International Journal of Innovative Research in Advanced Engineering (IJIRAE)
Volume 1 Issue 1 (April 2014)

A Brief on MapReduce Performance

Kamble Ashwini
Information Technology Department, DCOER
Pune University
ashwinikamble1992@gmail.com

Kanawade Bhavana
Computer Department DCOER,
Pune university
brkanawade@gmail.com

Abstract*— Hadoop is an open- source implementation of MapReduce. Hadoop is useful for storing the large volume of data into Hadoop Distributed File System (In distribute Manner) and that data get processed by MapReduce model in parallel. MapReduce is scalable and efficient programming model to perform large scale data intensive application. In Homogeneous Environment, it decreases the data transmission overhead because Hadoop schedules data location and all nodes have ideal work with computing speed in a cluster. Unhappily, it's difficult to take data locality in heterogeneous or shared environments. The performance of MapReduce is improved using data Prefetching mechanism which can fetch the data from slower remote node to faster node and as a result job execution time is reduced. This mechanism hides the overhead of data processing and data transmission when data is not local. Data prefetching design reduces data transmission overhead effectively.

***Keywords*— Hadoop, HDFS, MapReduce, Serial Transmission, Data Prefetching, Data transmission**

I. Introduction

Hadoop was created by Doug cutting in 2005. The name was invented, is of toy elephant of his son. At that time he was working at Yahoo Company [1]. Hadoop is an open-source software framework that supports data-intensive applications Like data mining, web indexing and web application(web crawler, web search engine, email, email auction).It is analyse structure and unstructured data format and distribute applications work on thousands of working independent computers. Volume of information TERABYTES (1TB=1000GB) TO PETABYTES (1PB=1000 TB) OR EXABYTES (1 EB= 1000 TB) stored at Hadoop distributed file system (HDFS) in distribute manner. Hadoop is cluster of nodes. Hadoop cluster have two main layers are HDFS means hadoop distributed file system useful for data storage in distribute mode and Map-Reduce means data processing programming model for large scale data processing .

In HDFS one master node is Name node which maintains the information about large scale file. HDFS have Metadata of files. Large file is distributed into number of blocks for storage. Each block size is fixed i.e. 64MB.These blocks are stored into number of data nodes depends upon local disk capacity of nodes. Set of data nodes into one cluster. Each data node called machine and each machine has different storage capacity. Every each machine writes the data into local disk. Many data nodes store the actual data. Each block is replicated N (default =3) times in data nodes. Balancing a load and replication these are features of HDFS system.

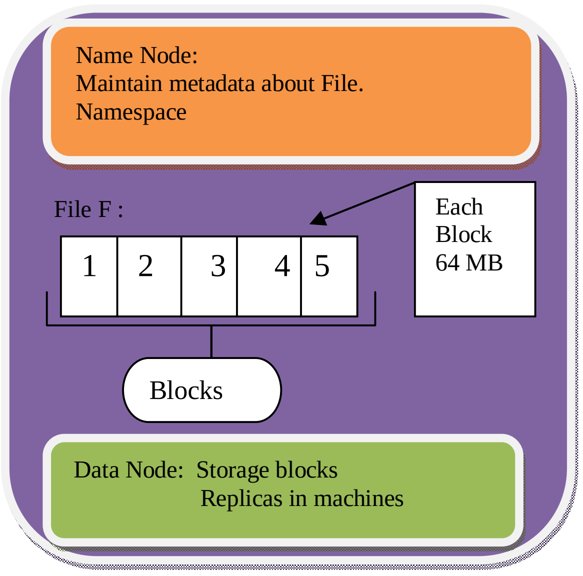


Fig1. Overview of HDFS

MapReduce is parallel programming model. It is created by Yahoo! And popularized by Google. MapReduce has becoming popular tool for processing large scale datasets in parallel with cluster of nodes. User wants a data processing system which is elastically as well as efficient. Main goal of MapReduce is hide the all description of parallel execution and allocate users to focus only on data processing strategies.

