Ishuffle: Improving Hadoop Performance of the Mapreduce Framework Using EWMA (Exponentially Weighted Moving Average)

C. Jayabarathi
PG Scholar, Department of Computer Applications, P. R. Engineering College, Vallam, Anna University, Chennai

Abstract: Hadoop is a popular implementation of map reduce framework. It is automatically parallelize job execution with the concurrent map and reduce task. Mapping phase is used to read the input. Data in each split is passed to mapping to produce the value of output. Shuffling consume the output of map phase and it is the process of transfer the mapper from 0&1 or more. The input data fetching phase between the map and reduce phase can affect the performance of job. In reducing phase the output value from shuffling are aggregated. In propose I used to decouple the shuffle from reduced task convert it into a platform service. Present iShuffle, a user transparent shuffle service that pro-actively pushes map output data into node through a novel shuffle on write using EWMA. Floods workload trace show that iShuffle reduces job completion time.

I. INTRODUCTION

Hadoop mapreduce is a software for easily writing application that process a huge amount of data simultaneously on large cluster. Mapreduce is used normally for split the input data into independent that are processed by map task in parallel manner. The input data into independent that are processed by map task in parallel manner. The input and output are always stored in the file system. It will monitoring the task. Mapreduce framework as single master and single slave in per cluster node. Decentralized execution is important for hadoop scalability to a large number of node. In existing studies they focused on improving the map task performance. For example, work has been done to preserve locality through map scheduling or input are reproduce as a data. While there is exploiting the parallelism and improving the efficiency in map tasks. Work tries to overlap shuffle with map by sending map output or fetching the data. Shuffling map is the process of transferring the data from mapper to reducers is known as shuffling. Map reduce shuffle phase is necessary for the reducer, otherwise, they would not have any input. The input data is fetching between the map task and reduce task that affect the performance of job. In shuffle it takes the more to complete the job and also wastage of resource. Propose iShuffle to decouple the shuffle from the reduce task and convert it into platform service provided by hadoop. And also I present the iShuffle user transparent shuffle that are used to push the map output into node through the novel shuffle on write using EWMA. EWMA used to seeks to investigate the performance of the Exponentially Weighted Moving-Average Chart (EWMA) for mining big data and detection of DDoS attacks in Internet of Things (IoT) infrastructure. Representative workloads and Floods workload trace show that iShuffle reduces job completion time by as much as 29.6 and 34 percent in single-user and multi-user clusters, respectively can able to overlap the shuffle phase and map phase.

1. Hadoop Acceleration through Network Levitated Merge:

Hadoop is a popular open-source implementation of the Map Reduce programming model for cloud computing.

These include a serialization barrier that delays the reduce phase, repetitive merges and disk access, and lack of capability to leverage latest high speed interconnects.

A novel network-levitated merge algorithm is introduced to merge data without repetition and disk access.

In addition, a full pipeline is designed to overlap the shuffle, merge and reduce phases.

Technique Used:

Hadoop acceleration a full pipeline overlap the shuffle, merge and reduce phases:

We describe Hadoop-A, an acceleration framework in a novel network-levitated merge algorithm to merge data without repetition and disk access.

Hadoop-A doubles the data processing throughput of Hadoop, and reduces CPU utilization by more than 36%.

Disadvantages:

Delays of reduce phase
Repetitive merges
Disk access
Lack of capability to leverage latest high speed interconnects.

2. HybridMR: A Hierarchical Map Reduce Scheduler For Hybrid Data Centers:

Virtualized environments are attractive because they simplify cluster management, while facilitating cost-effective.

As a result, virtual machines in public clouds or private data centers have become the norm for running transactional applications like web services and virtual desktops.

And propose a 2-phase hierarchical scheduler, called HybridMR, for the effective resource management of interactive and batch workloads.
Technique Used:
HybridMR manages the run-time performance of MapReduce jobs collocated with interactive applications in order to provide best effort delivery to batch jobs.

while complying with the Service Level Agreements (SLAs) of interactive applications. These results indicate that a hybrid data center with an efficient scheduling mechanism can provide a cost-effective solution for hosting both batch and interactive workloads.

Disadvantages:
Effective resource management
Slow Scheduling.
Effect of workload.

3. Spongefiles: Mitigating Data Skew in Mapreduce Using Distributed Memory:
Data skew (Basically the data is not evenly distributed across partitions an uneven distribution degrades the performance of the overall execution as CPUs are sitting idle and waiting for some other partitions to finish their job with larger volumes.) is a major problem for data processing platforms like Map Reduce.

Skew causes worker tasks to spill to disk what they cannot fit in memory, which slows down the task and the overall job.

Moreover, performance of other jobs sharing same disk degrades. In many cases, this situation occurs even as the cluster has plenty of spare memory.

Technique Used:
Sponge files Data Skew Using Distributed Memory: We introduce Sponge Files, a novel distributed-memory abstraction tailored to data processing environments like Map Reduce.

A Sponge File is a logical byte array, comprised of large chunks that can be stored in a variety of locations in the cluster.

By enabling memory-sapped nodes to tap into the spare capacity of their neighbours, Sponge Files minimize expensive disk spilling, thereby improving performance.

