ISSN 2395-1621

Dynamic Slot Allocation In Hadoop

[#1]Miss.NeetaMohite, [#2]Miss.ShubhangiMahandule, [#3]Prof.V.N.Nandgaonkar

[1]mahanduleshubhangi@gmail.com
[2]Neeta.mohite1@gmail.com
[3]Vikas.nandgaonkar@gmail.com

[#123]B.E. Computer Engineering, Universal College of Engineering ,Sasewadi (Pune)

ABSTRACT

In recent years , advent of new technologies , devices, smart phone and communication like social media sites, the amount of data produced is growing rapidely every year. To hareness the power of big data , you would require an infrastructure that can manage and process huge volumes of structured and unstructured data in real time and can protect data privacy and security.Apache hadoop is an open source software framework written in java for distributed storage and distributed processing of very large datasets on computer cluster. Usully Map Reduce designed for processing data of files. It is a framework which we can write applicaton to process huge amount of data ,in parallel on large cluster in rerliable manner.However, the slot-based Map Reduce system (e.g., Hadoop MRv1) can suffer from poor performance due to its un-optimized resource allocation. To address it, this paper identifies and optimizes the resource allocation from three key aspects. First, due to the pre-configuration of distinct map slots and reduce slots which are not fungible, slots can be severely under-utilized. Because map slots may be fully utilized while slots are empty and vice-versa. We propose alternative technique called Dynamic Hadoop Slot Allocation by keeping slot based model.

Keywords: Map Reduce , DHSA , MRV2 , PostScheduling, PreScheduling Technique,post Scheduling.

ARTICLE INFO

Article History

Received :28th April 2016

Received in revised form :

30th April 2016

Accepted : 2nd May 2016

Published online :

4th May 2016

I. INTRODUCTION

Hadoop is java based framework that allows to process large data sets in distributed environment. Hadoop has been used by many large scale companies like Amazon, Facebook, and Yahoo[2]. Hadoop consist of two important concepts:Hadoop Distributed File System (HDFS) and Hadoop Map Reduce. Map Reduce workloads may be very heterogeneous in terms of theirdata size and their re- source requirements , and mixing them within a single instanceof a computing framework may lead to conflicting optimization goals. Therefore,isolating Map Reduce workloads and their data while dynamically balancing theresources across them is very attractive for many organizations .Hadoop is an open source framework that allows to store and process big data in a distributed environment across clusters of computers using simple programming mode[5]l. It is designedto scale up from single servers to thousands of machines, each offering local computationand storage. Our System relaxes the slot allocation constraint to allow slots tobe reallocated to either map or reduce tasks depending on their needs. Second, thespeculative execution can tackle the straggler problem, which has shown to improvethe performance for a single job but at the expense of the cluster efficiency[4]. In viewof this, we propose Speculative Execution Performance Balancing to balance theperformance tradeoff between a single job and a batch of jobs. Third, delay schedulinghas shown to improve the data locality but at the cost of fairness. Alternatively,we propose a technique called Slot PreScheduling that can improve the data localitybut with no impact on fairness. Finally, by combining these techniques together, weform a step-by-step slot allocation system called Dynamic MR that can improve theperformance of Map Reduce workloads substantially.

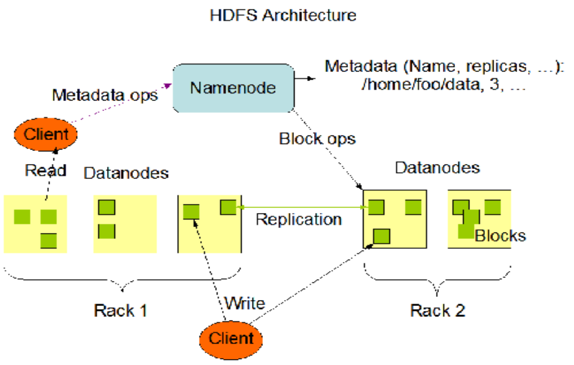
II. BASIC CONCEPT

2.1 Map Reduce:

Map Reduce is a processing technique and a program model for distributed computing based on java. It contains two important tasks, namely Map and Reduce. The major advantages of MapReduce is that it is easy to scale data processing over multiple computing nodes.

2.2 Hadoop Distributed File System(HDFS):

It is distributed file system designed to run on commodity hardware. This system provides high-throughput access to application data. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. Application that run on HDFS has large data sets.[4] Typically file in HDFS is gigabytes to terabytes in size. It should support tens of millions of files in a single instance. HDFS is designed more for batch process in grather than interactive use by users. Detection of faults and quick, automatic recovery from them is a core goal of HDFS. HDFS has been designed to e easily poratable from one platform to another. HDFS has a Master-slave architecture [6]. An HDFS cluster consist of a single NameNode, a master serves that manages the file system namespaces and regulates access to files by clients. In addition, there are number of datanodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on.



III.BACKGROUND AND COMPARATIVE ANALYSIS

MR1 architecture, the cluster was managed by a service called the Job Tracker. Task Tracker services lived on each node and would launch tasks on behalf of jobs. The Job Tracker would serve information about completed jobs.[9]**MRv1 uses the Job Tracker** to create and assign tasks to task trackers, which can become a resource bottleneck when the cluster scales out far enough (usually around 4,000 clusters).

