International Journal of Scientific & Engineering Research Volume 9, Issue 4, April-2018
ISSN 2229-5518

An Efficient Clustering Using Hadoop Mapreduce for Mobile Clouds and Survey

[1]S.Amutha, [2]N.V.Chinnasamy

Abstract— the new generations of mobile devices have high processing power and storage, but they lag behind in terms of software systems for big data storage and processing. Hadoop is a scalable platform that provides distributed storage and computational capabilities on clusters of commodity hardware. Building Hadoop on a mobile network enables the devices Torun data intensive computing applications without direct knowledge of underlying distributed systems complexities. As mobile devices are more susceptible to unauthorized access, when compared to traditional servers, security is also a concern for sensitive data. The MDFS (Mobile Distributed File System) addresses these issues for big data processing in mobile clouds. The Hadoop Map Reduce framework over MDFS and have studied its performance by varying input workloads in real heterogeneous mobile cluster. The implementation addresses all constraints in processing large amounts of data in mobile clouds.

Keywords—Mobile Computing, Hadoop Map Reduce, Cloud Computing, Mobile Cloud, Energy-Efficient Computing, Fault-Tolerant Computing.

1 INTRODUCTION

In technology, mobile devices are slowly replacing traditional personal computers. The new generations of mobile devices are powerful with gigabytes of memory and multi-core processors. These devices have high-end computing hardware and complex software applications that generate large amounts of data on the order of hundreds of megabytes. This data can range from application raw data to images, audio, video or text files. Big data processing on mobile devices has become a key emerging necessity for providing capabilities similar to those provided by traditional servers.

2 BIG DATA

Big data is a collection of large datasets that cannot be processed using traditional computing techniques. Processing large-scale datasets has become an increasingly important and challenging problem as the amount of data created by online social networks, healthcare industry, and scientific research. This has certain challenges, it is very difficult to transfer such a huge data to the computing node and very high speed networks will be needed.

The era of big data requires new ways to store, manage, access and process the colossal amount of available data. Parallel processing is scalable and can gain performance improvement by several orders of magnitude, it is generally accepted as a must for data intensive applications in the big data era.

3 HADOOP

Hadoop is an Apache open source framework that allows distributed processing of large datasets across clusters of computers using simple programming models. A Hadoop frame worked application works in an environment that provides distributed storage and computation across clusters of computers. The core of Hadoop consists of a storage part, Hadoop Distributed File System (HDFS) and a processing part called Map Reduce.

The Hadoop File System (HDFS) has master/slave architecture. The master process (called as Name Node) manages the global name space and controls the operations on files. Each slave process (called as Data Node) stores the files in form of data blocks and performs operations as instructed by the Name Node. The Name Node manages the data replication and placement for fault tolerance, performance and reliability. Job Tracker is the Map Reduce master daemon that accepts the user jobs and splits them into multiple tasks. It then assigns these tasks to Map Reduce slave nodes in the cluster called the Task Trackers. Task Trackers are the processing nodes in the cluster that run the tasks- Map and Reduce. The Job Tracker is responsible for scheduling tasks on the Task Trackers and re-executing the failed tasks. Task Trackers report to Job Tracker at regular intervals through heartbeat messages which carry the information regarding the status of running tasks and the number of available slots.

- [1]*S.Amutha,Research Scholar in M.Phil Sri Vijay Vidyalaya College of Arts and Science Dharmapuri, E-mail:amuthasekar0101@gmail.com*
- [2]*N.V.Chinnasamy, Assistant Professor, Department of Computer Science, Sri Vijay Vidyalaya College of Arts and Science, Dharmapuri, E-mail:chinnasamy.vk@gmail.com*

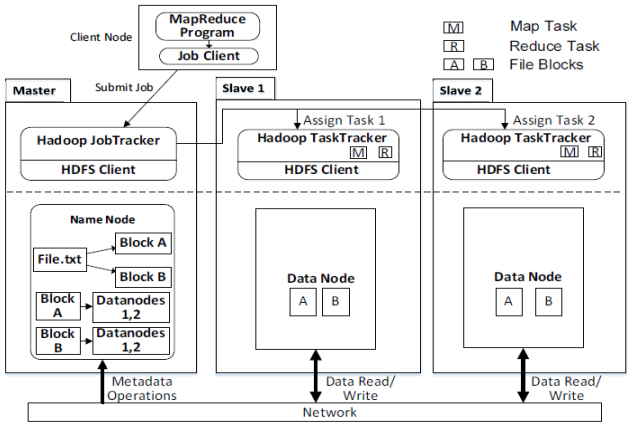

Fig.1.1Hadoop Architecture

4 MAP REDUCE

Map Reduce is a simple yet powerful programming model designed for processing large scale datasets in a distributed and parallel fashion. Originally developed by Google, and later on popularized by its open source implementation.

