IMPLEMENTATION OF SPARK PLATFORM IN CLOUD ARCHITECTURE

K. SHANMUGA SUNDAR, B. SREE RAGHAV, K. GOBU

Computer Science and Engineering, MCA, Jaya engineering college
E-mail: shanmugas66@gmail.com, raghavnonstop95@gmail.com, kgobu.11@gmail.com

Abstract—we have enter the big data era, where massive data are generated each single day. Most of this new generated big data are images and videos, in this paper we present our design for implementation of spark platform in cloud architecture. we also report the performance scalability of spark platform.

Keywords—Spark, Cloud Computing, RDD, HDFS, Dataset.

I. INTRODUCTION

Map reduce and its varies have been highly success full in implementing large scale data intensive applications on commodity cluster. However most of these system are built around an acyclic data flow model that is not suitable for other popular applications this paper focuses on one such class of application: those that reduce a working set of data across multiple parallel operations this includes many iterative machine learning algorithms as well as interactive data analysis tool. We propose a new frame work called spark that supports these applications while retaining the scalability and fault tolerance of map reduce. spark introduces an abstraction called resilient distributed datasets (RDDs). An RDD is a read only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost. Spark can out perform hadoop by 10x in iterative machine learning jobs.

II. RELATED WORK

There are several related work in processing images in parallel using Hadoop platform. The biggest difference between our work and others is that our solution provides a PaaS and supports the multiple languages in implementing image processing algorithms. HIPI is one of them that is simile at to our work. In contrast to our work, HIPI creates an interface for combining multiple image files into a single large file in order to overcome the limitation of handling large number of small image files in Hadoop. The input type used in HIPI is referred to as a HipImageBundle (HIB). A HIB is a set of images combined into one large file along with some meta data describing the layout of the images. HIB is similar with Hadoop sequence file input format, but it is more customizable and mutable. However, users are required to modify the image storage using HIB, which creates additional overhead in programming. In our work, we make the image storage transparent to users, and there is no additional programming overhead for users to handle image storage. Hadoop Map reduce for Remote Sensing Image Analysis aims to find an efficient programming method for customized processing within the Hadoop Map Reduce framework. It also uses the whole image as Input Format for Hadoop, which is similar with our solution. However, the work only supports Java so that all mapper codes need to be written in Java. Compared with our solution, he performance is not as good as the ours since we use native C++ implementation for OpenCV. PVAMU Cloud Computing Center Parallel Image Database Processing with Map Reduce and Performance Evaluation in Pseudo Distributed Mode performs parallel distributed processing of a video database by using the computational resource in a cloud environment. It uses video database to store multiple sequential video frames, and uses Ruby as programming language for Mapper, thus runs on Hadoop with streaming mode same as ours. As a result, our platform is designed to be more flexible and supports multiple languages. HPC jobs Virtual Machine Farm to Support High Thought put Computing HPC Cluster High Speed Interconnect Large-scale Image Processing Using Map Reduce try to explore the feasibility of using Map Reduce model for doing large scale image processing. It packaged large number of image files into several hundreds of Key-Value collections, and split one huge image into smaller pieces. It uses Java Native Interface(JNI) in Mapper to call Open CV C++ algorithm. Same with the above work, this work only supports a single programming language with additional overhead from JNI to Mapper.

III. EXISTING SYSTEM

Image processing execution environment based hadoop platform provides distributed file system (HDFS) that supports large amount of data storage and access. Hadoop Map Reduce programming model supports parallel processing data based on the widely used map and reduce parallel execution pattern. In order to support the multiple language requirements in image processing domain, we choose Hadoop streaming programming model by revising standard input and output, and stream data to
applications written with different programming languages. Moreover, the streaming model is also easy to debug in a standalone model, which is critical to test and evaluate an algorithm before going to large scale. The image processing application execution environment with Map Reduce on Hadoop. On the left side, a large number of images are stored in HDFS, which are distributed across the cluster with 128MB as one block. These images are split by Hadoop Map Reduce engine with customized Input Format, and are distributed to large number of mappers that execute image processing applications to the assigned images.