Today, Map Reduce model complete twenty petabytes data processing. Hadoop is an open – source implementation of MapReduce Framework, has adopted by several of companies including IBM, Yahoo, Face book, Apple [10, 11]. Hadoop is also used to support MapReduce data intensive Application like as data mining [12] and web indexing.

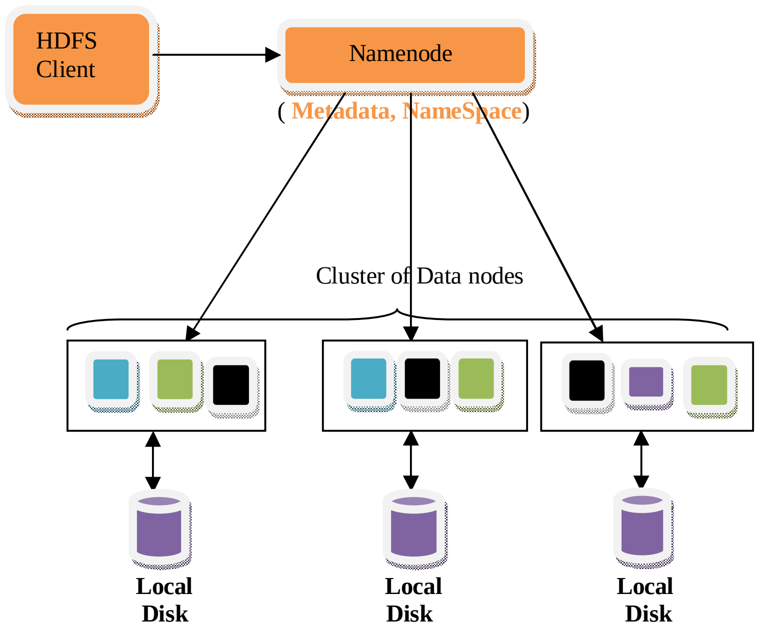


Fig.2 Hadoop distributed data storage

Data Transmission of data intensive application is affect on MapReduce Performance. The overhead of data transmission, Hadoop try to decrease collision of data transmission. For data locality in scheduling Hadoop works well in homogeneous and dedicated environment [2]. But in Heterogeneous environment difficult to Hadoop achieve good data locality. Data processing and data transmission serially perform in serial execution. Serial execution cannot avoid overhead of data processing with data transmission. To decrease the performance prevent by a data perfetching mechanism. A data prefetcher is designed for fetch input data from remote node. In data perfecting design overlapping data processing with data transmission in time dimension and hide the overhead of data transmission time. Improve the performance of MapReduce job execution time i.e. MapReduce performance is increases.

In section II, define MapReduce Importance. In section III states literature survey. The Mechanism is useful for improve MapReduce performance in section IV and Conclusion of final performance is in section V.

II. MAPREDUCE

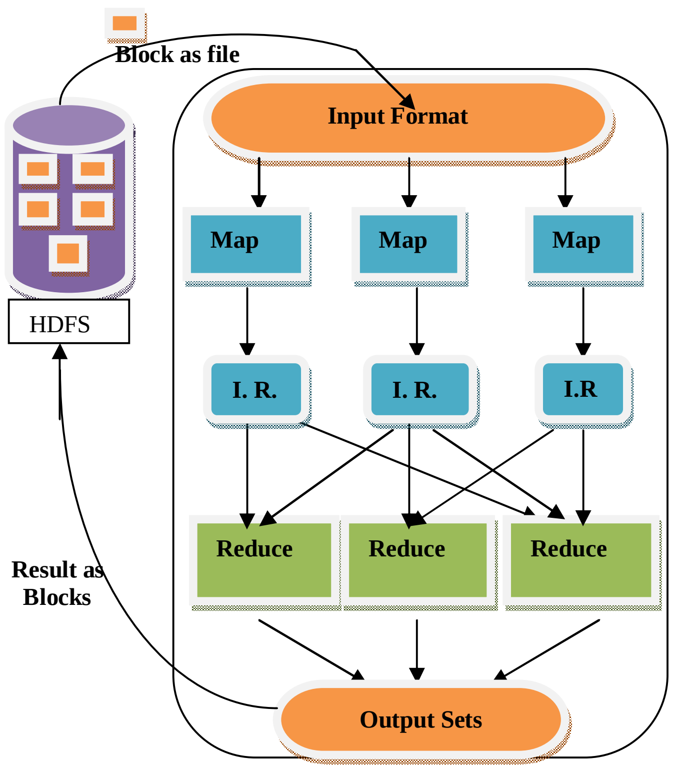
A. MapReduce Environment:

Hadoop implements MapReduce cluster in Master-Slave architecture [9], where job tracker as master node and task tracker as slave or worker nodes. The master node manages and schedules jobs to number of slave nodes, and slave nodes responsible for executing map and reduce tasks. When a job is submitted on master node, firstly master node divides the job into multiple map tasks and then schedules map and reduce task to appropriate nodes and keeps monitoring the state of tasks and nodes during the execution of jobs into multiple map tasks. A slave nodes runs tasks on its task slots and keeps updating task progress to master by periodic heartbeat.

MapReduce programming model have two functions: map ( ) and reduce ( ) [9]. The map and reduce functions are distributed across multiple machine. How many map functions in one node it's not depends upon how many nodes in cluster but depends upon input format for per node. Two map functions default in per node. The details of data processing in map and reduce phase can be design differently by respect to different requirement.

Map tasks extract <key, value> pairs from the input, transfer them to some user defined map function and Reduce function and finally generates the intermediate <key, value> pairs[3]. Shuffling for load balancing on nodes and sorting is useful for Reduce task. After That, the reduce tasks copy their input pieces from map task using same intermediate key then merge these pieces to a single ordered (key, value list) pair stream by a merge sort, transfer the stream to some user defined Reduce function, finally generate the result for job. In general, a map task divided into map and combine phases, while a reduce task is divided into copy, sort and reduce phase. In hadoop, reduce tasks can start when only map task complete, which allows reduce tasks to copy map outputs earlier as they becomes available and hence mitigates network congestion[9]. The input data for map tasks and the result of reduce tasks are stored into Hadoop Distributed File System (HDFS) [2].

Word count and CCV (Class-centre Vector evaluator) these are simple applications of MapReduce model [2].



I.R. = Intermediate Result
map (k1,v1) → list(k2,v2)
reduce (k2,list(v2)) → list(v2) [3]

Fig. 3 . MapReduce Programming Model

Example: Finding an open table in an open-seating arrangement at a restaurant. [4]

Ghirardelli's is a famous tourist destination in San Francisco, California that specializes in manufacturing chocolate as well as stuffing food. When most Families, group of friends go to Ghirardelli's they repeatedly stick together and keep walking around the place until a table clears. Splitting up and walking around in non-overlapping areas of the seating area is probably .Splitting up and walking around in non – overlapping areas of the seating area is probably more efficient. This is the map phase. When one person finds a free table big enough to seat their whole party, that person claims that table and uses an app like I- message or Group Me to text the other members of the party. Who are around the shortest path from final table then sort by shortest path. The other members then walk around the entire restaurant until they find their party. This concludes the reduce phase.

Two people find a table at exact same time. When this happens, the team's unofficially elected leader will walk to both tables and pick the better table based on location, comfort. The all other members relocate based on leader final decision.

