce step is the output of the MapReduce job. (In Hadoop the reduce outputs are merged after the the job has finished, when the user uses the "getmerge" command to get the output from the distributed file system.)

The main place where parallelization is exploited in MapReduce is during the running of the map and reduce tasks, depicted as respectively steps three and five in the above description. Although in the above overview the steps two to six seem to be distinct phases, the more advanced MapReduce implementations run these phases in parallel, a reduce task can for example start working as soon as the first key-value pair is emitted by a map task."

3.3 Energy Efficiency for Large-Scale MapReduce Workloads with Significant Interactive Analysis "
Massive computing clusters are increasingly being used for data analysis. The sheer scale and cost of these clusters make it critical to improve their operating efficiency, including energy. Energy costs are a large fraction of the total cost of ownership of datacenters. Consequently, there is a concerted effort to improve energy efficiency for Internet datacenters, encompassing government reports, standardization efforts, and research projects in both industry and academia. Approaches to increasing datacenter energy efficiency depend on the workload in question. One option is to increase machine utilization, i.e., increase the amount of work done per unit energy. This approach is favored by large web search companies such as Google, whose machines have persistently low utilization and waste considerable energy. Clusters implementing this approach would service a mix of interactive and batch workloads, with the interactive services handling the external customer queries, and batch processing building the data structures that support the interactive services. This strategy relies on predictable diurnal patterns in web query workloads, using latency-insensitive batch processing drawn from an "infinite queue of low-priority work" to smooth out diurnal variations, to keep machines at high utilization. This survey focuses on an alternate use case—what we call MapReduce with Interactive Analysis (MIA) workloads. MIA workloads contain interactive services, traditional batch processing, and large-scale, latency-sensitive processing. The last component arises from human data analysts interactively exploring large data sets via ad-hoc queries, and subsequently issuing large-scale processing requests once they find a good way to extract value from the data. Such human-initiated requests have flexible but not indefinite execution deadlines.
MIA workloads require a very different approach to energy-efficiency, one that focuses on decreasing the amount of energy used to service the workload. As we will show by analyzing traces of a front-line MIA cluster at Facebook, such workloads have arrival patterns beyond the system's control. This makes MIA workloads unpredictable: new data sets, new types of processing, and new hardware are added rapidly over time, as analysts collect new data and discover new ways to analyze existing data. Thus, increasing utilization is insufficient: First, the workload is dominated by human-initiated jobs. Hence, the cluster must be provisioned for peak load to maintain good SLOs, and low-priority batch jobs only partially smooth out the workload variation. Second, the workload has unpredictable high spikes compared with regular diurnal patterns for web queries, resulting in wasted work from batch jobs being preempted upon sudden spikes in the workload."

3.4 A Big Data Model with Map Reduce Concept in Storage System "In recent years there has been an extraordinary growth of large-scale data processing and related technologies in both, industry and academic communities. This trend is mostly driven by the need to explore the increasingly large amounts of information that global companies and communities are able to gather, and has lead the introduction of new tools and models, most of which are designed around the idea of handling huge amounts of data. Alongside the need to manage ever larger amounts of information, other developments such as cloud computing have also contributed significantly to the growth of large-scale technologies. Cloud computing Has dramatically transformed the way many critical services are delivered to customers, posing new challenges to data centers. As a result, there is a complete new generation of large-scale infrastructures, bringing an unprecedented level of workload and server consolidation, that demand not only new

programming models, but also new management techniques and hardware platforms. These infrastructures provide excellent opportunities to build data processing solutions that require vast computing resources. However, despite the availability of these infrastructures and new models that deal with massive amounts of data, there is still room for improvement,
Specially with regards to the integration of the two sides of these environments: the systems running on these infrastructures, and the applications executed on top of these systems. A good example of this trend towards improved large-scale data processing is MapReduce, a programming model intended to ease
the development of massively parallel applications, and which has been widely adopted to process large datasets thanks to its simplicity. While the MapReduce model was originally used primarily for batch data processing in large static clusters, nowadays it is mostly deployed along with other kinds of workloads in shared environments in which multiple users may be submitting concurrent jobs with completely different priorities and needs: from small, almost interactive, executions, to very long applications that take hours to complete. Scheduling and selecting tasks for execution is extremely relevant in MapReduce environments since it governs a job's opportunity to make progress and determines its performance. However, only basic primitives to prioritize between jobs are available at the moment, constantly causing either under or over-provisioning, as the amount of resources needed to complete a particular job are not obvious a priori."

