3. Maximising MapReduce Process Using a Hierarchical Approach

Zhiwei Xiao et al found out that Hadoop had limitations in exploiting data locality and task parallelism for multi-core platforms. Then they extended Hadoop with a hierarchical MapReduce scheme. An in-memory cache scheme is also seamlessly integrated to cache data that is likely to be accessed in memory. They proposed Azwraith, a hierarchical MapReduce approach aiming to maximize data locality and task parallelism of MapReduce applications on Hadoop. They discovered that there are multiple levels of data locality and parallelism in typical multicore clusters that could affect performance [7].

It has been identified that, the open-source implementation of MapReduce, Hadoop [7], makes use of the JVM runtime to run the actual MapReduce tasks, which is not the best way to explore the cache hierarchy and task parallelism existing in many multi-core based commodity clusters. Hadoop requires both key and value objects to implement the Hadoop Writable interface to support serialization and deserialization, causing extra objects creation and destroy overhead as well as memory footprint.

There are also some applications that require processing the same piece of data several times or iteratively to get the final results. Although Hadoop exploits data locality with a single iteration of jobs by shifting computation to its data as much as possible, unfortunately, it does not consider data locality across multiple processing iterations, and thus requires the same data being loaded multiple times from the networking file systems to nodes that process the data. As a result of all these shortcomings Zhiwei Xiao et al designed Azwraith to counter them.

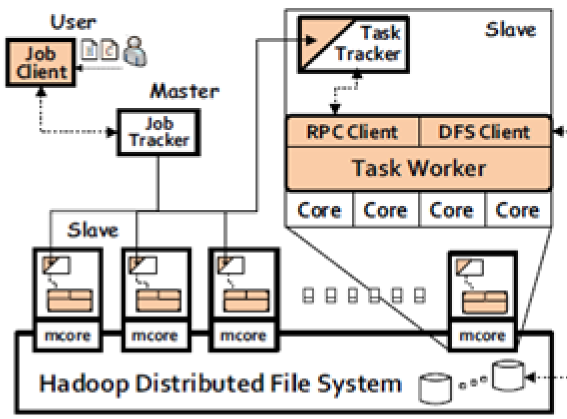

Figure 4: Azwraith Architecture

Azwraith, is a hierarchical MapReduce approach aiming to maximize data locality and task parallelism of MapReduce applications on Hadoop[4]. In the hierarchical MapReduce model of Azwraith, each Map or Reduce task assigned to a single node is treated as a separate MapReduce job and is further decomposed into a Map and a Reduce task, which are processed by a MapReduce runtime specially optimized on a single node. Specifically, Azwraith integrates an efficient MapReduce runtime for multi-core to Hadoop.

To exploit data locality among nodes at networking level, Azwraith integrates an in-memory cache system that caches data in memory that will likely be reused again, to avoid unnecessary networking and disk traffics. Through the use of word count, gigasort algorithm and linear regression their experiments proved that Azwraith, their extension to Hadoop outperformed Hadoop.

4. Accountable MapReduce in Cloud Computing

Zhifeng Xiao et al proposed Accountable MapReduce, which forces each machine to be held responsible for its behavior. They set up a group of auditors to perform an Accountability Test (A-test) which will check all working machines and detect malicious nodes in real time. They tapped into a very sensitive area which is very much of great importance in improving the overall performance of MapReduce[8].

They introduced a component known as the **Auditor Group (AG)** which carries out Accountability Test to detect malicious nodes. Normally, as shown in Figure 5, cloud resource will be divided into multiple slices, each of which is rented by a customer. A slice is a group of working machines assigned to a customer. An AG manager is maintained for the entire cloud, and one AG for each slice that runs MapReduce. The reason of associating each slice with one AG is to conserve the privacy and independence of customers.

The AG Manager is a coordinator that conducts AG creation, management, and disposal. After the AG manager becomes aware of the customer's data size, timing, and other requirements, it will determine the AG size and then create an AG for the slice. Each AG is internally structured as a cluster. The head node is the Group Head (GH), and the member node is the Group Member (GM). The GH picks up workers as test targets randomly. The master has a protocol with the GH to provide all information needed for an A-test. The GH assigns A-test tasks to the available GMs, which are the actual machines that accomplish the tasks and report their status.