Disadvantages:
Cannot fit in memory
Degrades overall job
Slows down the task

II. HADOOP MAPREDUCE FRAMEWORK
Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. It spawns one map for each that are generated by the input format for the job.

Ishuffle:
Here I propose the ishuffle, that push the map output to the reduce task. It is easily retrieve the output from map task. Automatically balance the map output data. Ishuffle decouple the shuffle from the reduce task. And perform independently. In ishuffle I use the data as flood.

User Transparent Shuffle Service:
User transparent ishuffle is compact to the hadoop jobs in existing. At the end master and slave as job independent components. It increasing the hadoop performance. It consists of two node master and slave. Job tracker is used to track the data from shuffle manager and shuffler. The shuffler as two task reduced and map task, master send a data to slave and the slave collect the data from master and produce the output. Any user job can use ishuffle service without any modifications.

Shuffle on Write:
The shuffle on write is implemented by the shuffler that push the map output to reduce node for future use.
The data are written to local disk during every task. Output of map data can be performed before the execution of reduced task. Input data are sent to map and then map send the data to memory buffer. Buffer is used to indicate that the data is available or not. Then all data is stored on local disk. The local disk is eliminate the redundant data and then produce the output with redundant.

**Design Trade-Off Ishuffle:**

The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. HDFS is highly fault-tolerant the ishuffle alternate the workflow of hadoop, it also affect the fault tolerance in hadoop and **a tradeoff** is where one thing increases and another must decrease.

It is monitor the sending data incase if there is some data are are failed during the when sending. It monitor and resend the failed data on different nodes. it monitor the data from source to destination. it will help the user to find any fault occur during the data send.

**Reducing Shuffle Delay:**

Shuffle delay is used to calculate the duration between the last wave and the next reduce wave. Also measure the period of shuffle. Shuffle delay is more flexible in shuffling scheme. It consists of some benchmark like Hadoop-A, DynMR, sailfish and ishuffle, and store the time of start and complete of each map.

**Reducing Job Completion Time:**

It is used to reduce the completion of job timing with the above benchmarks. Use the job completion time in stock Hadoop implementation as the baseline and compare the normalized performance of IShuffle, Sailfish, DynMR and Hadoop-A the normalized job completion time of all shuffle heavy Benchmarks, which are self-join, tera-sort, page rank, and bayes.

**EWMA (Exponentially Weighted Moving Average):**

This used to seeks to investigate the performance of the Exponentially Weighted Moving-Average Chart (EWMA) for mining big data and detection of DDoS attacks in Internet of Things (IoT) infrastructure. EWMA is a control charts for variables data (data that is both quantitative and continuous in measurement, such as a measured dimension or time). It plots weighted moving average values. EWMA control charts use information from all samples, they detect much smaller process shifts than a normal control chart. DDoS attack, need for simple, fast and effective protection that can guard their web presence, protect their brand and secure their revenues.

**Sheduling Algorithm:**

Scheduling algorithm is used for scheduling the data from output.here is the example of scheduling using priority algorithm. It is associated with each process. Cpu allocated to the process with highest priority. If equal, use FCFS (first come, first serve)

**Example:**

<table>
<thead>
<tr>
<th>Countries</th>
<th>Flood Data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Partition Algorithm:
Partition algorithm is used to reduce the time of execution. The partition phase take place after the map phase and before the reduce phase. The partitioner partition the key value pairs the intermediate map output. The data are user defined, that works like a hash function. The total number of partition is same as the number of reduce tasks for the job.

Example:

<table>
<thead>
<tr>
<th>Country</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>India</td>
<td>1</td>
</tr>
<tr>
<td>Russia</td>
<td>2</td>
</tr>
<tr>
<td>America</td>
<td>3</td>
</tr>
<tr>
<td>Germany</td>
<td>4</td>
</tr>
</tbody>
</table>

Input Data:
The data is saved as input.txt in the hadoop partitioner directory.

<table>
<thead>
<tr>
<th>Country</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>India</td>
<td>1</td>
</tr>
<tr>
<td>Russia</td>
<td>2</td>
</tr>
<tr>
<td>America</td>
<td>3</td>
</tr>
<tr>
<td>Germany</td>
<td>4</td>
</tr>
<tr>
<td>Australia</td>
<td>5</td>
</tr>
</tbody>
</table>

Here i calculate the value of input data from the list using split function that separate the countries and store in string variable.

Merge Sort Algorithm:
Merge sort algorithm is used to sort the value in ascending order. It merge all value and sort the value in right manner.

Example:

16, 32, 9, 25, 18 next it will combine and compare the value with each other. And merge them into a list of founded data value.

9, 16, 25, 32 this is called merge sort.

III. CONCLUSION
Hadoop design poses challenges to attain the best performance in job execution. In this paper, I propose iShuffle, a novel user-transparent shuffle service that provides optimized data shuffling to improve job performance. It decouples shuffle from reduce tasks that pushes data to be shuffled to Hadoop node via a novel shuffle-on-write operation in map tasks. iShuffle further optimizes the scheduling of reduce tasks. We implemented iShuffle as a configurable plug-in in Hadoop and evaluated its effectiveness on a 32-node cluster with various workloads.

IV. REFERENCE