An Improved Map Reduced Framework for Mobile Computing

Dheeraj S Patil[1] Abhijit J Patankar[2] Nikita Singhi[3]
[1,2,3]Department of Computer Engineering
[1,3]Alard College of Engineering Pune, Pune University, India [2]D.Y.P.C.O.E Akurdi Pune, Pune University, India

Abstract— We had seen new technology trend over a last few decades where mobile communication has come to next level by upgrading its superfast computation by increased processor and memory. But still they lack behind the high computation software system to process big data. To overcome on this many researcher are working on to build a scalable platform with the help of Hadoop distributed storage and computational capabilities on clusters of commodity hardware. Building Hadoop on a mobile network allows the devices to run knowledge intensive computing applications while not direct information of underlying distributed systems complexities. Our aim is to build a scalable and integrated system using Hadoop HDFS file system to process high end and big data size application on mobile computing.
Keywords: Mobile Computing, Map Reduced Framework, MDFS

I. INTRODUCTION

Hadoop [1] could be a wide applied cloud computing platform that provides 2 main functions: Map Reduce [2] and the Hadoop Distributed classification system (HDFS) [3]. The previous performs the parallel computing, whereas the latter allows distributed file backups. Analysis on cloud computing is gift at intervals varied domains. A study in Norway established algorithms on Hadoop to store and analyze historical knowledge regarding rock oil and gas explorations, thereby enhancing the performance [4], and another analysis used the HDFS of Hadoop to save school data [5]. Several different studies have additionally integrated cloud computing to realize data processing [6], implement algorithms on Hadoop [7], and execute process biology. These studies concentrate on mistreatment Hadoop to resolve and enhance performance problems. There square measure solely a couple of studies have centered on Hadoop management. [8] Its terribly inconvenient for manager mistreatment the Secure Shell (SSH) [9] protocol to regulate and management Hadoop cluster. Though the Hadoop cluster info is displayed in a very web-based interface, there aren't any management functions. The Hadoop manager got to link to the Name node with associate SSH affiliation. The initial SSH affiliation info show in a very itinerant is restricted and incomplete. This work develops a Hadoop atmosphere management App for mechanical man phone, that allows directors to manage Hadoop clusters via 3G or Wi-Fi while not the time and placement restriction

II. RELATED WORK

The MapReduce of Hadoop is a parallel computation method to simplify the process of parallel programming. It is particularly effective for big data, as MapReduce divides the problem into a number of smaller problems and assigns them to each individual Mapper for processing. Subsequently, Reduce integrates the outputs of each Mapper to form the final result. Therefore, MapReduce is suitable for big data computing. It uses multiple nodes to increase computing capabilities and reduce computation time. MapReduce also provides another feature that solves the issue of data transfer from multiple nodes by sending the operational program to the data end for computation rather than sending the data to be processed in the computing system. This greatly reduces the time that have been required to transfer big data. Mapper uses key/value pairs to divide problems into many smaller problems. MapReduce distributes 100 students to each segment, which is assigned to an individual Mapper for computation. Reduce then performs the final executions and output.

Introduced the Hadoop based platform Hyrax for cloud computing on smartphones. Hadoop Task Tracker and Data Node processes were ported on Android mobile phones, while a single instance of Name Node and Job Tracker were run in a traditional server. Porting Hadoop processes directly onto mobile devices doesn't mitigate the problems faced in the mobile environment. As presented earlier, HDFS is not well suited for dynamic network scenarios. There is a need for a more lightweight file system which can adequately address dynamic network topology concerns. Another Map Reduce framework based on Python, Misco was implemented on Nokia mobile phones. It has a similar server-client model where the server keeps track of various user jobs and assigns them to workers on demand. Yet another server-client model based Map Reduce system was proposed over a cluster of mobile devices where the mobile client implements Map Reduce logic to retrieve work and produce results from the master node. The above solutions, however, do not solve the issues involved in data storage and processing of large datasets in the dynamic network.