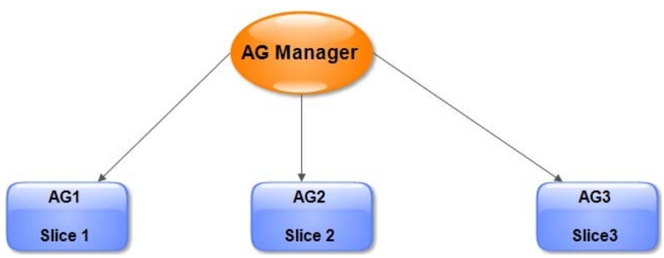

Figure 5: Audit Group in Cloud Platform

Accountable MapReduce comes in with a twofold form of benefit. Unauthorized tasks are not given a chance to execute and this will automatically improve the performance of MapReduce in the cloud. However, the drawback in this method is that sometimes false positives can distort everything. This is also introduced as an extra independent component which will be added on top of the inbuilt MapReduce such that the original set up of MapReduce is not altered as it may create some new problems [8].

5. Data aware caching for BigData applications using MapReduce

An observation regarding Hadoop and NoSQL database applications is that they generate and store a large amount of intermediate data [1], and this abundant information is thrown away after the processing finishes. Motivated by this observation, Yaxiong Zhao et al proposed a data-aware cache framework for big-data applications, which they called Dache. In Dache, tasks submit their intermediate results to the cache manager. A task, before initiating its execution, queries the cache manager for potential matched processing results, which could accelerate its execution or even completely saves the execution. A novel cache description scheme and a cache request and reply protocols are designed. They implemented Dache by extending the relevant components of the Hadoop project [9]. Testbed experiment results demonstrated that Dache significantly improves the completion time of MapReduce jobs and saves a significant chunk of CPU execution time.

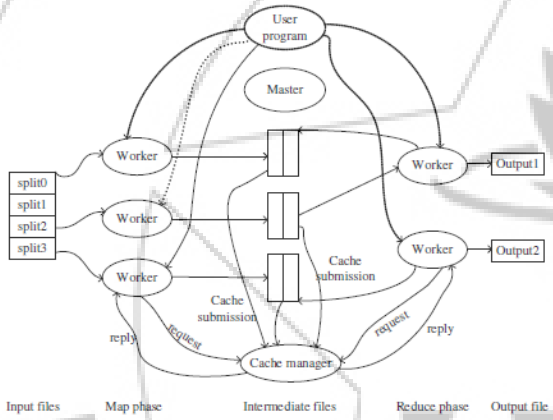

Figure 6: High level description of the architecture of Dache

Dache requires only a slight modification in the input format and task management of the MapReduce framework, and applications need only slight changes in order to utilize Dache[9]. They implemented Dache in Hadoop. Testbed experiments showed that it can eliminate all the duplicate tasks in incremental MapReduce jobs and doesn't require substantial changes to the application code hence improving the overall performance of Hadoop.

Dache identifies the source input from which a cache item is obtained, and the operations applied on the input. In the reduce phase, they devised a mechanism to take into consideration the partition operations applied on the output in the map phase. They also discovered a method for reducers to utilize the cached results in the map phase to accelerate their execution. They implemented Dache in the Hadoop project by extending the relevant components. Their implementation follows a non-intrusive approach, so it only requires minimum changes to the application code.

6. Shared Disk BigData Analytics with Apache Hadoop

For organizations which don't need a horizontal, internet order scalability in their analytics platform, Big Data analytics can be built on top of a traditional POSIX Cluster File Systems employing a shared storage model. Anirban Mukherjee et al in their study compared a widely used clustered file system: VERITAS Cluster File System (SF-CFS) with Hadoop Distributed File System (HDFS) using popular Map-reduce benchmarks like Terasort, DFS-IO and Gridmix on top of Apache Hadoop[10]. In their experiments, VxCFS could not only match the performance of HDFS, but also outperformed in many cases. This way, enterprises can fulfill their Big Data analytics need with a traditional and existing shared storage model without migrating to a different storage model in their data centers. This also includes other benefits like stability & robustness, a rich set of features and compatibility with traditional analytics applications.

They gathered a credible reasoning behind the need of a new non-POSIX storage stack for Big Data analytics and advocate, based on evaluation and analysis that such a platform can be built on traditional POSIX based cluster file systems. They developed a file system connector module for SF-CFS to make it work inside Apache Hadoop platform as the backend file system replacing HDFS altogether and also have taken advantage of SF-CFS's potential by implementing the native interfaces from this module. This scheme did not require any changes in the Map Reduce applications. Just by setting a few parameters in the configuration of Apache Hadoop, the whole Big Data analytics platform can be made up and running very quickly.

