IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

Hadoop-GPU Based K-Means for Data Clustering

Parag Pachouri[1], Prof. Manaswini Panigrahi[2]

Department of Computer Science and Engineering, IES Institute of Technology and Management, Bhopal, India[1, 2]

Abstract: For achieving data parallelism in Apache Hadoop, MapReduce is the most prominent programming model. Lots of efforts are attempting for boost the computational speed of MapReduce in Hadoop framework. In this paper, we present a MapReduce programming model focused on the Kmeans clustering algorithms that leverage the acceleration potential of the integrated GPU in a multi-node cluster environment. It accelerated the framework by providing intra parallelism between the MapReduce function by using modified k-means algorithm. Based on various experiments on multi node cluster and depth analysis, we find that utilizing of the integrated GPU via OpenCL offers significant performance and power efficiency gains over the original CPU based or sequential approaches.

Keywords: Hadoop, Map/Reduce, OpenCL and KMeans.

I. INTRODUCTION

Hadoop is an open framework overseen by the Apache Software Foundation for storing and processing huge data set with use of cluster commodity hardware. The core of Hadoop consist of two parts, the one is storage part known as Hadoop Distributed File System and another is processing part known as MapReduce. Hadoop splits the file into blocks and distributes amongst the nodes in a cluster. The Hadoop framework composed by the following modules as discussed below

- Hadoop Common: This module consist of libraries, utilities etc which are used for other Hadoop framework modules.
- Hadoop Yarn: Hadoop Yarn is resource management platform which is responsible for managing resources in the cluster.
- Hadoop Distributed File System: Hadoop Distributed File System: It is specially designed file system for storing and processing data set with the use of cluster of commodity hardware and with fixed streaming access pattern. In Hdfs write once read any number of times but doesn't change the content of the file once it keeps into the Hdfs. The storage location of the file is also fixed. Hdfs is Master/slave architecture. Hdfs cluster consists of a single Name node, which is known as master node or server node that manages the Meta data and regulates access to files by clients. In HDFS there are a number of Data Nodes usually one per node in a cluster, which manage storage attached to the nodes that they run on. HDFS exposes the file system namespace and allows user data to be stored files in it. A file is split into one or more blocks and set of blocks are stored in Data Nodes which provides read, write requests, and performs block creation, deletion, and replication upon instruction from Name node in HDFS.

HDFS Daemons are
Name Node: Namenode is the master node in Hdfs for maintain and manage the blocks present on the Data node. It also controls access of fields by client. The main function of Name node is to maintain and execute the file System namespace. If any modification in the file it tracked by Name node. It keeps a record of how the filed in hdfs are divided into blocks and in which node can store or execute which block. It maps a file name to a set of blocks and blocks with Data node. It also records the metadata of all the files which are stored in hdfs.

Secondary Node: Secondary node are the substitute of Name node. As we known Name node store vital information such as metadata of the blocks stored in an hdfs. This data is not only stored in the main memory but also stored in the disk. A secondary node constantly reads all the file system and metadata from the main memory (RAM) of the name node and writes into the hard disk.
Job Tracker: Job Tracker receives the requests from the client. Job Tracker talks with Name node to determine the location of the data. Job Tracker finds the Task Tracker node for the execution of task based on the locality of the data.

Data Node: Data node are the slaves' node in hdfs architecture. Data nodes are a block server that stores the data locally. Data node performs low level read write requests from the file system. Data nodes are responsible for creating blocks, deleting blocks and replicating blocks. Data node regularly reports the blocks present in the cluster to the name node.

Task Tracker: Task Tracker runs on Data node.Mapper and reducer tasks are executed on the Data node administered by the task tracker. Task Tracker assign the task of the mapper and reducer.

IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

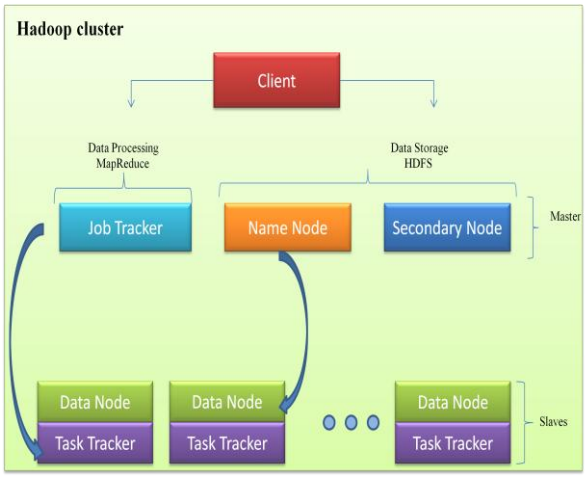

Figure 1 Hadoop Cluster

MapReduce: The programming model used in Hadoop is MapReduce. MapReduce is the basic data processing method used in Hadoop which includes breaking the entire task into two parts, known as mappers and reducers.