3.1 Limitation:-

1)**It limits scalability:** Job Tracker runs on single machine doing several task like

- Resource management
- Job and task scheduling and
- Monitoring

Although there are so many machines (Data Node) available; they are not getting used. This limits scalability.

2) **Availability Issue:** In Hadoop 1.0, Job Tracker is single Point of availability. This means if Job Tracker fails, all jobs must restart.

3)**Problem with Resource Utilization:** In Hadoop 1.0, there is concept of predefined number of map slots and reduce slots for each Task Trackers. Resource Utilization issues occur because maps slots might be 'full' while reduce slots is empty (and vice-versa). Here the compute resources (Data Node) could sit idle which are reserved for Reduce slots even when there is immediate need for those resources to be used as Mapper slots.

3.2 Map Reduce: Difference between MR1 and MR2:

Earlier version of map- reduce framework in Hadoop 1.0 is called as **MR1**. The new version of Map Reduce is known as **MR2**.

No more Job Tracker and Task Tracker needed in Hadoop 2. With the introduction of YARN in Hadoop2, the term Job Tracker and Task Tracker disappeared. Map Reduce is now streamlined to perform processing data.

The new model is more isolated and scalable as compared to the earlier MR1 system. MR2 is one kind of distributed application that run Map Reduce framework on top of YARN. Map Reduce perform data processing via YARN. Other tools can also perform data processing via YARN. Hence Yarn execution model is more generic than earlier Map Reduce model.

MR1 was not able to do so. It would only run Map Reduce applications

IV. PROPOSED SYSTEM

4.1 Slot prescheduling:-

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 o 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.

4.2 Dynamic Hadoop Slot Allocation:-

It attempt to maximize slot utilization while maintainingthe fairness, when there are pending tasks (e.g., map tasks orreduce tasks). We break implicit assumption of MapReducethat the maptasks can only run on map slots & reduce taskscan only run on reduce slots. In our proposed system wemodify it that map and reduce tasks can berun on either mapor reduce slots. There are 3 cases, Consider, NM = Total number of Map tasks NR = Total number of Reduce tasks SM = Total number of map slots SR = Total number of reduce slots

Case 1: NM ≤ SMandNR ≤ SR The map tasks which are running on map slots and reduce tasks are run on reduce slots, There is no borrowing of map and reduce slots.

Case 2: NM>SMandNR<SR We satisfy reduce tasks for reduce slots first and then use those idle reduce slots for running map tasks.

Case 3: NM<SMandNR>SR We can schedule those unused map slots for running reducetasks. Case 4: NM>SMandNR>SR The system should be in completely busy state.

4.3 Delay time scheduler :

- Time threshold decide on hadoopnamenode datanode configuration .
- After delay time data will be temporary stored on pool and slot factorization will be done.
- Before delay time slot allocation will be occurred by standard hadoop configuration

4.4 Features

- Stores large database at the same time it can analyze the data using Map Reduce Algorithm.
- Hadoop processes data fast which is very useful for Real Time System .
- Improves the performance of Map Reduce workloads with maintaining the fairness.
- Balances the performance trade-off between a single job & a batch of jobs dynamically.
- Slot pre-scheduling improves the efficiency of slot utilization by further maximizing its data locality.
- SEPB identify the slot inefficiency problem of speculative execution.

4.5 Application

- Providing Dynamic MR over Hadoop Framework as a service to IT companies.
- Providing our software as a service to Government System.
- Providing our system to any end-user or company needing Hadoop on multi-node cluster.
- Providing our software as a solution to any company having big data handling issues

V. WORK FLOW

The admin login to system then upload to file.

First, we can classify the slots into two types, namely, busy slots (i.e., with running tasks) and idle slots (i.e., no running tasks). Given the total number of map and reduce slots configured by users, one optimization approach (i.e., macro-level optimization) is to improve the slot utilization by maximizing the number of busy slots and reducing the number of idle slots. Second, it is worth noting that not every busy slot can be efficiently utilized. Thus, our optimization approach (i.e., micro-level optimization) is to improve the utilization efficiency of busy slots after the macro level optimization. Particularly, we identify two main affecting factors(1). Speculative tasks . (2). Data locality . Based on these, we propose Dynamic MR, a dynamic utilization optimization framework for Map Reduce, to improve the performance of a shared Hadoop cluster under a fair scheduling between users.

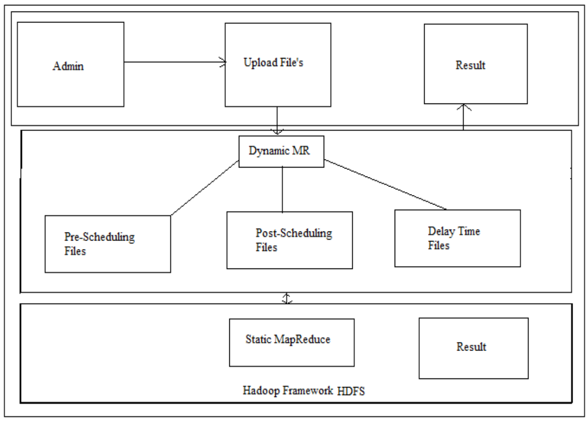


Fig5.1 Workflow of System

VI. RESULT

In This section we have shown the working of the proposed system .the fig 6.1 shows the total files how to use Memory load and fig 6.2 shows the total files how to use CPU load.

