DYNAMIC JOB ORDERING AND SLOT CONFIGURATIONS FOR MAPREDUCE WORKLOADS

M VINOD KUMAR[1], K PRAVEEN KUMAR[2], K SOWMYA[3], P V KOMALI[4]
[1, 2, 3, 4]*Assistant Professor, Department of Computer Science and Engineering, AnuBose Institute of TechnologyFor Women's,, KSP Road, New Paloncha, Bhadradri Kothagudem District, Telangana (TS), 507115*

Submitted: 18-04-2024 Accepted: 04-06-2024 Published: 11-06-2024

ABSTRACT: ***Map Reduce is a popular parallel computing paradigm for large-scale data processing in clusters and data centers. A Map Reduce 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 paper proposes two classes of algorithms to minimize the makespan 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 Map Reduce 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 un optimized Hadoop, leading to significant reductions in running time in practice.***

1. INTRODUCTION

MapReduceis a widely used computing model forlarge 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 singleMapReduce job, there is relatively little attention paid to the system performance of MapReduce workloads. Therefore, this paper 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 dividing the total completion time by the number of jobs (i.e., average completion time).

Therefore, in this paper, 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.

Objectives of the study

1) MapReduce is a popular parallel computing paradigm for large-scale data processing in clusters and data centers.
2) A MapReduce workload generally contains a set of jobs, each of which consists of multiple map tasks followed by multiple reduce tasks.
3) This study proposes two classes of algorithms to minimize the makespan and the total completion time for an offline MapReduce workload.
4) 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.
5) 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 unoptimizedHadoop, leading to significant reductions in running time in practice.

2. LITERATURE REVIEW

PIXIDIA: Optimizing Data Parallel Jobs in Wide-Area Data Analytics

In this paper, we present PIXIDA, a scheduler that aims to minimize data movement across resource constrained links. To achieve this, we introduce a new abstraction called SILO, which is key to modeling PIXIDA's scheduling goals as a graph partitioning problem.

A Dynamic Map Reduce Scheduler for Heterogeneous Workloads

In this paper, we give a new view of the Map Reduce model, and classify the Map Reduce workloads into three categories based on their CPU and I/O utilization. With workload classification, we design a new dynamic Map Reduce workload predict mechanism, MR-Predict, which detects the workload type on the fly. We propose a Triple-Queue Scheduler based on the MR-Predict mechanism.

3. ANALYSIS

Existing system

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 ofmachines 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 paper tries to improve the performance of MapReduce workloads.

Proposed system

In this paper, 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.

External Interface Requirements

User Interface

The user interface of this system is a user friendly Java Graphical User Interface.

Hardware Interfaces

The interaction between the user and the console is achieved through Java capabilities.

Software Interfaces

The required software is JAVA1.6.

Operating Environment

Windows XP, Linux.

HARDWARE REQUIREMENTS:

- Processor - Pentium –IV
- Speed - 1.1 Ghz
- RAM - 256 MB(min)
- Hard Disk - 20 GB
- Key Board - Standard Windows Keyboard
- Mouse - Two or Three Button Mouse
- Monitor - SVGA

SOFTWARE REQUIREMENTS:

- Operating System : Windows XP
- Programming Language : Java
- Software : jdk and Cygwin

4. SOFTWARE USED

Java

Initially the language was called as “oak” but it was renamed as “java” in 1995.The primary motivation of this language was the need for a platform-independent (i.e. architecture neutral) language that could be used to create software to be embedded in various consumer electronic devices.

- Java is a programmer’s language
- Java is cohesive and consistent
- Except for those constraint imposed by the Internet environment. Java gives the programmer, full control

Finally Java is to Internet Programming where c was to System Programming.

Importance of Java to the Internet

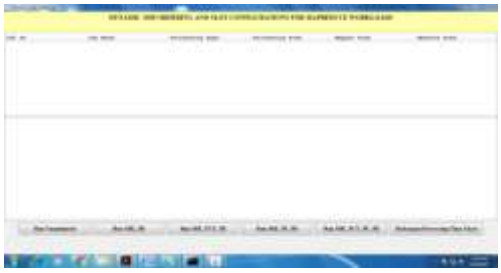
Java has had a profound effect on the Internet. This is because; java expands the Universe of objects that can move about freely in Cyberspace. In a network, two categories of objects are transmitted between the server and the personal computer. They are passive information and Dynamic active programs in the areas of Security and probability. But Java addresses these concerns and by doing so, has opened the door to an exciting new form of program called the Applet.

Applications and applets

An application is a program that runs on our Computer under the operating system of that computer. It is more or less like one creating using C or C++ .Java's ability to create Applets makes it important. An Applet I san application, designed to be transmitted over the Internet and executed by a Java-compatible web browser. An applet I actually a tiny Java program, dynamically downloaded across the network, just like an image. But the difference is, it is an intelligent program, not just a media file. It can be react to the user input and dynamically change.

5. RESULTS AND ANALYSIS

Home screen



Run unoptimized (by using normal map reducer concept). Here we are running 3 types of jobs (word count, sorting and creating inverted index)

(in the above, the total time taken by mapper and reducer to process the job is represented as processing time in ms and their individual timings in ns)

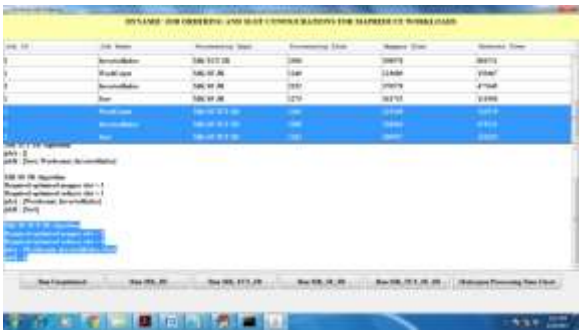
Run MK_JR:

As per this algorithm, we order jobs in J based on the following principles:
Partition jobs set J into two disjoint sub-sets JobA and JobB:
JobA = when T(m) <= T(r)
JobB = when T(m) > T(r)

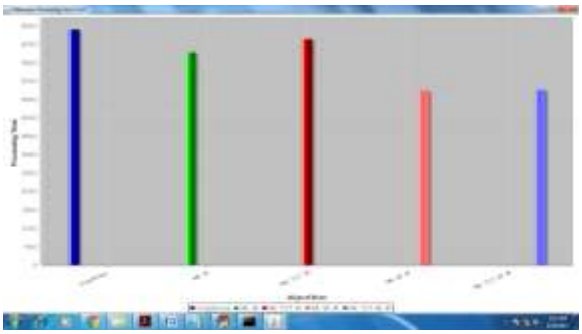
Run MK_TCT_JR: This algorithm is similar to MK_JR but the jobs will be arranged on the time threshold:

Run MK_SF_JR:
Here it shows how many slots required processes the given jobs:

Run MK_TCT_SF_JR:
Here it is similar to above but time threshold is there



The processing time comparison chart:



6. CONCLUSIONS

This paper 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 makespan, therefore, we further propose a new greedy job ordering algorithm and a map/reduce slot configuration algorithm to minimize the makespanand 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, we conduct extensive experiments to validate the effectiveness of our proposed algorithms and their theoretical results.

REFERENCES

[1] Amazon ec2 [Online]. Available: http://aws.amazon.com/ec2, 2015.

[2] Apache hadoop [Online]. Available: http://hadoop.apache.org, 2015.

[3] Howmanymapsandreduces [Online]. Available: http://wiki. apache.org/hadoop/HowManyMapsAndReduces, 2014.

[4] Lognormal distribution [Online]. Available: http://en.wikipedia. org/wiki/Log-normal_distribution, 2015.

[5] The scheduling problem [Online]. Available:http://riot.ieor. berkeley.edu/Applications/Scheduling/algorithms.html, 1999.

[6] Nandigama, N. C. (2019). Hybrid Facial Recognition System Using Histogram of Oriented Gradients and Deep Learning with Dimensionality Reduction. Research Journal of Nanoscience and Engineering, 3(4), 30–35. https://doi.org/10.22259/2637-5591.0304005

[7] 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.

[8] Nandigama, N. C. (2021). Advancing Underwater Image Segmentation through Pix2Pix Generative Adversarial Networks. Research Journal of Nanoscience and Engineering, 5(1), 20–25. https://doi.org/10.22259/2637-5591.0501004

[9] Vikram, S. (2023). Enhancing Credential Security in Distributed Manufacturing: Machine Learning for Monitoring and Preventing Unauthorized Client Certificate Sharing. JOURNAL OF ADVANCE AND FUTURE RESEARCH, 1(7). https://doi.org/10.56975/jaafr.v1i7.501709

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

[11] Todupunuri, A. (2023). The Role of Artificial Intelligence in Enhancing Cybersecurity Measures in Online Banking Using AI. International Journal of Enhanced Research in Management & Computer Applications, 12(01), 103–108. https://doi.org/10.55948/ijermca.2023.01015

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

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

[14] 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.