<u>First Paper - Hadoop Distributed File System</u>

<u>Second Paper - iShuffle: Improving Hadoop Performance with Shuffle-on-Write</u>

Group 1: Mihika Naik, Nimmi Ghetia, Pallavi Choudhary, Swati Kandari

Hadoop provides a distributed file system and a framework for the analysis and transformation of very large data sets using the MapReduce paradigm. An important characteristic of Hadoop is the partitioning of data and computation across many hosts, and executing application computations in parallel close to their data. A Hadoop cluster scales computation capacity, storage capacity and IO bandwidth by simply adding commodity servers. HDFS is the file system component of Hadoop and it stores file system metadata and application data separately. Hadoop architecture comprises Namenode, Datanode, HDFS Client. Namenode maintains namespace hierarchy and file system metadata such as block locations. Namespace and metadata is stored in RAM but periodically flushed to disk. Modification log keeps on-disk image up to date. Datanodes store the HDFS file data in local file system and receives command from namenode that instructs to replicate blocks to other nodes, re-register or shutdown, remove local block replicas, and send immediate block report. HDFS client is a code library that exports HDFS file system interface to applications. It reads data by transferring data from a DataNode directly and Writes data by setting up a node-to-node pipeline and sends data to the first DataNode. The NameNode in HDFS, in addition to its primary role serving client requests, can alternatively execute either of two other roles, either a *CheckpointNode* or a *BackupNode*. The role is specified at the node startup.

Hadoop is a popular implementation of the MapReduce framework for running data-intensive jobs on clusters of commodity servers. Shuffle, the all-to-all input data fetching phase between the map and reduce phase can significantly affect job performance. However, the shuffle phase and reduce phase are coupled together in Hadoop and the shuffle can only be performed by running the reduce tasks. This leaves the potential parallelism between multiple waves of map and reduce unexploited and resource wastage in multi-tenant Hadoop clusters, which significantly delays the completion of jobs in a multi-tenant Hadoop cluster. More importantly, Hadoop lacks the ability to schedule task efficiently and mitigate the data distribution skew among reduce tasks, which leads to further degradation of job performance. In this work, we propose to decouple shuffle from reduce tasks and convert it into a platform service provided by Hadoop. In the second paper, the authors have presented iShuffle, a user-transparent shuffle service that pro-actively pushes map output data to nodes via a novel shuffle-on-write operation and flexibly schedules reduce tasks considering workload balance. Experimental results with representative workloads and Facebook workload trace show that iShuffle reduces job completion time by as much as 29.6 and 34 percent in single-user and multi-user clusters, respectively.

In MapReduce, Map takes a set of data and converts it into another set of data, where individual elements are broken down into key-value pairs and Reduce takes the output from the map as input and process further and MapReduce requires a lot of time to perform these tasks thereby increasing latency. Apache Spark is yet another batch system but it is relatively faster since it caches much of the input data on memory by RDD(Resilient Distributed Dataset) and keeps intermediate data in memory itself. Flink's data streaming achieves low latency and high throughput. Hadoop does not have any type of abstraction so MapReduce developers need to hand code for each and every operation which makes it very difficult to work. Hadoop is not efficient for caching. In Hadoop, MapReduce cannot cache the intermediate data in memory for a further requirement which diminishes the performance of Hadoop.
Map Overhead : Shuffle-on-Write technique moves the shuffle operation to the map phase. Decoupling shuffle and reduce, and using Shuffle-on-Write technique essentially moves the shuffle operation to the map phase. It is expected to have some performance impact on map tasks. However, the shuffler is mainly performing I/O operations, and should not interfere with map tasks substantially. iShuffle does not support handling of Stragglers. Speculative execution is introduced in the original design of MapReduce to address stragglers, which are usually due to faulty hardware, misconfigurations, and resource contentions. Running a backup task on a different node is likely to complete the task faster, thereby mitigating the overall job slowdown due to these stragglers. Since there may be multiple copies of the same tasks running simultaneously, Hadoop keeps the output of each task separately. As iShuffle shuffles and merges map output as soon as it is generated, we need sophisticated output deduplication to support speculative execution. Currently, iShuffle does not support speculative execution.Too many changes in hadoop framework. This largely rewrites Hadoop Fair Scheduler(HFS) by adding multiple independent modules. No performance improvement in the Map phase. There are many existing studies focusing on improving the performance of map tasks.

REFERENCES

[1] Yanfei Guo, Jia Rao, Xiaobo Zhou. iShuffle: Improving Hadoop Performance with Shuffle on-Write. IEEE transactions on parallel and distributed systems. 2017.

[2] Shvachko, Konstantin and Kuang, Hairong and Radia, Sanjay and Chansler, Robert and others. The hadoop distributed file system. 2010.