III .LITERATURE SURVEY

One way to balancing load, Hadoop using HDFS distributed big size data to multiple nodes based on local disk storage capacity in clusters [13]. The data location is efficient in homogeneous environment where all nodes have identical both computing speed and disk capacity. In this environment computes same workload on all nodes representing that no data needs to be moved from one node to another node. All nodes are independent as well as can not share data between two nodes in cluster of homogeneous environment. In heterogeneous Environment or clusters have set of nodes where each node computing speed capacities and local disk capacity may be significantly different. If all nodes have different size workload then a faster computing (high performance) nodes can complete processing local data faster than slow computing (low- performance)nodes. Faster node finished processing data then result residing into its local disk and handle unprocessed data of remote slow node. When move or transfer unprocessed data from low performance (remote) node to high performance node is huge then overhead of data transmission is occurring. If wants Progress the MapReduce performance in heterogeneous environment then reduce the amount of data moved between low performances nodes to high performance nodes.

Improve The MapReduce performance in Various Environments:

A. Data Placement in Heterogeneous Hadoop Clusters
B. Heterogeneous Network Environments and Resource Utilization
C. Smart Speculative Execution Strategy
D. Longest Approximate Time to End.

A. *Improve MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters [5].*

We want improve the performance then minimize data movement between slow and fast nodes achieved by data placement scheme that distribute and store data across multiple heterogeneous nodes based on their computing speed.

1) Data placement in Heterogeneous- Two algorithms are implemented and incorporated into Hadoop HDFS. The first algorithm is to initially distribute file into heterogeneous nodes in a cluster. When all file fragments of an input files are distributed to the computing nodes. The second algorithm is used to reorganize file fragments to solve the data skew problem. There two cases in which file fragments must be reorganized. First, new computing nodes are added to an existing cluster to have the cluster expanded. When, new data is appended to an existing input file. In both cases, file fragments distributed by the initial data placement algorithm can be disrupted.

B. *Improving MapReduce Performance in Heterogeneous Network Environments and Resource Utilization [6]*

1) Resource stealing- When number of map and reduce slots are carefully chosen to gain optimal resource usage. Resource utilization is inefficient when there are not some enough tasks to fill all task slots as the reserved resources for idle slots are just wasted. Then Resource stealing, which enables running tasks to steal the residual resources and return them when new tasks are assigned. There is use of wasted resources to improve overall resource utilization and reduce job execution. First-come-Most(FCM) , Shortest-Time-Left-Most(STLM) , Longest –Time-Left-Most(LTLM) these are resource allocation policies.

2) Benefit Aware Speculative Execution –This mechanism predicts the benefit of launching new speculative tasks and greatly eliminates unnecessary runs of speculative tasks. Speculative execution in Hadoop was observed to be inefficient, which is caused by the excessive runs of useless speculative tasks. Benefit Aware Speculative Execution manages speculative tasks in a benefit-aware manner and expected to improve the efficiency.

C. *Improving MapReduce Performance using Smart Speculative Execution Strategy [7]*

Multiple speculative execution strategies are improving the performance in Heterogeneous as well as Homogeneous. But there are some Pitfalls degrade the performance. When existing strategies cannot work well, then they develop a new strategy, MCP (Maximum Cost Performance), which improves the effectiveness of speculative execution significantly.

When a machine takes an unusually long time to complete a task (the so-called straggler machine), it will delay the job execution time (the time from job initialized to job retired) and degrade the cluster throughput (the number of jobs completed per second in the cluster) significantly. This problem handled speculative execution. A new speculative execution strategy named MCP for Maximum Cost Performance. We consider the cost to be the computing resources occupied by tasks, while the performance to be the shortening of job execution time and the increase of the cluster throughput. MCP aims at selecting straggler tasks accurately and promptly and backing them up on proper worker nodes.MCP is quite scalable, which performs very well in both small clusters and large clusters.

D. *Improving MapReduce Performance in Heterogeneous Environments [8]*

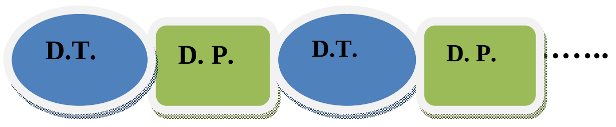
1) LATE, for Longest Approximate Time to End.