The results may be merged by the reducer that exports the results to customized Output Format class to finally save the outputs. Since large amount raw data are transferred among split, mappers and reducers, it is very important to keep data locality to minimize network traffic. All mappers are launched on the node where the processed images are physically stored. The emergence of cloud data centers enhances the capability of online data storage. Since massive data is stored in data centers, it is necessary to effectively locate and access interest data in such a distributed system. However, traditional search techniques only allow users to search images over exact match keywords centralized index. These techniques cannot satisfy the requirements of content based image retrieval (CBIR). problems face are scalability challenge and hybrid parallel programing challenges of creating code for morned computer hardware configurations with multilevel parallelism. The another major challenge face is reuse and share the existing research results since these results are largely depend on OS, libraries and underlaying architecture.

IV. PROPOSED SYSTEM

Instead of the existing system which uses Hadoop platform for image processing we use Apache Spark. The Apache Spark is a fast and general purpose cluster computing system. In contrast to Hadoop's two stage disk based Map Reduce paradigm, Spark's in memory primitives provide performance up to 100 times faster for certain applications. By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well suited to machine learning algorithms. Apache Spark is a framework for performing general data analytics on distributed computing cluster like Hadoop. It provides in memory computations for increase speed and data process over map reduce. It runs on top of existing hadoop cluster and access hadoop data store (HDFS) , can also process structured data in Hive and Streaming data from HDFS, flume, Kafka, etc........

Spark enables applications in Hadoop clusters to run up to 100x faster in memory, and 10x faster even when running on disk. Spark makes it possible by reducing number of read/write to disc. It stores this intermediate processing data in memory. It uses the concept of an Resilient Distributed Dataset (RDD), which allows it to transparently store data on memory and persist it to disc only it’s needed. This helps to reduce most of the disc read and write the main time consuming factors of data processing. Spark lets you quickly write applications in Java, Scala, or Python. This helps developers to create and run their applications on their familiar programming languages.
and easy to build parallel apps. It comes with a built in set of over 80 high level operators. We can use it interactively to query data within the shell too.

From a programming abstraction point of view:
1. It generalizes two stage Map/Reduce to support an arbitrary DAG of tasks. Most computation maps (no pun intended) into many maps and reduces with dependencies among them. Spark’s RDD programming model models these dependencies as a DAG of tasks. This way, it is much more natural to express computation.
2. Through better language integration to model data flow, it does away with the huge amount of boilerplate code required in Hadoop Map Reduce. Typically when you look at a Hadoop Map Reduce program, it is difficult to extract what it attempts to do because of the huge amount of boilerplates, whereas it is much more natural to read a Spark program.
3. Due to the more flexible programming model, some operations that are essential & embedded in Hadoop Map Reduce are now in the “application space”. What this means is that applications can actually rewrite the way shuffle or aggregation is done, which is not really possible in Map Reduce. Not a lot of applications do this, but this enables certain applications to do hyper optimization for their specific workload.
4. Optionally, applications can choose to put datasets in memory across a cluster. This simple primitive is actually fundamental in many applications that need to scan some intermediate dataset multiple times (e.g. most machine learning algorithms are iterative).

From a high-level system engineering point of view:
1. Spark uses fast RPCs for task dispatching and scheduling.
2. Spark uses a thread pool for execution of tasks rather than a pool of JVM processes.

The above two combined enables Spark to schedule tasks as fast as milliseconds, whereas MR scheduling takes seconds and sometimes minutes in busy clusters.
3. Spark supports both checkpointing-based recovery (the way fault-tolerance is implemented in Hadoop MR) and lineage-based recovery. Unless fault is very common, lineage-based recovery is faster (because replicating state across the network is slow for high-throughput data flow applications).
4. Partially due to its academic root, the Spark community often embrace novel ideas. One example of this is the use of a torrent-like protocol in Spark for doing one-to-all broadcast of data

CONCLUSION

Through this paper we conclude that Apache Spark is the fast and general purpose cluster computing system, Which can outperform the hadoop platform. Spark programs can better utilize the cluster resources such as CPU, network. When you look at an map reduce program in hadoop you cannot understand what it is trying to do because of the huge amout of boilerplates, instead in spark we can easily understand the program.
The fault-tolerance ability of the cloud is improved while implementing spark, and elasticity of the cloud is also improved in great manner. The spark platform behaves as an common execution model for multiple workload.

REFERENCE


Spark: cluster computing with working sets, matei Zaharia, university of california, berkeley.

***