HDFS is the fundamental file system of Hadoop. The large datasets are required to store within HDFS, which adopts distributed storage technologies to increase storage space and fault tolerance. It utilizes the advantages of parallel extension in Hadoop and low hardware requirements to improve space limitations. HDFS consists of Name node and Data node; the former is responsible for access control and Data node management, whereas Data node is responsible for data storage. Name node divides the data into multiple segments, each of which is 64 MB. Based on the set number of copies, backups are created for each segment and then stored on Data node. In this manner, fault tolerance is enhanced. As shown in the figure 1, a 192 MB file is divided by Name node into three segments. According to the settings, two duplicates are made and stored. Segment 1 is stored on Datanode1 and Datanode2; Segment 2 is stored on Datanode2 and Datanode3, and etc.

Hadoop originated from the Apache Nutch project, an open source web-search engine. In 2003 and 2004, Google published papers on GFS and MapReduce, resolving issues encountered in Nutch. Derived from these studies. Hadoop

have become a top-level project of Apache, which contains of different modules such as Common, Avro, MapReduce, HDFS, Pig, Thrift, Hive, HBase, Zookeeper, and Sqoop. Any user can use Hadoop to set up their own cloud computing application and environment. Hadoop primarily provides PaaS and IaaS services, which were written in Java. These services can be executed under multiple operating systems and provide distributed computing for exploiting large datasets. Hadoop-based cloud services are integrated with MapReduce, HDFS, and HBase.

In this paper, we present the inspiration and starter structure for a system to make Ad Hoc distributed computing suppliers. This system exploits the inescapability of cell phones, making a cloud among the gadgets in the region, enabling them to execute employments between the gadgets. The work displayed here is fundamental, and makes the establishment for future work.

HDFS is the key record arrangement of Hadoop. The huge datasets are required to store inside HDFS, which embraces conveyed capacity advances to expand extra room and adaptation to internal failure. It uses the upsides of parallel augmentation in Hadoop and low equipment prerequisites to improve space confinements. HDFS comprises of Name hub and Data hub; the previous is in charge of access control and Data hub the executives, though Data hub is in charge of information stockpiling. Name hub separates the information into various fragments, every one of which is 64 MB. In light of the set number of duplicates, reinforcements are made for each fragment and afterward put away on Data hub. As such, adaptation to non-critical failure is improved. As appeared in the figure 1, a 192 MB record is separated by Name hub into three sections. As per the settings, two copies are made and put away. Section 1 is put away on Datanode1 and Datanode2; Segment 2 is put away on Datanode2 and Datanode3, and so forth.

III. PROPOSED SYSTEM & ALGORITHM

We are developing a Map reduced framework over MDFS.

The traditional MDFS was primarily targeted for military operations where front line troops are provided with mobile devices.

A collection of mobile devices form a mobile ad-hoc network where each node can enter or move out of the network freely.

MDFS is built on a k-out-of-n framework which provides strong guarantees for data security and reliability. K-out-of-n enabled MDFS finds n storage nodes such that total expected transmission cost to k closest storage nodes is minimal.

Instead of relying on conventional schemes which encrypt the stored data per device, MDFS uses a group secret sharing scheme.

IV. MATHEMATICAL MODEL

Hadoop Map lessen structure fundamentally utilizes and depends on nearby information for expanding generally execution of the framework. Calculation is drawn nearer to the hubs where the information lives. Job Tracker first attempts to dole out assignments to Task Trackers in which the information is locally present (nearby). In the event that this is beyond the realm of imagination (no free guide spaces or on the off chance that errands have just bombed in the particular hub), it at that point attempts to allot assignments to different Task Trackers in a similar rack (non-nearby). This lessens the cross-switch system traffic accordingly diminishing the general execution time of Map errands. If there should arise an occurrence of non-nearby errand preparing, information must be gotten

Algorithm 1: Task Scheduling
Input: Sb, Fb, d, k Output: X,C⋆ i
// index b in C⋆
i (b) is omitted
C⋆ i = 0
X − 1 × N array initialized to 0
D − 1 × N array
for j=1 to N
do
D[j].node=j D[j].cost=(Sb/k) × Fb(j) × dij
if D[j].cost == 0
then D[j].cost=N2 // Just assign a big number
end
end
D − Sort D in increasing order by D.cost
for i=1 to k do
X[D[i].node]=1 C⋆ i += D[i].cost
End
return X,C⋆

When we work on mobile platform due to increase in higher network traffic which generally increases energy consumption and therefore fetching data from local drive will not be suitable or ideal which results in higher energy consumption and thus increasing transmission cost. That's why Map Reduce Task processing nearer to nodes that store the data for minimum latency and increased energy efficiency is more important on mobile platform.

There are many challenges in bringing data locality to MDFS. Unlike native Hadoop, no single node running MDFS has a complete data block; each node has at most one fragment of a block due to security reasons. Consequently, the default MapReduce scheduling algorithm that allocates processing tasks closer to data blocks does not apply. When MDFS performs a read operation to retrieve a file, it finds the k fragments that can be retrieved with the lowest data transferring energy. Specifically, the k fragments that are closest to the file requester in terms of the hop-count are retrieved. As a result, knowing the network topology (from the topology maintenance component in MDFS) and the locations of each fragment (from the fragments mapper). We could estimate the total hop-count for each node to retrieve the closest k fragments of the block. Smaller total hop-count indicates lower transmission time, lower transmission energy, and shorter job completion time.

Although this estimation adds a slight overhead, and is repeated again when MDFS actually retrieves/reads the data block, we leave the engineering optimization as the future work. We now describe how to find the minimal cost (hop-count) for fetching a block from a task Tracker.

Algorithm 1 illustrates the main change made to the default Hadoop Map Reduce scheduler such that the data transferring energy is taken into account when scheduling a

task. The improved scheduler uses the current network condition (topology and nodes 'failure probability) to estimate the task retrieval energy and assign tasks. Only nodes that are currently functional and available may be selected. C⋆i (b) is defined as the minimal cost of fetching block b at node i.

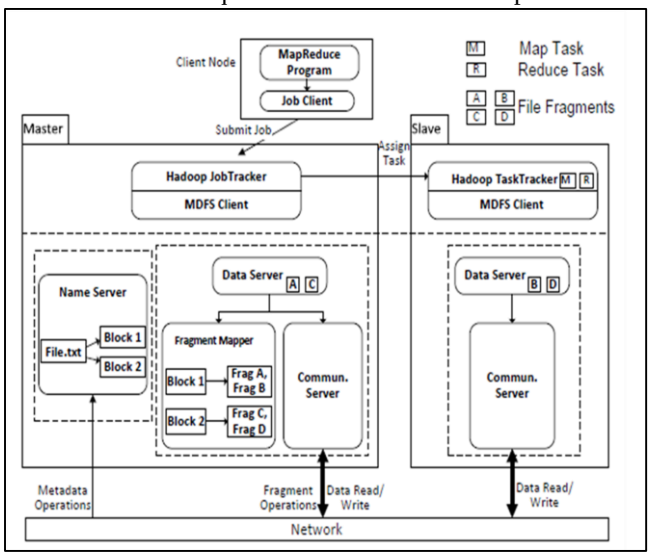
Let Fb be a 1×N binary vector where each element Fb(j) indicates whether node j contains a fragment of block b (note that PN j=1 Fb(j) = n 8b); Sb is the size of block b; di,j is the distance (hop-count) between node i and node j; the pair-wise node distance can be estimated efficiently using all pair shorted distance algorithm X is a 1 × N binary decision variable where Xj = 1 indicates that node j sends a data fragment to node i.

C⋆i (b) = min
N
X
j=1
(Sb/k)Fb(j)di,jXj , s.t.
N
X
j=1
Xj = k

C⋆i (b) can be solved by Algorithm 1, which minimizes the communication cost for node i to retrieve k data fragments of block b. Once C⋆ i (b) for each node is solved, the processing task for block b is then assigned to node p where p = argmini C⋆i (b). The time complexity of Algorithm 1 is NlogN due to the sorting procedure, and because it is performed once for each of the N node, the time complexity for assigning a block to a task Tracker is N2logN. Considering the size of our network (100) and the processing power of the modern mobile devices, the computational overhead of the scheduling algorithm is minimal.