3.5 Interactive Analytical Processing in Big Data Systems: A Cross-Industry Study of MapReduce Workloads **"**Many organizations depend on MapReduce to handle their large-scale data processing needs. As companies across diverse industries adopt MapReduce alongside parallel databases, new MapReduce workloads have emerged that feature many small, short, and increasingly interactive jobs.
These workloads depart from the original MapReduce use case targeting purely batch computations, and shares semantic similarities with large-scale interactive query processing, an area of expertise of the RDBMS community. Consequently, recent studies on query-like programming extensions for MapReduce and applying query optimization techniques to MapReduce are likely to bring considerable benefit. However, integrating these ideas into business-critical systems requires configuration tuning and performance benchmarking against reallife production MapReduce workloads. Knowledge of such workloads is currently limited to a handful of technology companies. A cross-workload comparison is thus far absent, and use cases beyond the technology industry have not been described. The increasing diversity of MapReduce operators create a pressing need to characterize industrial MapReduce workloads across multiple companies and industries. Arguably, each commercial company is rightly advocating for their particular use cases, or the particular problems that their products address. Therefore, it falls to neutral researchers in academia to facilitate cross-company collaboration, and mediate the release of cross-industries data. In this survey, we present an empirical analysis of seven industrial MapReduce workload traces over long-durations. They come from production clusters at Facebook, an early adopter of the Hadoop implementation of MapReduce, and at ecommerce, telecommunications, media, and retail customers of Cloudera, a leading enterprise Hadoop vendor. Cumulatively, these traces comprise over a year's worth of data, covering over two million jobs that moved approximately 1.6 exabytes spread over 5000 machines. Combined, the traces offer an opportunity to survey emerging Hadoop use cases across several industries (Cloudera customers), and track the growth over time of a leading Hadoop deployment (Facebook). We believe this survey is the first study that looks at MapReduce use cases beyond
the technology industry, and the first comparison of multiple large-scale industrial MapReduce workloads.Our methodology extends, and breaks down each MapReduce workload into three conceptual components: data, temporal, and compute patterns. The key findings of our analysis are as follows:
• There is a new class of MapReduce workloads for interactive, semi-streaming analysis that differs considerably from the original MapReduce use case targeting purely batch computations.
• There is a wide range of behavior within this workload class, such that we must exercise caution in regarding any aspect of workload dynamics as "typical".
• Query-like programatic frameworks on top of MapReduce such as Hive and Pig make up a considerable fraction of activity in all workloads we analyzed.

• Some prior assumptions about MapReduce such as uniform data access, regular diurnal patterns, and prevalenceof large jobs no longer hold.

Subsets of these observations have emerged in several studies that each looks at only one MapReduce workload. Identifying these characteristics across a rich and diverse set of workloads shows that the observations are applicable to a range of use cases."

3.6 A Storage-Centric Analysis of MapReduce Workloads: File Popularity, Temporal Locality and Arrival Patterns "Due to an explosive growth of data in the scientific and Internet services communities and a strong desire for storing and processing the data, next generation storage systems are being designed to handle peta and exascale storage requirements. As Big Data storage systems continue to grow, a better understanding of the workloads present in these systems becomes critical for proper design and tuning. We analyze one type of Big Data storage cluster: clusters dedicated to supporting a mix of MapReduce jobs. Specifically, we study the file access patterns of two multipeta byte Hadoop clusters at Yahoo! across several dimensions, with a focus on popularity, temporal locality and arrival patterns. We analyze two 6-month traces, which together contain more than 940 million creates and 12 billion file open events. We identify unique properties of the workloads and make the following key observations:

• Workloads are dominated by high file churn (high rate of creates/deletes) which leads to 80% − 90% of files being accessed at most 10 times during a 6-month period.

• There is a small percentage of highly popular files: less than 3% of the files account for 34% − 39% of the accesses (opens).

• Young files account for a high percentage of accesses, but a small percentage of bytes stored. For example, 79% − 85% of accesses target files that are most one day old, yet add up to 1.87% − 2.21% of the bytes stored.

• The observed request interarrivals (opens, creates and deletes) are bursty and exhibit self-similar behavior.

• The files are very short-lived: 90% of the file deletions target files that are 22.27 mins − 1.25 hours old.

Derived from these key observations and knowledge of the domain and application-level workloads running on the clusters, we highlight the following insights and implications to storage system design and tuning:

• The peculiarities observed are mostly derived from short-lived files and high file churn.

• File churn is a result of typical MapReduce workflows: a high-level job is decomposed into multiple MapReduce jobs, arranged in a directed acyclic graph (DAG). Each of these jobs writes its final output the storage system, but the output that interests the user is the output of the last job in the graph. The output of the jobs is deleted soon after it is consumed.

• High rate of change in file population prompts research on appropriate storage media and tiered storage approaches.

• Caching young files or placing them on a fast storage tier could lead to performance improvement at a low cost.

• "Inactive storage" (due to data retention policies and dead projects) constitutes a significant percentage of stored bytes and files; timely recovery of files and appropriate choice of replication mechanisms and media for passive data can lead to improved storage utilization.

• Our findings call for a model of file popularity that accounts for a very dynamic population. To the best of our knowledge, this is the first study of how MapReduce workloads interact with the storage layer."

