IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering

ISO 3297:2007 Certified

Vol. 5, Issue 12, December 2016

Optimization of Slot and Map Reduce Workload

Vinayak Kadam[1], Rutuja Aughad[2], Priyanka Gaikwad[3], Jidnyasa khanore[4], Pragati Naykodi[5]

Professor, Computer Dept., JSPM, Pune, India [1]

Student, Computer Dept., JSPM, Pune, India[2, 3, 4, 5]

Abstract: The increasing use of internet leads to handle lots of data by internet service providers. MapReduce is one of the goodsolutions for implementing large scale distributed data application. AMapReduce workload generally contains a set of jobs, each of which consists of multiple map tasks followed by multiple reducetasks. 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 executionconstraints 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. Makespanand total completion time are two key performancemetrics T his paper proposes two algorithm for these two key. Our first class of algorithms focuses onthe job ordering optimization for a MapReduce workload under a given map/reduce slot configuration. Our second class ofalgorithms considers the scenario that we can perform optimization for map/reduce slot configuration for a MapReduce workload.

Keywords: MapReduce, Hadoop, Flow-shops, Scheduling algorithm, Job ordering.

INTRODUCTION

A MapReduce job consists of a set of map and reduce tasks, where reduce tasksare performed after the map tasks. Hadoop [2], an open source implementation of MapReduce, has been deployed in largeclusters containing thousands of machines by companies such as Amazon and Facebook. Make span and total completion time are two key performance metrics. Generally, make span is defined as the timeperiod since the start of the first job until the completion ofthe last job for a set of jobs. It considers the computationtime of jobs and is often used to measure the performance andutilization efficiency of a system. In contrast, total completiontime is referred to as the sum of completed time periods for alljobs since the start of the first job. It is a generalized make span with queuing time (i.e., waiting time) included. We can use itto measure the satisfaction to the system from a single job'sperspective through dividing the total completion time by thenumber of jobs (i.e., average completion time). Therefore, inthis paper, we aim to optimize these two metrics

Objectives:-
- To improve the performance for MapReduce workloads with job ordering and slot configuration optimization approaches.
- Propose slot configuration algorithms for make span and total completion time.
- Perform extensive experiments to validate the effectiveness of proposed algorithms and theoretical results.

EXISTING SYSTEM

Scheduling: Given the distributed nature of most data analytics systems, scheduling thequery execution plan makes it an important part of the system. Systems must now take severalscheduling decisions, including scheduling where to run each computation, scheduling inter-nodedata transfers, as well as scheduling rolling updates and maintenance tasks.Many researchers have worked on optimization work for MapReduce jobs, and paid attention on computation scheduling and resource allocation topics of the same. Also many authors considered job ordering optimization for MapReduce workloads. The modelingof the MapReduce as a two-stage hybrid flow is described in. This hybrid flow shop has multiprocessor tasks, where job submission orders affect the results of cluster utilization and system performance. The execution time formapping and reducing the tasks for each job must be known earlier, but this phenomenon is not implemented in the applications. Also this method has not considered for the dependent jobs and suitable only for the independent jobs. Example of such method is MapReduce workflow.

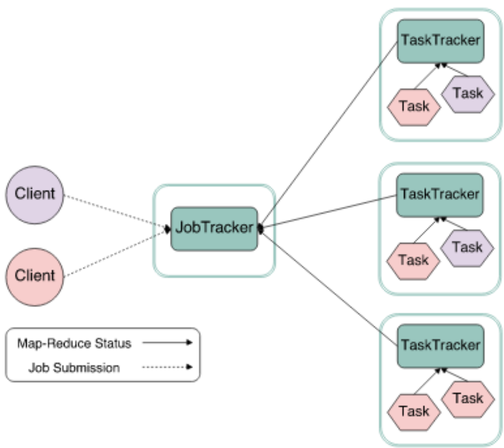

Fig: Existing system

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 12, December 2016

RELATED WORK

MapReduce [1] is a programming model and an associated implementation for processing and generating large datasets. Users specify amap function that Processes a key/ value pair to generate a set of intermediate key/ value pairs, and are duce function that merges all Intermediate values associated with the same intermediate key.