The clustered file system connector module they developed for Apache Hadoop platform has a very simple architecture. It removes the HDFS functionality from the Hadoop stack and replaces it with VERITAS Clustered File System. It introduces SF-CFS to the Hadoop class by implementing the APIs which are used for communication between Map/Reduce Framework and the File System. This could be achieved because the Map-Reduce framework always talks in terms of a well-defined FileSystem [10] API for each data access. The FileSystem API is an abstract class which the file serving technology underneath Hadoop must implement. Both HDFS and their clustered file system connector module implement this FileSystem class, as shown in Figure 7.

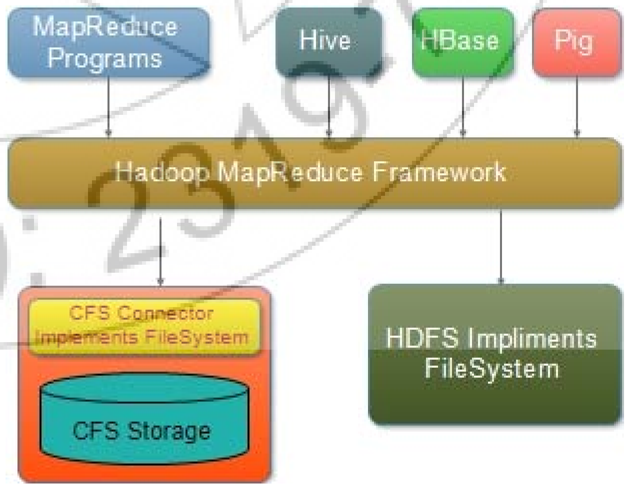

Figure 7: Architecture of SC-CFS Hadoop Connector

7. Conclusion and Future work

In this paper we have managed to combine several methods discovered by several researches in solving the problem of performance in MapReduce. In-memory cache is beginning to become a very exciting application in solving many technological problems including MapReduce. We see it as an open and available area for more future researches.

References

[1] Kudakwashe Zvarevashe, " A Survey of the Security Use Cases in Big Data", International Journal of Innovative Research in Computer and Communication Engineering, Vol. 2, Issue 5, May 2014
[2] https://www.usenix.org/legacy/publications/login/2011-10/openpdfs/Burd.pdf
[3] GoogleBigTable:http://labs.google.com/papers/bigtable.html
[4] Kudakwashe Zvarevashe, "A Random Walk through the Dark Side of NoSQL Databases in Big Data Analytics", International Journal of Science and Research (IJSR).
[5] http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
[6] Jing Zhang,"A Distributed Cache for Hadoop File Distribution System in Real Time Cloud Services", 2012 ACM/IEEE 13th International Conference on Grid Computing.
[7] Zhiwei Xiao, " A Hierarchical Approach to Maximizing MapReduce Efficiency" 2011 International Conference on Parallel Architectures and Compilation Techniques
[8] Zhifeng Xiao, “Accountable MapReduce in Cloud Computing", 2011 IEEE.
[9] Yaxiong Zhao," Dache: A Data Aware Caching for Big-Data Applications Using The MapReduce Framework" 2013 Proceedings IEEE INFOCOM
[10] Anirban Mukherjee ,"Shared Disk Big Data Analytics with Apache Hadoop" 2012 IEEE

Author Profile

Kudakwashe Zvarevashe, Attained his BSc degree in Information Systems at MSU, Zimbabwe in 2010. He is currently doing M Tech IT final year at JNTUH, India. He is a HIT staff development research fellow. His research interests are in the area of big data, information security, cloud computing and web services.

Dr. A. Vinay Babu, Professor in CSE and Principal of JNTUH College of Engineering, JNT University Hyderabad. His work of experience is 27 years in Teaching, Research and Administrative as an eminent educationist. He secured the Best State Teacher Award , State Govt. of Andhra Pradesh in 2011,Best Computer Science Teacher Award by ISTE, AP Chapter in Nov 2009, Eminent Educationist, awarded by National & Inter compendium, New Delhi, India April 2009,â€œEminent Citizen of India, awarded by National & International Compendium New Delhi, India March 2008, Distinguished Academician awarded by Pentagram Research centre Private Limited Hyderabad, India on Jan 2008 and Jewel of India awarded by Indian solidarity council, New Delhi, India. He guided 9 PhDs and many more PhD Scholars are working under his Guidance. He had held number of administrative positions in the University including that of Director Admissions, SCDE, and HOD of CSE etc.