If nodes in cluster ran at reliable speeds and no cost to initiation a speculative task on an ideal node then LATE policy would be best. The LATE algorithm has various applications. First, it is dynamic to node heterogeneity, because it will relaunch only the low performance tasks and only a small number of fragment parts of large file with slow performing as tasks. LATE prioritizes amongst the slow tasks based on how much they injure job response time. LATE also capture the number of slow performance tasks to limit argument for shared resources. In contrast, Hadoop's native scheduler has a fixed threshold [8].

IV. TECHNIQUES TO IMPROVE MAPREDUCE PERFORMANCE

A. *Serial Execution:[2]*

For processing, input data to map tasks is not in one block, but in a small size pieces. Map task takes a small Piece of input data and starts processing on that data. When finishes the processing of this piece of data, map task begins transfer to reduce task and process another small piece for processing. This process repeats again and again until the map task complete processing all input data. In serial execution, data transformation and data processing in serial manner then transparency of data transformation cannot be avoided.



D.T. =Data Transmission
D.P. =Data Processing

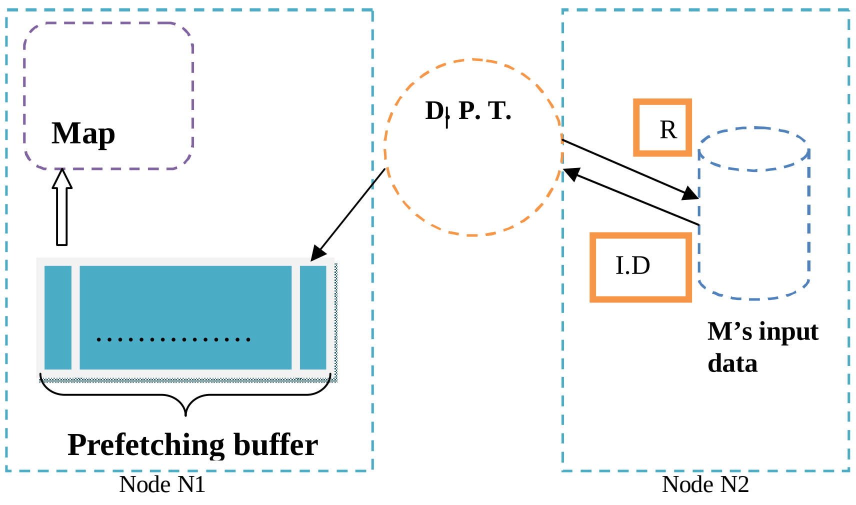
Fig. 4 . Serial Execution of data

Actually, data transmission is not allowed when data is processing in a map task. The data processing has to wait for the next piece of input data when data is transferred. Using data Prefetching mechanism hide the overhead of data transmission by overlapping the data transmission process with data processing process and overcome serial execution problem with improve the performance of MapReduce.

B. Data Prefetching Design[2]

When the input data of map tasks is not local, the overhead of data transmission is hidden [2] then reduce the job execution time. Data Prefetching design useful in MapReduce Model. Map () and Reduce () invocations are different machines in hadoop cluster. In this mechanism have two nodes N1 and N2.Map task M is assign to node N1 but that node don't have local data. Replication of M's input data is located into node N2.Node N1 has to fetch data from N2 for map processing. A data Prefetching thread is created for requesting corresponding input data and a buffer as Prefetching buffer is allocated on node N1 for storing fetched data temporarily.

This mechanism efficient in Heterogeneous environment because faster node finished local data processing and save into local disk after that data is not locally for a processing then using data prefetching design take a data from another remote slow node then automatically hide the overhead of data transmission from one slow node to fast node for data processing.