The Apache Hadoop [2] software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures. The problem of map-reduceScheduling [3] by abstracting the above requirements and desiderata inscheduling terms. In particular, we focus on multiple-task multiple-machine two-stage non-migratory scheduling with precedence constraints; these constraints exist between each map task and reducetask for a job.We consider a subset of [4]production workload that consists of MapReduce jobs with nodependencies. We observe that the order in which these jobsare executed can have a significant impact on their overallcompletion time and the cluster resource utilization. Our goalis to automate the design of a job schedule that minimizes thecompletion time (makespan) of such a set of MapReduce jobs. We consider the impact of thearchitectural design of MapReduce,[5] including programming model, storage-independent design and scheduling. In particular, we identify have factors that affect the performanceof MapReduce: I/O mode, indexing, data parsing, groupingschemes and block-level scheduling. Performance optimization for MapReduce jobs is a very attention captivating topic for researchers. We survey some of the relating topic to our proposed work.

ALGORITHM

1. Scheduling and Resource Allocation Optimization :-
Compared to this phenomenon, our proposed method is suitable for all types of jobs. Starfish [6] framework can modify the hadoopconfiguration automatically for the MapReduce jobs. By using sampling technique and cost based model we can maximize the utilization of hadoop cluster. But still we can improve the performance of this technique by maximizing the utilization of map and by reducing slots.[11] proposed a technique for MapReduce multi job workloads based on resource aware scheduling technique. This technique focus on improving resource utilization by expanding the abstraction of existing task slot to job solve the inefficiency problem of the Hadoop MRv1 in the perspective of resource management. Instead of using slot, it manages resources into containers. The Map and Reduce operation are performed on any container.

2. Speculative Execution Optimization :-
In MapReduce we need task scheduling strategy for dealing with problems such as straggler problem for a single job, which include [8], Speculative execution is such an important task scheduling strategy. The speculative execution algorithm speculates the task by prioritizing and pays attention on heterogeneous environments. To run, selecting the fast nodes and the speculative tasks are covered over, this speculative execution algorithm is a longest approximate time to end (LATE) [13], and the prioritizing of task is required for speculation. Guo et al. [9] proposes a Benefit Aware Speculative Execution (BASE) algorithm which evaluate the potential benefit of the speculative tasks and the unnecessary runs are eliminated. This BASE algorithm of the evaluating and elimination can improve the performance for LATE. The speculative execution strategy magnifies its focus mainly on saving cluster computing resource. Maximum Cost Performance (MCP) is a new speculative execution algorithm proposed by the proposed for fixing the problem that was affecting the performance of the prior speculative execution strategies. We proposed speculative Execution Optimization strategy that balances the tradeoffs between a single job and a group of jobs.

3. Slot Pre-Scheduling:-
It improves the slot utilization efficiency and performance by improving the data locality for map tasks while keeping the fairness.
Step 1: Compute load factor mapSlotsLoadFactor = Pending map tasks +running map tasksfrom all jobs divided by the cluster map slot capacity.
Step 2: Compute current maximum number of usable map Slots = number ofmap slots in a tasktracker* minmapSlotsLoadFactor, 1.
Step 3: Compute current allowable idle map (or reduce) slots
For a tasktracker= maximum number of usable map slots - current number ofused map (or reduce) slots.

PROPOSED SYSTEM

A. Problem Definition
To maximize the slot utilization for MapReduceand balancethe performance tradeoff between a single job and a batch of jobs with fair scheduling and improving the performance of MapReduce cluster in Hadoop.

B. Goals and Objective
The objective is to utilize the slots in MapReduce cluster. The slot utilization remains a challenging task due to fairness and resource requirements. It is fair when all pools have been allocated with the same amount of resources. The resources requirements between themap slot and reduce slot are generally different.This isbecause the map task and reduce task are often exhibit completely different execution patterns.
We review job ordering optimization. To model performance of system, makespan and total completion

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 12, December 2016

time is used. Total time taken to complete job is calculated.We describethe dynamic slot allocation frameworkthat produces the optimized job orderand also prove its approximation ratio. We also describethe job order which gives the worst, i.e., longest makespan,which is used for derivation of the upper bound makespanof a workload.We propose an alternative technique called dynamic hadoop slot allocation by keeping the slot based model. It relaxes the slot allocation constraint to allow slots to be reallocated to either map or reduce tasks depending on their needs. Second, the speculative execution can tackle the straggler problem, which is shown to improve the performance for single job but at the expense of the clustering. In the view, we propose speculative execution performance balancing to balance performance trade-off between single job and a batch of jobs. Third, delay scheduling has shown to improve the data locality but at the cost of fairness. Finally, by combining these techniques together, we form step by step slot allocation system called Dynamic MR that can improve performance of map reduce workloads substantially.