Hadoop Map Reduce has been widely used in practice by companies including Google, Yahoo!, Face book, Amazon, and IBM. A data processing request under the Map Reduce framework, called a job, consists of two types of tasks: map and reduce. A map takes a set of data and processes it to produce intermediate results (key value pairs). Then, reduce tasks fetch the intermediate results and carry out further computations to produce the final result. Map and reduce tasks are assigned to the machines in the computing cluster by a master node that keeps track of the status of these tasks to manage the computation process. The major advantage of Map Reduce is that it is easy to scale data processing over multiple computing nodes.

5 MDFS [MOBILE DISTRIBUTED FILE SHARING]

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.

As every file is encrypted using a secret key and partitioned into n1 file fragments using erasure encoding (Reed Solomon algorithm). Unlike conventional schemes, the secret key is not stored locally. The key is split into n2 fragments using Shamir's secret key sharing algorithm. File creation is complete when all the key and file fragments are distributed across the cluster. For file retrieval, a node has to retrieve at least k1 (<n1) file fragments and K2 (<n2) key fragments to reconstruct the original file.

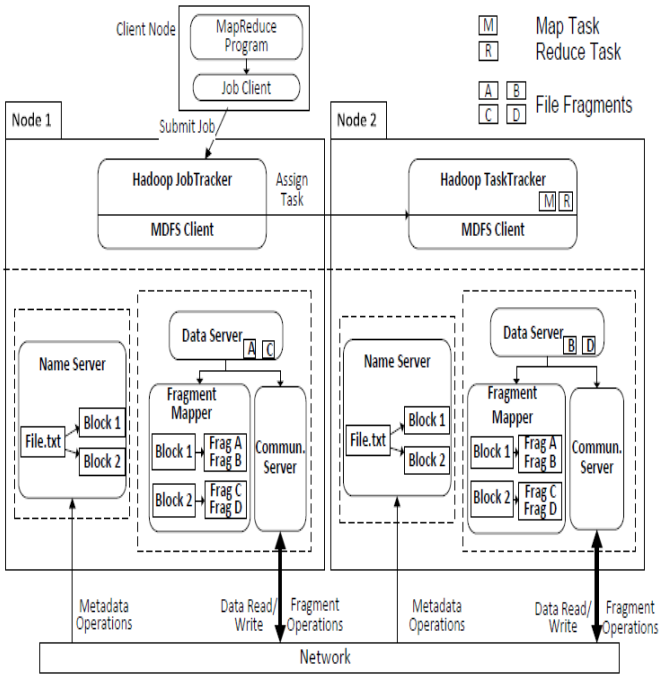

Fig.1.2 Distributed Architecture of MDFS

6 RELATED WORKS

There have been several research studies that attempted to bring Map Reduce framework to the heterogeneous cluster of devices, due to its simplicity and powerful abstractions. The Hadoop based platform Hyrax for cloud computing on smart phones. 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. P2P-MapReduce describes a prototype implementation of a Ma Reduce framework which uses a peer-to-peer model for parallel data processing in dynamic cloud topologies. It describes mechanisms for managing node and job failures in a decentralized manner. The previous research focused only on the parallel processing of tasks on mobile devices using the Map Reduce framework without addressing the real challenges that occur when these devices are deployed in the mobile environment. Proposed a k- Resilient Mobile

Distributed File System (MDFS) for mobile devices targeted primarily for military operations. Chen et al. proposed a new resource allocation scheme based on k-out-of-n framework and Implemented a more reliable and energy efficient Mobile Distributed File System for Mobile Ad Hoc Networks (MANETs) with significant improvements in energy consumption over the traditional MDFS architecture. In the remaining part of this section we present background material on Hadoop and MDFS.