D.P.T. = Data Prefetching Thread
R = Request For Input Data
I.D. = Input Data

Fig.5 Data Prefetching Design for MapReduce

Above data prefetching mechanism process in steps for Map task of MapReduce model:

1. The data Prefetching thread acquire the location of input data, from node N2 (slower) to node N1(faster).
2. Working producer-consumer model like as producer produce data for consumer and consumer consumes that data i.e. where the data prefetching thread is the producer and the map task is the consumer.
3. Then, producer consumer approach: The data prefetching thread store fetching input data from node N2 to prefetching buffer. And the map task M keeps processing input data in Prefetching buffer.
4. Needed to be explained is the size of Prefetching buffer.[2]
5. Larger Prefetching buffer means better performance.
6. According to the producer-consumer model, two buffer units are enough but proposed prefetching mechanism needs little extra memory.

V. CONCLUSION

In this paper, improve the performance of MapReduce in Heterogeneous and shared environment decreases by data location mechanism. Then analyzing the process of Map tasks in Hadoop ,the serial execution of data transmission and data processing is discovered to overhead is occurs because data is not local . All improvement of MapReduce performance strategies is not suitable in Heterogeneous environment. These all problems overcome by data prefetching Mechanism. By using data prefetching mechanism fetch data from slow node to faster node in Heterogeneous and shared Environment. Hide the overhead of data transmission and data processing when data is not local. Reduce the job execution time by data prefetching design means improve the MapReduce programming model performance.

VI. **REFERENCES**

1. Hadoop Wiki Website, Apache, http://wiki.apache.org/hadoop
2. Improving MapReduce Performance Using Data Prefetching mechanism in heterogeneous or Shared Environments Tao gu,Chuang Zuo,Qun Liao , Yulu Yang and Tao Li, International Journal of grid and distributed computing (2013)
3. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Communications of the ACM, vol. 51, no. 1, (2008).
4. Map Reduce in Real Life - meetrajesh.com.html
5. J. Xie, S. Yin, X. Ruan, Z. Ding, Y. Tian, J. Majors, A. Manzanares and X. Qin, "Improving MapReduce Performance through Data Placement in Heterogeneous Hadoop Clusters", IEEE International Symposium on Parallel & Distributed Processing, Workshops and PhD Forum (IPDPSW), **(2010) April 19-23: Arlanta, USA**
6. Improving MapReduce Performance in Heterogeneous Network Environments and Resource Utilization, Zhenhua Guo, Geoffrey Fox IEEE (2012)
7. Improving MapReduce Performance Using Smart Speculative Execution Strategy Qi Chen, Cheng Liu, and Zhen Xiao, Senior Member, IEEE 0018-9340/13/$26.00 © 2013 IEEE
8. S. Khalil, S. A. Salem, S. Nassar and E. M. Saad, "Mapreduce Performance in Heterogeneous Environments: A Review", International Journal of Scientific & Engineering Research, vol. 4, no. 4, (2013).
9. Jiang, B. Chin, L. Shi and S. Wu, "The performance of MapReduce: an in-depth study", Proceedings of the VLDB Endowment. vol. 3, no. 1, (2010).
10. T. White, "Hadoop: The Definitive Guide (3rd Edition)", O'Reilly Media/Yahoo Press, (2012).
11. Hadoop Wiki Website, Apache, http://wiki.apache.org/hadoop/PoweredBy.
12. S. Papadimitriou and J. Sun, "DisCo: Distributed Co-clustering with Map-Reduce: A Case Study towards Petabyte-Scale End-to-End Mining", Eighth IEEE International Conference on Data Mining, (2008) December 15-19: Pisa, Italy.
13. "Improve the MapReduce Performance through complexity and performance based on data placement in Heterogeneous Hadoop Cluster " Rajashekhar M. Arasanal, Daanish U. Rumani Department of Computer Science University of Illinois at Urbana-Champaign