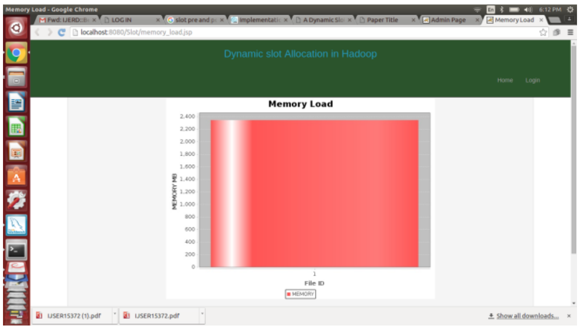


Fig 6.1Memory load

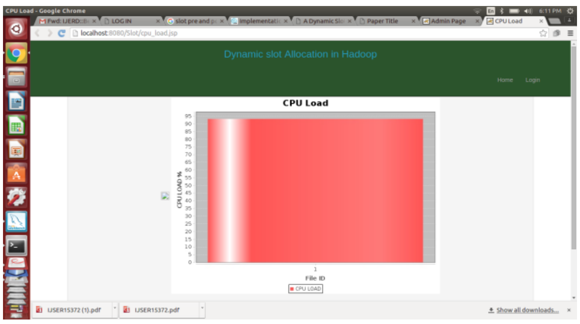


Fig6.2 CPU Load

VII. CONCLUSION

The aim of the proposed system is to improve the performance of Map Reduce workloads. It considered three techniques: Dynamic Hadoop Slot Allocation, Speculative Execution Performance Balancing, and Slot Pre-Scheduling. Dynamic Hadoop Slot Allocation uses allocation of map to maximize the slot utilization and it reduces the task dynamically. It does not require any prior information or any assumption and it can be run on any kind of Map reduce jobs. Speculative Execution Performance Balancing identifies the slot inefficiency problem. It manages the balance between single and batch of jobs dynamically. Slot Pre-Scheduling are used to enhance the efficiency of slot utilization by maximizing data locality. We can enhance the utilization by adding above concept in traditional system. A good trade-off betweendata utility and data consistency.

REFERENCE

[1]MapReduceTutorial ,http://hadoop.apache.org/docs/r1.2.1/mapred tutorial:html

[2]F:Ahmad; S:Y:Lee; M:T hottethodi; T:N:v ijaykumar :P UMA : Purdue Map Reduce Benchmarks Sui

[3]G:Ananthanarayanan; S:Kandula; A:Greenberg; I:Stoica; Y:Lu; B:Saha; andE:Harris; Reininginth

[4]Apache HadoopNextGenMapReduce(Y ARN):http : ==hadoop:apache:org=docs=current=hadoop yarn=hadoop yarn site=YARN:html:

[5]J:Chao; R:Buyya:MapReduceP rogramming Model for :NET Based Cloud Com putting :In EuroP ar09; pp:417 428;2009:

[6]Q:Chen; C:Liu; Z:Xiao; Improving Map Reduce Performance Using Smart Speculative Execution

[7]J:DeanandS:Ghemawat:MapReduce : Simplified Data Processing on Large Clusters; In OSDI 04; pp 113; 2004:

[8]Z:H:Guo; G:F ox; M:Zhou; Y:Ruan:ImprovingResourceUtilizationinMapReduce:InIEEECluster1410; 2012:

[9]Z:H:Guo; G:F ox; andM:Zhou:InvestigationofdatalocalityandfairnessinMapReduce:InMapReduc 32; 2012:

[10]Z:H:Guo; G:F ox; and M:Zhou:InvestigationofDataLocalityinMapReduce:InIEEE=ACMCCGri 426; 2012:

[11]Hadoop:http : ==hadoop:apache:org:

[12]M:HammoudandM:F:Sakr:Locality Aware Reduce Task Scheduling for Map Reduce :In IEEECL 576; 2011:

[13]M:Hammoud; M:S:Rehman; M:F:Sakr:Center of Gravity Reduce Task Scheduling to Lower Map 58; 2012:

[14]H:Herodotou; H:Lim; G:Luo; N:Borisov; L:Dong; F:B:Cetin; and S:Babu:Starfish :A Self tuningSystemforBigDataAnalytics:InCIDR11; pp:261C 272; 2011:

[15]H:HerodotouandS:Babu; P rofiling; What if Analysis; and Cost based Optimization of Map Reduce

[16]SIbrahim; HJin; LLu; BHe; SWu:AdaptiveDiskI=OSchedulingforMapReduceinVirtualizedEnv 344; 2011:

[17]Y:C:Kwon; M:Balazinska; B:Howe; andJ:Rolia:SkewTune : mitigatingskewinmapreduceapplicat 36; 2012:

