IHadoop: Improve MapReduce Performance

Mohammed Qunper, Osama Badawy, Mohammed Kholief College of Computing and Information Technology, Arab Academy for Science, Technology, and Maritime Transport, Alexandria, Egypt
m.qunper@gmail.com, obadawy@aast.edu, kholief@aast.edu

Abstract: *The performance of Hadoop dependent on many of points, the data partitioning is one of this point, the node specification is the other point. In the real world, the data is often highly skewed, which may cause losing time for the jobs. In this paper we study the skew problem in reduce and map phases, where map phase need to collect bocks and reduce phase need to collect key groups. We outline our solution to develop map and reduce phases, by decrease preparation Maper and Reducer, so we select locality base partitioning and developed to solve power loss by using node specification to decrease map time, and key group robust to decrease reduce time*.

Keywords: *Hadoop, MapReduce, Big data, Cloud computing, Parallel computing*.

1. Introduction

Over the years, a lot of online applications have led to the production of a huge amount of data; a challenge involves retrieving the information of these resources with high performance. Hadoop [1] is one of open source implementations, backed by yahoo, Facebook and other famous company, has been widely employed in academia and in industry, is highly fault tolerant and has large throughput. MapReduce [2] framework has had most popularity for its benefits and features such as scalability, reliability, high throughput, analysis and large computations on these massive amounts of data. Fundamentally; a MapReduce has two primary phases to execute a job, map and reduce phases. Actually, to process big data sets need to increase utilization and decrease power by cloud computing environments, and need to divide the data into fixed size chunks which are processed in parallel by MapReduce. Although MapReduce succeeds; however, there are many challenges such as data skew problem [3], interaction fallen, reduce-phase skew problem [4]. And also the performance and power implications of the integrated environment are still not well investigated, so there are a lot of researches on this popular model, to improve its performance.

In this paper, we put our focus on the mechanism before Map phase layer, and before Reduce phase layer of each task in a job. Each job has three phases to complete the process [5]. Namely, preparing phase, running phase, finishing phase. Preparing phase reads a collection of input data split from HDFS and generates task tracker and job tracker to execute MapReduce tasks, in running phase each job are waiting to be scheduled for execution then take a time to execute tasks, and the finishing phase cleanup task is scheduled to a task tracker then the job becomes successful [6]. In the first two phases there are many obstacles lead to a reduction of efficiency. One of the reasons is data partitioning[3], where they cause the content network resource busy then job completion time will increase, other reason a varying number of intermediate key value pairs that assigned to reducer [4] , leads to skew the load in Reduce phase . According to MapReduce mechanism, data splitting controls the Map completion time and key group controls running of the Reduce task. There are several challenges that have to be well studied to take advantage a MapReduce mechanism with the best performance. We propose integrate the two solutions with some modifications to solve a wasting job time, and solve map skew and reduce skew. The main contributions of our work are summarized as follows. **First**, our theoretical analysis shows that reduce the wasted time, and the node hardware is a one of main attribute to run a job. **Second**, using LDB method to improve map skew, based on the basis of novel splitting mechanism to address the remote read problem according to node availability. **Third**, we integrate two compatible methods to improvement MapReduce lifetime.

2. Related Work

Even though MapReduce framework has been the most popular .However, the performance and power implications of the integrated environment are still not well investigated; therefore, there are many research works on MapReduce. A lot of efforts have been made to improve the performance of Hadoop at the level of job scheduling or job parameter optimization.

2.1 MapReduce Framework

Figure 1 shows the work flow of a MapReduce operation, a client posts jobs to the master node to process files; it assigns JobTrackers to coordinate map and reduce phases provide job progress information. Over multiple slave nodes, JobTrackers are regrouping, stored, and saved into information files. Both phases have inputs and outputs, a key-value pair list, The transactions are handled by a JobTracker daemon, that runs the initial data partitioning and the intermediate data combination, by posting tasks of type Map and type Reduce over the TaskTracker daemons of the nodes involved in the cluster, according to the data being processed. The Reduce phase only starts when on finishing the Map, caused after the Map the resulting keys are combined, to distribute a sorted list of key-value pairs between the Reducers, which can be matched at the end of them. The process is transactional, those map or reduce tasks are not executed, (for data availability issues) will be reattempted a number of times, and then redistributed to other nodes [7].

After dividing file into M splits by districted file system by HDFS [8] or Gfarm [9],[10]; fork many copies of program on different machines. A master machine assigns either map or reduce task to any idle worker machine. Map worker read splits pass each pair to user's Map function, buffer result in memory. The Map worker periodically writes buffered pairs to local disk, partitioned into many of regions and updates master with region locations and sizes; master remembers these.

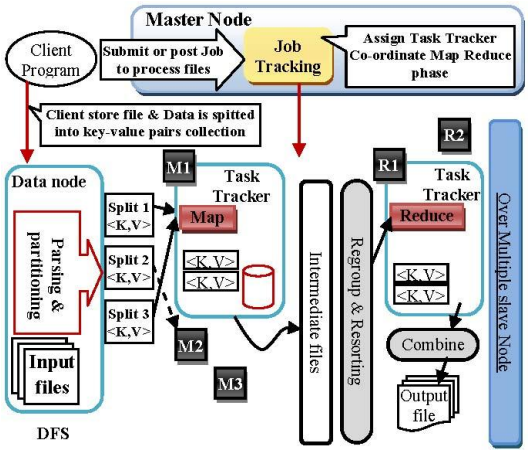


Figure 1 MapReduce Workflow

The reduce worker sends a remote procedure call to the map workers to read pairs from map workers' disks, and sorts pairs by intermediate key and group by intermediate key, for each intermediate key, pass all intermediate values to user's Reduce function. When all reduce tasks is finished Master node wakes up user program.

2.2 Skew Problem

The skew problem is highly variable task runtimes in MapReduce applications [11]. Runtime task distributions from these applications demonstrate the presence and negative impact of skew on performance behavior. We introduce some works about avoiding such behavior and their limitations.

Data Locality is one of critical impacts on data parallel performance; therefore, many researchers have been promoted to address this challenge [2] [3]. Streaming data [12] from multiple available replicas have been proposed to improve the remote data accessing performance. Locality Based Partitioning [3] Strategy can cluster blocks co-located in the same node into one partition, it releases data skew and improves the MapReduce processing performance. Shuffle: The procedure reduces fetching the immediate data from each map task after launching is called copy phase, and reduce sorting and merging the input date fetched from map tasks is called merge phase. Usually, shuffle phase means the whole procedure consisting of sort and merge

The shuffle phase consists of sort phase and merge phase. Whereas; reducer fetching the immediate data from each map task after launching is called copy phase. Reduce sorting and merging the input date fetched from map tasks is called merge phase. It is one of the reasons the performances problem, the shuffle stage was the main cause of network traffic. Jingui Li and etc. [13] developed the shuffle stage with more efficient I/O policy; to decrease the whole job's

execution time and make full use of cluster resources. Sketch-based data structure [4] proposed to capture MapReduce key group size statistics and present an optimal packing algorithm which assigns the key groups to the reducers in a load balancing manner.

2.3 Map Phase Improvements

After distributing files by HDFS or Gfarm or other distributed file system, MapReduce is running programs in parallel on top data splits. But, distributed file system is designed to store data across many nodes or servers for load balance. In practice, its balance is not well for each file distribution [3] . Figure 2 shows, distributed file effects.

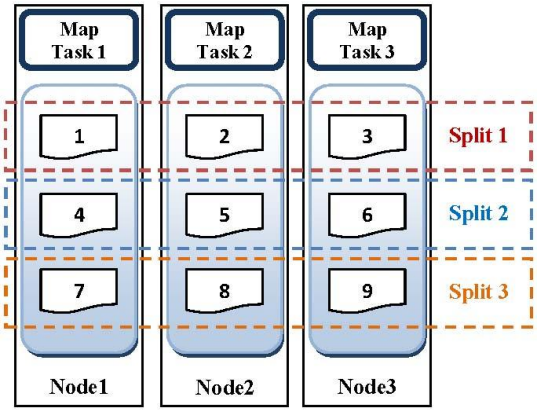
Hash-base partitioning in Figure 2 (a), has sets of disadvantages [3]

- May split block across nodes into the same partition, which will cause non-local map tasks reading data cross nodes/racks from network
- Increases the job own completion time
- Content network resource is busy

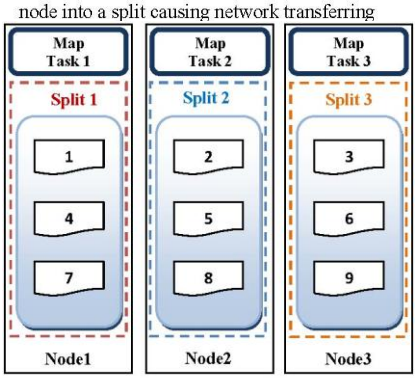
- Multiple jobs will interfere with each other for sequenced jobs will be burdened by one job's improper partitioning

When map task reads data splits; which causes a lot of TaskTrackers wasting time for initialization and reallocate slots.

