Adaptive Replication Management in HDFS based on similarity based prediction Techniques

Mr.Yogesh Shivaji Sapnar, Mr.Milindkumar Balchandra Vaidya
ME Student, Department of Computer Engineering, Amrutvahini COE,Maharashtra, India
Professor, Department of Computer Engineering, Amrutvahini COE,Maharashtra, India

ABSTRACT

The number of applications based on Apache Hadoop is dramatically increasing due to the robustness and dynamic features of this system. At the heart of Apache Hadoop, the Hadoop Distributed File System (HDFS) provides the reliability and high availability for computation by applying a static replication by default. However, because of the characteristics of parallel operations on the application layer, the access rate for each data file in HDFS is completely different. Consequently, maintaining the same replication mechanism for every data file leads to detrimental effects on the performance. By rigorously considering the drawbacks of the HDFS replication, this paper proposes an approach to dynamically replicate the data file based on the predictive analysis. With the help of probability theory, the utilization of each data file can be predicted to create a corresponding replication strategy. Eventually, the popular files can be subsequently replicated according to their own access potentials. For the remaining low potential files, an erasure code is applied to maintain the reliability. Hence, our approach simultaneously improves the availability while keeping the reliability in comparison to the default scheme. Furthermore, the complexity reduction is applied to enhance the effectiveness of the prediction when dealing with Big Data.

Keyword:- *Replication, HDFS, Proactive Prediction, Optimization, Bayesian Learning, Gaussian Process.*

INTRODUCTION

THE evolution of big data has created a phenomenon in application and solution development to extract, process and store useful information as it emerges to deal with new challenges. In this area, Apache Hadoop is one of the most renowned parallel frameworks. Not only is it used to achieve high availability, Apache Hadoop is also designed to detect and handle the failures as well as maintain the data consistency. Coming along with the development of Apache Hadoop, the Hadoop Distributed File System (HDFS) has been introduced to provide the reliability and high-throughput access for data-centric applications. Gradually, HDFS has become a suitable storage framework for parallel and distributed computing, especially for MapReduce engine, which was originally developed by Google to cope with the indexing problems on big data. To improve the reliability, HDFS is initially equipped with a mechanism that uniformly replicates three copies of every data file. This strategy is to maintain the requirements of fault tolerance. Reasonably, keeping at least three copies makes the data more reliable and more robust when tolerating the failures. However, this default replication strategy still remains a critical drawback with regards to the performance aspect. Intuitively, the purpose of inventing Apache Hadoop was to achieve better performance in data manipulation and processing [1]. Therefore, this purpose should be carefully studied at every component. In the performance perspective, based on the well-known research of delay scheduling [2], if the task is placed closer to the required data source, the system can achieves faster computation and better availability. The metric measures the distance between the task and the corresponding data source can be referred to as the data locality metric. The main reason for the improvement is twofold. First, the network overhead can be reduced on runtime due to the availability of the local data, and so no inter-communication is needed to transfer the required data from the remote nodes. Second, it is clear that the computation can start immediately on the input data which is locally available, and so no extra task scheduling effort is consumed. Consequently, it is meaningful to say that improving the data locality would immensely enhance the system performance in terms of availability and calculation time. Although there are some studies on this subject matter, very few proactive solutions are proposed that rigorously consider the nature of the job workload. Due to the fact that workload in Apache Hardtop consists of short and long tasks together, these tasks should be handled fairly to accelerate the computation. Typically, the Fair scheduler and delay scheduling algorithm [2] provide the optimal data locality when the system is filled with head-of-line jobs and short tasks. However, the long tasks, if they are present would not be treated appropriately and thus

make the system imbalanced. One solution is to pro-actively prepare the potential replications before scheduling the tasks in order to redirect and balance the computation. To do that, we aim to improve the data locality metric by changing the replication scheme adaptively with regards to the popularity of the data file. Not only is the nature of the access rate taken into account, but the replica placement is also carefully considered. Note that the access rate is defined as the number of accesses in a given unit of time. Subsequently, the data files in HDFS are replicated based on their own access potential as well as the overall status of the system. By definition, the access