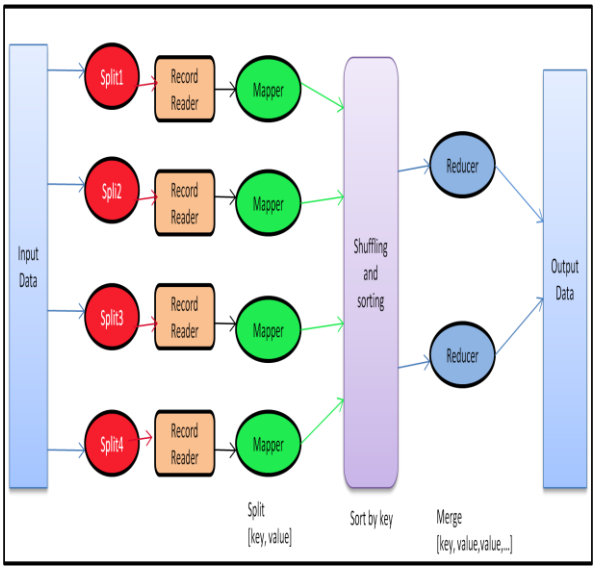

Figure 2 MapReduce process

II. HADOOP-GPU BASED K-MEANS CLUSTERING ALGORITHM

Our method is used integrated GPUs for accelerating MapReduce workloads, modifies Map or Reduce tasks algorithm in such a manner we achieve intra-node parallelism between the mapper of the each data nodes, which support GPU acceleration also. When we analyzing some workloads, it was identified that usually Mapper tasks consist of the largest portion of processing time, while Reducer tasks perform more IO processing, such as writing results into HDFS.We also modified K-means algorithm in our experiment. In a Mapper, a map function is executed on each key/value pair which required a large number of map calls for processing all data on each Data Node, Thus, some overhead from each mapper and intercommunication between JobTracker and TaskTracker is inevitable. To reduce the mapper overhead as well as prepare batched data for OpenCL kernels, we utilize a data caching strategy to support GPU acceleration in the Mapper. We must try to reduce the amount of unnecessary data copying in order to efficiently transfer data to and from the GPU. It is also imperative to pass data to the GPU in large enough chunks so that the GPUs' data parallelism can be exploited.

III. EXPERIMENT AND RESULT

We experiment on multi-nodes cluster thus to execute maximum n number of block in it. In which inter- node parallelism is occurred but we try to achieve intra-node parallelism in mapper by using GPU via Opencl with Hadoop.We evaluate performance gains in time in mm, we measure total processing time of our CPU version and OpenCL GPU version, with different number of data nodes in a cluster. Experiment on 8 data node using table 1 data and evaluating result shown in Figure 1

Table 1 On 8 Data nodes

ON 8 DATANODES		
DATA in GB	KMEANS	PARALLEL KMEANS
2	2917	1195
5	8497	6218
10	18942	11798

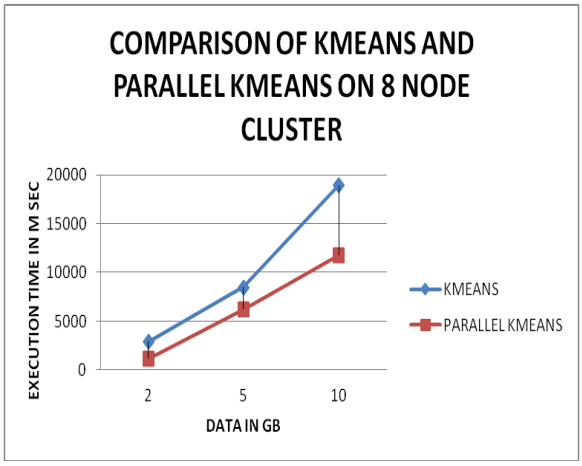

Figure 1 Comparison of K-means and Parallel K-means

Again we perform experiment on 6 data node using table 2 data and evaluating result shown in Figure 2.

Table 2 On 6 Data node

ON 6 DATANODES		
DATA	KMEANS	PARALLEL KMEANS
2	3421	2249
5	10764	7142
10	23974	14134

DOI 10.17148/IJARCCE.2016.57103

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 5, Issue 7, July 2016

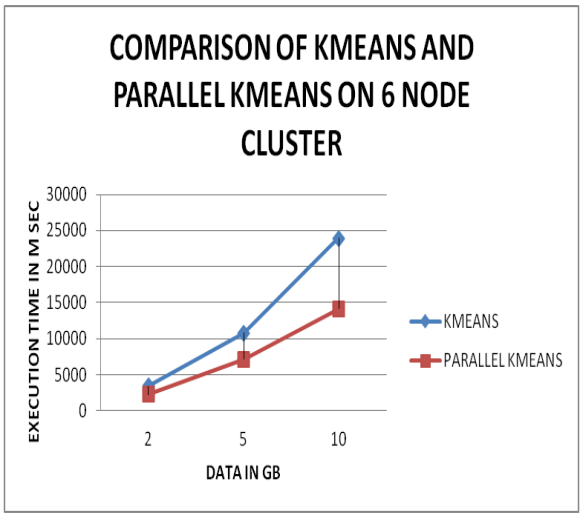

Figure 2 Comparison of K-means and Parallel K-means

Experiment performs on 4 data node on table 3 data and evaluating result shown in Figure 3.