Many researchers improve the defects by their technologies, Wang etc. [3] exploited data allocation technologies to select best worker node and present some research directions and challenges. Gu etc.[6] Propose others technologies to optimize task execution mechanism.



(a) Hash based portioning will split block across-node into a split causing network transferring



(b) Locality Base Partition strategy can cluster blocks co-located in the same node to one partition

Figure 2 Data Splitting Controls

2.4 Reducer Phase Improvements

Once the map phase is completed and its results have been transferred to the reducers, the reduce phase begins. In this phase, the reduce function is applied in parallel to each key group and produces the final results. Many works upgrade the reduce phase performance by their methods. Streaming input data from multiple replicas [12] one of improvement methods to remote data access and accelerate the map phase, a sketch-based compact data profiling [14] is another method to solve the data skew problem in record linkage. Yan etc. [4] introduce a novel sketch-based data structure as a base method to improve reduce lifetime.

3. IHadoop

3.1 Hadoop Background

In the MapReduce framework, to compute the job execution performance, we need to compute Maper workload and Reducer Workload. Maper Workload is computed as sum of collecting time of all partitions to Map task. Reducer workload is computed as a sum of workload of all key groups assigned to Reducer task.

Based on the above and shown in Figure 3 Job life time, the execution mechanisms of a MapReduce are fully integrated between Map and Reduce tasks; therefore, the two critical limitations in the standard Hadoop MapReduce framework that effect execution performance.

- In first limitation improve the splitting to decrease initialization Map phase time
- In second limitation improve the splitting to decrease initialization Reduce phase time

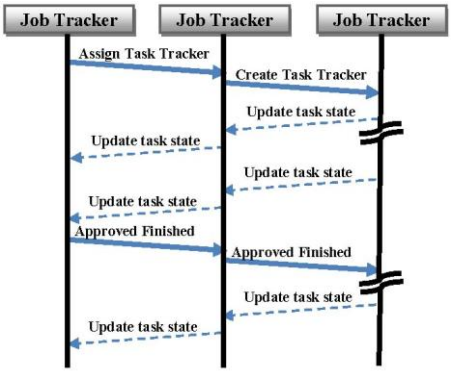


Figure 3 Job life time

3.2 Proposed System

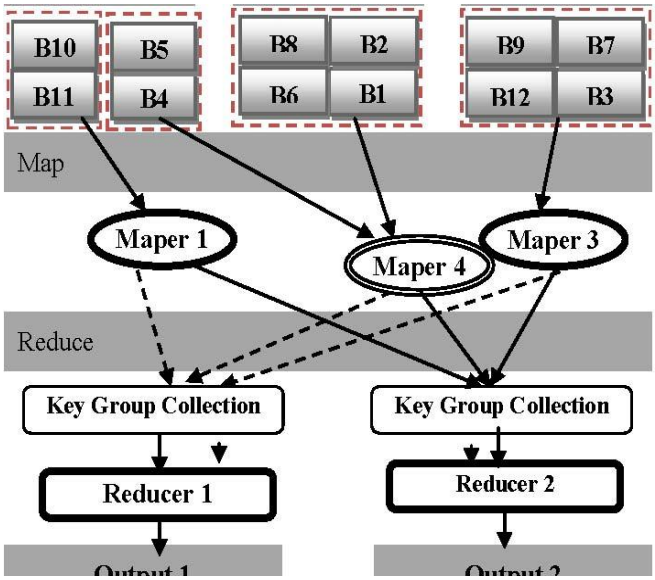
We propose solutions depended on hardware architecture Hadoop and improve both main tasks Map and Reduce; we will be integrated locality base Partitioning based on the cores of nodes mechanism as interface to collect splits to increase Map task initialization and solve LBP power loss, with an

interface to collect key groups and sort it to increase Reduce task initialization. In this section, we introduce the notation of locality base Partitioning based on the cores of nodes, and present its construction, and describe sorting key group packing interface.

As a first step we present an overview of our system illustrating how our modifications work in MapReduce framework. Show in Figure 4

Figure 4 illustrates the MapReduce processing phases and sequences. Data partitioning is a step processed before Map phase, then before starting Maper we need to select available node depended on node structure, Reducer will be started after the results of Maper is available, key group collection is being processed before reduce phase.

3.3 Locality base Partitioning based on the cores of nodes



In this section, we describe our improving Local Base Partitioning mechanism by detecting processors and others hardware configuration in nodes to solve power loss. LBP clusters data blocks co-located in the same node, to address the problems of original splitting method of Hadoop implementation and improve the MapReduce performance [3], but it ignores power loss or node architecture configuration.

There are two points have to be answered: which node is available to work and which blocks and their replicas are collected into the same partition. Wang Answered the second part by designing LBP system, we have benefited from his answer and add an answer to the first part.

The JobTracker will be consulted for corresponding nodes utilization to decide which node replica we choose and which node is available. The detail steps of Locality Based Partition based on Hardware are described in Algorithm 1. SplitList is an array list whose each array contains the blocks of the same partition, and there are SplitMemNum arrays in this list (Step 4). In step 6 we choose the proper replica for every block which we will describe the detailed steps of this in next subsection. Then we cluster the replicas into partitions based on its locality and number of processor in the node (step 8- step 16).

- Splitting Numbers
- Replica Selection
- Processor utilization

3.4 Sorting key group packing interface

In this section, we introduce the notion of sketch-based key group size [4] into the MapReduce framework and a distributed method for its construction. Yan and Xue [4] investigate the design of the partition function, which maps the intermediate key to a reducer index, based on the key group size information as a summary in the global sketch.

In the Algorithm 2 K be the key space and R be the number of reducers. A partition function Φ:K→{1, 2, …..,R} maps key k to the index of the desired reducer r = Φ(k) ∈ {1, 2, ...,R}. We assume the reducer load is proportional to its input key group sizes. As such, the load at reducer r can easily be derived as $S_{k:\Phi(k)=r_k}$, where S_k is the size of key group k. The targets of this method are the load balance at different reducers, and minimize the task time by 30% as a minimum.

4. Conclusion and future work

Despite its popularity deployed, Hadoop also have some problems that have to be studied and addressed. In this paper, we analyze the disadvantages of existing splitting mechanisms, losing time before reducer task. We have benefited from previous studies to develop and integrate them to decrease from 30% to 60 on completion time of jobs. **Future Works in this paper,** we are developing the idea by the following steps; implement and applying the idea, developing its parts to be smarter

References

[1] Welcome to Apache™ Hadoop®! [Online]. http://hadoop.apache.org

[2] Sanjay Ghemawat Jeffrey Dean, "MapReduce: Simplified Data Processing on Large Clusters," vol. 51, 2008.

[3] Qingbo Wu, Yusong Tan, Wenzhu Wang, Quanyuan Wu Chunguang Wang, "Locality Based Data Partitioning in MapReduce," *IEEE 16th International Conference on Computational Science and Engineering*, 2013.

[4] Yuan Xue, Bradley Malin Wei Yan, "Scalable and Robust Key Group Size Estimation For Reducer Load Balancing in MapReduce," *IEEE International Conference on Big Data*, 2013.

[5] Hadoop Internals. [Online].

http://ercoppa.github.io/HadoopInternals/AnatomyMapReduceJob.html

[6] Xiaoliang Yang, Jinshuang Yan, Yuanhao Sun, Bing Wang,Chunfeng Yuan, Yihua Huang Rong Gu, "SHadoop: Improving MapReduce performance by optimizing job execution mechanism in Hadoop clusters," Journal of Parallel and Distributed Computing, vol. 74, no. 3, 2014.

[7] Platform base: HDFS + MR. [Online]. http://hadooper.blogspot.com/

[8] Tom White, Hadoop: The Definitive Guide.: O'Reilly Media, 2011.

[9] Osamu Tatebe. (2013) Gfarm: Present Status and Future Evolution.

[10] Gfarm Document. [Online]. http://datafarm.apgrid.org/

[11] Magdalena Balazinska, Bill Howe,Jerome Rolia YongChul Kwon, "A Study of Skew in MapReduce Applications," *Open Cirrus Summit* , 2011.

[12] Bo Hong Jiadong Wu, "Improving MapReduce Performance by Streaming Input Data From Multiple Replicas," *IEEE International Conference on Cloud Computing Technology and Science*, 2013.

[13] Xuelian Lin, Xiaolong Cui, Yue Ye Jingui Li, "Improving the Shuffle of Hadoop MapReduce," *IEEE International Conference on Cloud Computing Technology and Science*, 2013.

[14] Wei Yan, Yuan Xue, and Bradley Malin, *Scalable Load Balancing for MapReduce-based Record Linkage*.

[15] Apache Hadoop , Wikipedia. [Online]. http://en.wikipedia.org/wiki/Apache_Hadoop

[16] PoweredBy - Hadoop Wiki. [Online]. http://wiki.apache.org/hadoop/PoweredBy

[17] Gfarm file system - Wiki. [Online]. http://en.wikipedia.org/wiki/Gfarm_file_system

[18] Hadoop. [Online]. http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html