2. RELATED WORK

As the technology is growing the size of data is also growing accordingly. People are living in the world of data. The term big data came into the picture due to the awareness of people towards the technology. The term big data refers to the dataset of huge size which are unable to store in typical database. These huge datasets cannot be analyzed by simple RDBMS tools. Generally the RDBMS can store and process the structured dataset but the huge amount of generated data can be structured unstructured or semi-structured [1]. Researchers are deluged with this continuously increasing amount of data processing which is storm of data is flowing in almost all science research areas like web data, biomedical, Bio-Informatics and other disciplines due to its high accuracy and capability to deal with high dimension data [2-4]. The biggest challenge in front of researchers is how to do the proper analysis of this much large scale of data so that the meaningful results can be drawn from it. To give better visualization of the large scaled data, data mining comes into the picture. Data mining is the procedure to discover the new pattern from the existing datasets [5-7]. Various data mining algorithm has been developed and implemented in practice by many researchers. But now in the era of big data there is need to develop data mining algorithms which are suitable for big data analysis. Several parallel algorithms have been developed using threads, MPI, Map Reduce and so on [5, 8]. Among all these techniques Map Reduce is practically well suited for large scale data analysis. In this paper an algorithm for Map Reduce based SVM is implemented which run on several data size files and training time has been calculated on Hardtop cluster. A major problem with SVM is to select the proper kernel parameters [9-10]. In this paper the number of support vectors has been calculated on several dataset by varying the value of penalty parameter C and RBF kernel function parameter σ. The corresponding accuracy and training time has been calculated for the same. The paper is organized as follows. Section II describes about the basic of SVM, SVM kernels, advantages and disadvantages of SVM and why there is need of parallel SVM. Section III describes the architecture of parallel SVM. Section IV describes the Hardtop framework which is mainly focused on its two core components HDFS and Map Reduce distributed programming model. Section V focuses on architecture and algorithm of Map Reduce based parallel SVM. Section VI includes the experimental results. And finally Section VII concludes with future work.

3.LITURATURE SURVEY

I will discuss the basic configurations of storage devices. These are static and do not allow the flexibility to tradeoff capacity for availability or performance. I will also discuss the tiered storage systems that use multiple configurations, or a caching configuration to provide static configurations that do not allow for a capacity tradeoff.

4.RAID

Redundant Arrays of Inexpensive Disks Redundant Arrays of Inexpensive Disks (RAID), have been around for many years [1] and are in use throughout systems today. There are three basic types of array configurations, Figure 2.2 illustrates these configurations. Others exist, but for the most part they are a mixture of one of these basic types and parity. • Striping (RAID-0) is the configuration where one places a chunk of data on individual storage devices then accesses the data as a stripe across all storage devices. This configuration will aggregate bandwidth across disks. • Mirroring (RAID-1) is the configuration where one places a copy of the data on another storage device. One can then steer accesses to a storage device that is available to serve the data. This configuration will utilize single-disk bandwidth of each disk in the configuration, effectively aggregating the disks bandwidth. • Concatenation (JBOD) is the configuration where one linearly concatenates storage devices. All devices are able to store individual files, no copies of data are made on this configuration. This configuration will aggregate bandwidth of disks based on file system block allocation which spaces files to lower fragmentation.[1]
M. Zaharia, D. Borthakur, J. Sen Sarma, [2] Hadoop's implementation of MapReduce resembles that of Google. Hadoop runs over a distributed file system called HDFS, which stores three replicas of each block like GFS. Users

submit jobs consisting of a map function and a reduce function. Hadoop breaks each job into tasks. First, map tasks process each input block (typically 64 MB) and produce intermediate results, which are key-value pairs. There is one map task per input block. Next, reduce tasks pass the list of intermediate values for each key and through the user's reduce function, producing the job's final output. Job scheduling in Hadoop is performed by a master, which manages a number of slaves. Each slave has a fixed number of map slots and reduce slots in which it can run tasks. Typically, administrators set the number of slots to one or two per core. The master assigns tasks in response to heartbeats sent by slaves every few seconds, which report the number of free map and reduce slots on the slave. Hadoop's default scheduler runs jobs in FIFO order, with five priority levels. When the scheduler receives a heartbeat indicating that a map or reduce slot is free, it scans through jobs in order of priority and submit time to find one with a task of the required type. For maps, Hadoop uses a locality optimization as in Google's MapReduce [18]: after selecting a job, the scheduler greedily picks the map task in the job with data closest to the slave (on the same node if possible, otherwise on the same rack, or finally on a remote rack).