V. METHODOLOGY

1) Step 1: The user issues a write request for a file of length L. The file is split into blocks of size [L/B] where B is the user configured block size. The last block might not be complete depending on the file length. The user request can also be a streaming write where the user writes to the file system byte by byte. Once the block boundary is reached or when the file is closed, the block is written to the network. In both scenarios, the data to be written is assumed to be present in the local file system.
2) Step 2: Similar to HDFS block allocation scheme, for each block to be written, the MDFS client requests the Name Server to allocate a new block Id which is a unique identifier for each block.
3) Step 3: The Name Server returns a new block id based on the allocation algorithm and adds the block identifier in its local cache. The mapping of file to list of blocks is stored in the Name Server
4) Steps 4-5: The MDFS client issues a creation request to the Data Server which contains a specific opcode in the request message. The Data Server identifies the opcode and instantiates the File Creator module to handle the block creation.
5) Step 6: The block stored in the local file system is encrypted using the secret key. The encrypted block is partitioned into n fragments using erasure encoding.
6) Step 7: The key is also split into fragments using Shamir's secret key sharing algorithm.
7) Steps 8-9: The Data Server requests the k-out-of-n framework to provide n storage nodes such that total expected transmission cost from any node to k closest storage nodes is minimal.
8) Step 10: The Data Server requests the Fragment Mapper to add the fragment information of each file which includes the fragment identifier with the new locations returned by the k-out-of-n framework.
9) Steps 11-18: The file fragments are distributed in parallel across the cluster. The key fragments are also stored in the same manner.
10) Steps 19-20: Once the file and key fragments are distributed across the cluster, the Data Server informs the client that the file has been successfully written to the nodes.
11) File Append Operation: MDFS supports Append operation which was introduced in Hadoop 0.19. If a user needs to write to an existing file, the file has to be open in append mode.
12) File Delete Operation: For a file to be deleted, all file fragments of every block of the file have to be deleted. When the user issues a file delete request, the MDFS client queries the Name Server for all the blocks of the file. It then requests the Data Server to delete these blocks from the network.
13) File Delete Operation: The File Rename operation requires only an update in the namespace where the file is referenced with the new path name instead of the old path.
14) File Directory Operation: When the user issues the file commands to create, delete or rename any directory, the MDFS client requests the Name Server to update the



VI. RESULTS

The inherent performance differences between the phones and servers in our testbed were investigated by comparing the speeds of four micro-benchmarks, each of which is bound by

CPU, memory, disk, or network resources. In the CPU benchmark, an empty loop is executed for some number of iterations. In the memory benchmark, a buffer is sequentially written to for some number of iterations and then sequentially read from. The times for these benchmarks were not significantly affected by the memory reads and writes, indicating that the memory benchmark was still CPU-bound on both the server and the phone. We concluded that the server is about 370 to 430 times faster than the phone for CPU-bound operations. In the disk benchmark, data is sequentially written to the server's hard disk and to the phone's flash card. Of all the system capabilities that were tested, the server and the phone are closest in disk access speeds. The server is 7.6 times faster for writes and 30 times faster for reads

In figure 1, Results shows transfer time with respect to amount of data transferred for server to phone, phone to phone, and phone to server transfers. The analysis shows transfer time with respect to amount of data transferred for small transfers (up to 128 KB).