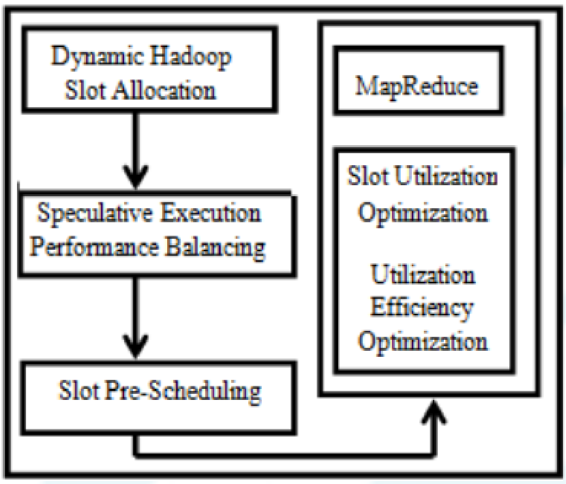

Figure 2: Overview of the Proposed System

Software requirement:-

Operating System : Windows 10
Technology : Java, J2EE
Web Technologies : Html, JavaScript, CSS
IDE : My Eclipse
Web Server : Tomcat
Database : My SQL
Java Version : J2SDK 1.7 / 1.8

Hardware requirement:-

Hardware - Pentium
Speed- 1.1 GHz
RAM - 1GB
Hard Disk - 20 GB
Floppy Drive - 1.44 MB
Key Board - Standard Windows Keyboard
Mouse - Two or Three Button Mouse
Computer - 3 Pc

CONCLUSION

Dynamic slot configuration is one of the important factorswhile processing a large data set with MapReduce paradigm. It optimizes the performance of MapReduceframework. Each job can be scheduled using any one of the scheduling policiesby the job tracker.The task managerswhich are presentin the task tracker allocateslots to jobs.From the examined paper,it is concluded to prefer a dynamic slot allocation strategy that includes active jobs workload estimation, optimal slot assignment, and scheduling policy.

REFERENCES

[1] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters, In Proceedings of the 6th Symposiumon OperatingSystemsDesign and Implementation (OSDI), 2004.
[2] Hadoop. http://hadoop.apache.org.
[3] B. Moseley, A. Dasgupta, R. Kumar, T. Sarl, On scheduling in map-reduce and flow-shops. SPAA, pp. 289-298, 2011.
[4] A. Verma, L. Cherkasova, R. Campbell. Two Sides of a Coin: Optimizing the Schedule of MapReduce Jobs to Minimize Their Makespan and Improve Cluster Performance. MASCOTS 2012.
[5] J. Dittrich, J.-A. Quiane-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad.Hadoop++: Making a Yellow Elephant Run Like a Cheetah, PVLDB,3(1), 2010.
[6] D.W. Jiang, B.C. Ooi, L. Shi, and S. Wu.The Performance of MapReduce:AnIndepth Study, PVLDB, 3:472-483, 2010.
[7] B. Moseley, A. Dasgupta, R. Kumar, T. Sarl, On scheduling in map-reduce and flow-shops. In SPAA'11, pp. 289-298, 2011.
[8] A. Verma, L. Cherkasova, R.H. Campbell, Orchestrating an Ensemble of MapReduce Jobs for Minimizing Their Makespan, IEEE Transaction on dependency and secure computing, 2013.
[9] A. Verma, L. Cherkasova, R. Campbell. Two Sides of a Coin: Optimizing the Schedule of MapReduce Jobs to Minimize Their Makespan and Improve Cluster Performance. In IEEE MASCOTS, pp. 11-18, 2012.
[10] S.J. Tang, B.S. Lee, and B.S. He. MROrder: Flexible Job Ordering Optimization for Online MapReduce Workloads. In Euro- Par'13, pp. 291 -304, 2013.
[11] S.J. Tang, B.S. Lee, R. Fan and B.S. He. Dynamic Job Ordering and Slot Configurations for MapReduce Workloads, CORR (Technical Report), 2013.