Table 3 On 4 Data node

ON 4 DATANODES		
DATA	KMEANS	PARALLEL KMEANS
2	3971	2807
5	12970	9794
10	29716	17087

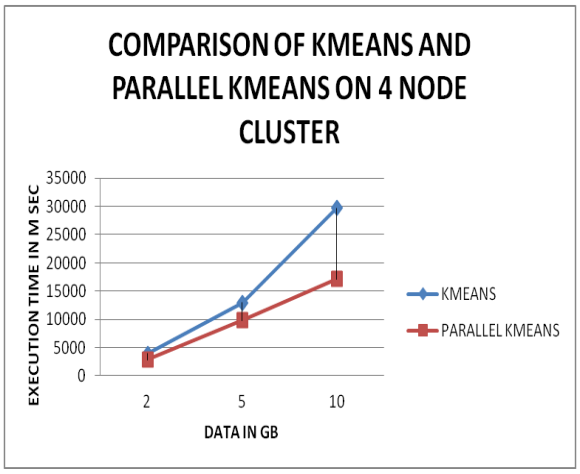

Figure 3 Comparison of K-means and Parallel K-means

Experiment on 2 data node on table 4 data abd result shown in figure 4.

Table 4 On 2 Data node

ON 2 DATANODES		
DATA	KMEANS	PARALLEL KMEANS
2	4679	3971
5	17804	13790
10	39764	21047

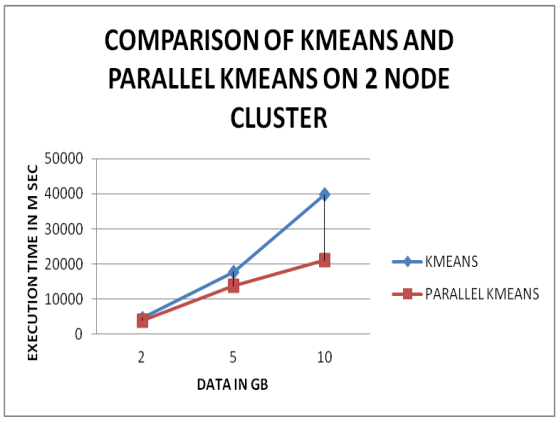

Figure 4 Comparison of K-means and Parallel K-means

V. CONCLUSION

Lots of data has been involved in processing these days. To solve big data problem Hadoop provide a solution in term of Map/Reduce. But the main objective of map/reduce is to make processing of big data possible. Map/reduce runs on all the data nodes present in the cluster of Hadoop thus provides inter node parallelism. To improve performance of map/reduce it is integrated with OpenCL to provide intra node parallelism. In this paper K-Means algorithm is implemented on map/reduce and accelerated map/reduce using OpenCL. It is observed that a parallel K-means out performs K-Means on Hadoop for all datasets.

REFERENCES

[1]. Sung Ye Kim,Jeremy Bottleson,et al, "Power Efficient MapReduce Workload Accerlation using Intergated-GPU", IEEE First International Conference on Big Data Computing Service and Application,2015,pp 162-169.
[2]. Himanshu Nayak, Rajesh Tiwari, et al, "A Study of GPU and CUDA for SIMD Concept" International Journal of Emerging Research in Management &Technology, 2015, volume-4.pp 2278-9359
[3]. Vincent Boyer, Didier El Baz , "Recent Advances on GPU Computing in Operations Research", IEEE 27th International Symposium on Parallel & Distributed Processing Workshops and PhD Forum,2013.
[4]. Jayshree Ghorpade, Jitendra Parande,et al. "GPGPU PROCESSING IN CUDA ARCHITECTURE", Advanced Computing: An International Journal (ACIJ), January 2012, vol.3, No.1.
[5]. Stephen W. Keckler , William J. Dally,et al, "GPUS and the Future of Parallel Computing", IEEE Computer Society,2011
[6]. John D. Owens, Mike Houston, David Lucbke, et al., "GPU Computing Proceedings", IEEE, 2008, 96(5): 879-899.
[7] Chris McClanahan, "History and Evolution of GPU Architecture", IEEE, 2010
[8] Stephen W. Keckler, William J. Dally,et al, "GPUS and the Future of Parallel Computing", IEEE Computer Society,2011
[9] Paramjeet kaur et al, "A Survey on CUDA", (IJCSIT) International Journal of Computer Science and Information Technologies, 2014, Vol. 5 (2) ,pp 2210-2214.
[10] R. Vuduc, A. Chandramowlishwaran, J. W. Choi, M. E. Guney, and A. Shringarpure, "On the Limits of GPU Acceleration," in Proc. USENIX Wkshp. Hot Topics in Parallelism (HotPar), Berkeley, CA, USA, June2010.
[11] Volodymyr V. Kindratenko, Jeremy J. Enos, et al, "GPU Clusters for High Performance Computing", National Center for Supercomputing Apllication,University of Illinious at Urbana-Champaign,USA,