7 Hadoop Map reduce For Mobile Clouds Implementation

7.1 System Implementation

In order to switch from HDFS to MDFS, the hadoop user only needs to add the location of jar file to the HADOOP_CLASSPATH variable and change the file system scheme to ′mdfs′. In the present implementation the fragment Mapped is started in the same node as the name server. Since no changes are required in the existing code base for MDFS integration, the user can upgrade to a different Hadoop release without any conflict.

7.2 Existing System

Mobile applications that perform massive computing tasks (big data processing) offload data and tasks to data centers or powerful servers in the cloud. There are several cloud services that offer computing infrastructure to end users for processing large datasets. The previous research focused only on the parallel processing of tasks on mobile devices using the Map reduce framework without addressing the real challenge that occur these devices are deployed in the mobile environment.

7.3 Disadvantage

Traditional security mechanisms tailored for static networks are inadequate for dynamic networks. Existing ignores energy efficiency mobile devices have limited battery power and can easily fail due to energy depletion. HDFS needs better reliability schemes for data in the mobile environment.

7.4 Proposed System

In this paper implement Hadoop Map reduce framework over MDFS and evaluate its performance on a general heterogeneous cluster of devices. Implement the generic file system interface of hadoop for MDFS system interoperable with other hadoop frameworks like HBase. There are no changes required for existing HDFS applications to be deployed over MDFS.

8 Performance Evaluation

We present formance results and identify bottlencks involved in processing large input datasets. The performance of MDFS on mobile devices. We ran Hadoop benchmarks on a heterogeneous 10 node mobile wirless cluster consisting of 1 personal desktop computer, 10 netbooks and 4G smartphones running Androp2.3 OS.we considered thefollowing parameters : 1.Size of input dataset 2.Bock size and 3.Cluster Size.

8.1 Effect of Bolck size on job Completion Time:

we found that the optimal value of Block size for a 50MB dataset is 4MB. The performance degrades when the block size is reduced or increeased our work. A larger block size will reduce the number of blocks and thereby limit the amount of possible parallelism in the cluster.

8.2 Effect Of Cluster Size On Job Completion Time

The cluster size determines the level of possible paralelization in the cluster.The cluster size increases more tasks can be run in parallel, thus reducing the job completion time.For larger files there are several map tasks that can be operated in parallel depending on the configured block size.For smaller files, the performance is not affected much by the cluster size as the perferance gain obtained as part of parallelism is comparable to the additional cost incurred in the task setup.

8.3 Effect Of Input Data Size

The effect of input dataset size on MDFS throughout experiment measures the average read and write throughout for different file sizes.the block size is set to 4 MB.Maximum throughput is observed for file sizes that are multiples of block size. The input dataset size is increased from 1MB to 4MB becausemore data can be transferred in a single block read/write request.

8.4 Centralized versus Distributed Architecture

The synchronization the distributed solution need to broadcast dirtectory information after a files is created or updated.A mapreduce job has more read operations distributed architecture might perform better as all metadata information can be queried locally rather than contacting the centralized server.Data reliablity is guaranted by k-out-of-n framework in both architecture.

8.5 Energy-aware task scheduling v.s. Random task scheduling

The default Map-Reduce task scheduling in a moblie as-hoc network is essentially a random task allocation. In both Teragen and Terasort experiments,Our scheduling algorithm effectively reduces the job completion time by more than 100%.Lower job completion time indicates lower data retrieval energy of each tasktracker.

9 Conclusion

There is a huge amount of data lying in the industry so Hadoop has gained lot of importance. Hadoop can implement on low cost hardware and can be used by many people on large number of dataset. Map Reduce is the most important component in Hadoop. The Hadoop Map Reduce framework over MDFS demonstrates the capabilities of mobile devices to capitalize on the steady growth of big data in the mobile environment. System addresses all the constraints of data processing in mobile cloud - energy efficiency, data reliability and security. The evaluation results show that our system is capable for big data analytics of unstructured data like media files, text and sensor data.

REFERENCES

[1] S. Huchton, G. Xie, and R. Beverly, "Building and evaluating a k-resilient mobile distributed file system resistant to device compromise," in Proc. MILCOM, 2011.
[2] G. Huerta-Canepa and D. Lee, "A virtual cloud computing provider for mobile devices," in Proc. of MobiSys, 2010.
[3] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" Computer, 2010.
[4]"Apachehadoop,"http://hadoop.apache.org/.
[5] J.-P. Hubaux, L. Butty´an, and S. Capkun, "The quest for security in mobile ad hoc networks," in Proc. of MobiHoc, 2001.