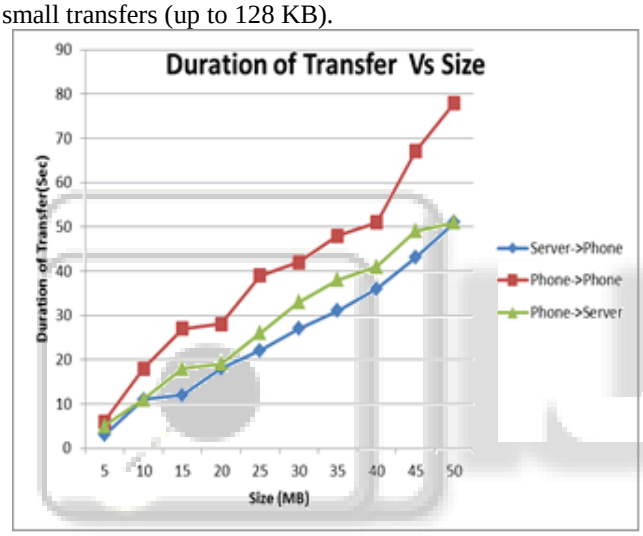


Fig. 1: Data Transfer time on various platforms

VII. Conclusions

The Hadoop Map Reduce system over MDFS shows the abilities of cell phones to gain by the relentless development of huge information in the portable condition. Our framework tends to every one of the requirements of information preparing in portable cloud vitality efficiency, information dependability and security. The assessment results demonstrate that our framework is competent for huge information investigation of unstructured information like media files, content and sensor information. In the event that static gadgets are accessible in the system, they can be organized over other versatile hubs in the bunch for information stockpiling as they are more averse to fizzle. Hadoop gives the majority of the fundamental highlights for a portable distributed computing framework. Moreover, there are a few arrangements given by Hadoop that can be legitimately applied to difficulties in a portable processing condition, for example, utilizing adaptation to non-critical failure for enduring hub takeoff.

Acknowledgment

So it gives me great pleasure, on the completion of this Paper, to acknowledge and appreciate all those who were there to help me. I express my sincere and profound thanks to all our teachers A.J.Patankar & Nikita Singhi (PG Coordinator and Guide).I would like to thank our college for the boost that it has provided.

References

[1] Hadoop-Apache Hadoop, Available from: http://hadoop.apache.org/.
[2] Gunarathne, T, 2010, "MapReduce in the Clouds for Science", Cloud Computing Technology and Science, 2010 IEEE Second International Conference on, pp.565-572, Dec. 2010.
[3] Oriani, A, 2012, "From Backup to Hot Standby: High Availability for HDFS, Reliable Distributed Systems", 2012 IEEE 31st Symposium on, pp.131-140, Oct. 2012.
[4] Baodong Jia, Wlodarczyk, T.W, Chunming Rong, 2010, "Performance IEEE Second International Conference on, pp. 5 Considerations of Data Acquisition in Hadoop System", Cloud Computing Technology and Science (Cloud Computing), 201045-549, 30 2010-Dec.
[5] Patrick Donnelly, Peter Bui., 2010, "Attaching Cloud Storage to a Campus Grid Using Parrot Chirp and Hadoop", Cloud Computing Technology and Science (Cloud Computing), 2010 IEEE Second International Conference on, pp.488-495, 30 2010-Dec.
[6] Yang Lai, Shi ZhongZhi, 2010, "An Efficient Data Mining Framework on Hadoop using Java Persistence API", Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on, pp.203-209, 29 2010-July.
[7] ChenHao, QiaoYing, 2011, "Research of Cloud Computing based on the Hadoop platform", Computational and Information Sciences (ICCIS), 2011 International Conference on, pp.181-184, 21 2011-Oct.
[8] Guanghui Xu, Feng Xu, Hongxu Ma, 2012, "Deploying and Researching Hadoop in Virtual Machines", Automation and Logistics (ICAL), 2012 IEEE International Conference on, pp.395-399, 15 Aug. 2012.
[9] Diatchenko L, Lau Y F, Campbell A P, Chenchik A, Moqadam F, Huang B, Lukyanov S, Lukyanov K, Gurskaya N, Sverdlov E D, Siebert P D. Suppression subtractive hybridization: a method for generating differentially regulated or tissue specific DNA probes and libraries. Proc Natl Acad Sci USA, pp. 6025–6030, 1996.
[10] Nutch-Apache Nutch, 2012; Available from:http://nutch.apache.org/.
[11] Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung, 2003, "The Google File System" ,pp.1-15 ,October 2003.
D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," the Journal of machine Learning research, vol. 3, pp. 993–1022, 2003.
D. M. Blei and J. D. Lafferty, "Dynamic topic models," in ICML, 2006