4 CONCLUSION

This survey focuses on the job ordering and map/reduce slot configuration issues for MapReduce production workloads that run periodically in a data warehouse, where the average execution time of map/reduce tasks for a MapReduce job can be profiled from the history run, under the FIFO scheduling in a Hadoop cluster. Two performance metrics are considered, i.e., makespan and total completion time. We first focus on the makespan. We propose job ordering optimization algorithm and map/reduce slot configuration optimization algorithm. We observe that the total completion time can be poor subject to getting the optimal make span, therefore, we further propose a new greedy job ordering algorithm and a map/reduce slot configuration algorithm to minimize the makespan and total completion time together. The theoretical analysis is also given for our proposed heuristic algorithms, including approximation ratio, upper and lower bounds on

makespan. Finally, survey conduct extensive experiments to validate the effectiveness of our proposed algorithms and their theoretical results.

OTHER REFERENCES

[1] S. R. Hejazi and S. Saghafian, "*Flowshop scheduling problems with makespan criterion: A review*," Int. J. Production Res., vol. 43, no. 14, pp. 2895–2929, 2005.

[2] S. Agarwal, S. Kandula, N. Bruno, M.-C. Wu, I. Stoica, and J. Zhou, "*Re-optimizing data-parallel computing*," in Proc. 9th USE- NIX Conf. Netw. Syst. Design Implementation, 2012, p. 21.

[3] P. Agrawal, D. Kifer, and C. Olston, "*Scheduling shared scans oflarge data files*," Proc. VLDB Endow., vol. 1, no. 1, pp. 958–969, Aug. 2008.

[4] W. Cirne and F. Berman, "*When the herd is smart: Aggregate behavior in the selection of job request*," IEEE Trans. Parallel Disturb. Syst., vol. 14, no. 2, pp. 181–192, Feb. 2003.

[5] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "*Mapreduce online*," in Proc. 7th USENIX Conf. Netw. Syst. Design Implementation, 2010, p. 21.

[6] J. Dean and S. Ghemawat, "*Mapreduce: Simplified data processing on large clusters*," in Proc. 6th Conf. Symp. Oper. Syst. Design Implementation, 2004, vol. 6, p. 10.

[7] J. Dittrich, J.-A.-Quiane Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad, "*adoop++: Making a yellow elephant run like a cheetah (without it even noticing),"* Proc. VLDB Endowment, vol. 3, nos. 1–2, pp. 515–529, Sep. 2010.

[8] P.-F. Dutot, L. Eyraud, G. Mounie, and D. Trystram, "*Bi-criteria algorithm for scheduling jobs on cluster platforms,*" in Proc. 16th Annu. ACM Symp. Parallelism Algorithms Archit., 2004, pp. 125–132.

[9] P.-F. Dutot, G.Mounie, and D. Trystram,"*Scheduling parallel tasks: Approximation algorithms,*" in Handbo ok of Scheduling: Algorithms, Models, and Performance Analysis, J. T. Leung, Ed. Boca Raton, FL, USA: CRC Press, ch. 26, pp. 26-1–26-24.

[10] A. Floratou, J. M. Patel, E. J. Shekita, and S. Tata, "*Columnoriented storage techniques for mapreduce*," Proc. VLDB Endowment, vol. 4, no. 7, pp. 419–429, Apr. 2011.

[11] J. Gupta, A. Hariri, and C. Potts, "*Scheduling a two-stage hybrid flow shop with parallel machines at the first stage,*" Ann. Oper. Res., vol. 69, pp. 171–191, 1997.

[12] J. N. D. Gupta, "*Two-stage, hybrid flowshop scheduling problem*," J. Oper. Res. Soc., vol. 39, no. 4, pp. 359–364, 1988.

[13] H. Herodotou and S. Babu, "*Profiling, what-if analysis, and costbased optimization of mapreduce programs*," Proc. VLDB Endowment, vol. 4, no. 11, pp. 1111–1122, 2011.

[14] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. B. Cetin, and S. Babu, "*Starfish: A self-tuning system for big data analytics*," in Proc. 5th Conf. Innovative Data Syst. Res., 2011, pp. 261–272.

[15] S. Ibrahim, H. Jin, L. Lu, B. He, and S. Wu, "*Adaptive disk I/O scheduling for mapreduce in virtualized environment*," in Proc. Int. Conf. Parallel Process., Sep. 2011, pp. 335–344.

[16] D. Jiang, B. C. Ooi, L. Shi, and S. Wu, "*The performance of mapreduce: An in-depth study*," Proc. VLDB Endowment, vol. 3, nos. 1–2, pp. 472–483, Sep. 2010.

[17] S. M. Johnson, "*Optimal two- and three-stage production schedules with setup times included*," Naval Res. Logistics Quart., vol. 1, no. 1, pp. 61–68, 1954.