K. S. Esmaili, L. Pamies-Juarez, and A. Datta, [3] Hadoop clusters are gaining more and more in popularity based on their ability to parallelize and complete large scale computational tasks on big data. Service offerings of this type have appeared in the recent years, covering a need for dynamic and on-demand creation of such data analytics frameworks. The aim of this paper is to provide a mechanism for offering such virtual clusters as a service, with built-in intelligence functionalities for efficient management. The target of these mechanisms is to predict future demand of the files in the HDFS cluster and dynamically manipulate the according replication factor for availability purposes, in order to improve performance and minimize storage overhead. To this end, real data have been utilized as a dataset input to the prediction framework, based on Fourier series analysis, due to the latter's ability to capture different periodicities that can influence service usage. Multiple time-step ahead prediction is performed in order to enable proactive management (e.g. suitable replication strategy). We describe the framework's architecture, necessary modifications to the client side of Apache Hadoop for data logging and the results of the applied method on two real world datasets.

G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. [4]Data replication and placement was first studied in thecontext of the ale assignment problem [2] and was shown to be a complex combinatorial optimization problem. Replica placement has received attention from diverse research areas,e.g., delivery networks, web caching, web proxy services, distribution storage system, etc.. They take into account probabilistic reliability to ensure mutual consistency of replicated data. Scarlett [4] replicates blocks based on the observed probability in the past. It computes a replication face-tor for each ale and creates budget-limited replicas distributed among the cluster, with the goal of minimizing hotspots. Replicas are also aged to reserve space for new replicas. These methods are rather static in the sense that they assume that access patterns are estimated in advance and remain unchanged, therefore a one-time replica scheme is implemented and lasts for a long time period. In such case, the cost imposed by replica adjustment is going to be amortized and can be ignored. However, the performance of these methods will deteriorate when the access pattern changes over time. In contrast, dynamic methods adapt replicas in the clusters frequently as upon every request, which are more responsive at the cost of higher system cost. One of the dynamic methods is DARE proposed in ,which uses probabilistic sampling and a competitive aging algorithm independently at each node to determine the number of replicas to allocate for each ale and the location for each replica. It tries to reduce consuming extra network and computation resources. Based on the data popularity, ERMS[6] increases the number of replicas for hot data and cleansup these extra replicas when the data cools down. The main purpose of ERMS is improving the reliability and performance of HDFS. CDRM [3] builds up a cost model to capture the relationship between availability and replication factor. The model is used to and the lower bound on the number of replicas which are placed among distributed nodes to minimize blocking probability. Dynamic methods can bring better performance to what they target at, but they also bring non-trivial extra replica adjustment cost that may deteriorate the system's overall performance. Of course if the replica adjustment cost is taken into consideration from the start of development, the overhead can be kept under control with the dynamic approach. The methods mentioned above all try to estimate a suitable access pattern for the coming future. Based on this access pat-tern, they focus on different criteria, including access latency, system availability, minimizing resource hotspots, or resource consumption . In this paper, an access pattern is estimated, too, but we will focus on reducing the internal traffic, because internal network bandwidth is a scarce resource, and for the data center studied in our research, fileserving is the main job which will be affected even more by heavy internal traffic than computing-centric data centers. There have been other papers concerning traffic reduction, for example. But in the traffic is reduced at the price of storage resource imbalance in which data are stored as much as possible at a small set of nodes in the cluster. In a real system, storage usage should be reasonably balanced to spread the load evenly across all the nodes, so each node

still has spare room for performance tuning and for sudden bursts in user demands. This is an important aspect considered in our method.

5. CONCLUSION

Conclusion Big-data analysis is emerging as an important tool and file system performance when manipulating large files is a critical aspect of performance for big-data analysis. Such analysis typically occurs over large collections of data on distributed file systems. However, such distributed file systems are overlaid on underlying single-node local file systems. In a busy distributed storage cluster, misplaced replicas will cause unnecessary internal traffic to move them to the serving nodes. In this paper, we design and implement CRMS,a centralized replication management scheme to solve the problem. The problem is rest formulated as a 0-1 IntegerPrograming problem, and a feasible placement scheme is reached by solving this IP problem based on access history collected in the cluster. In order to alleviate the overhead of adjustments and avoid over-sting, only a fraction of the replica adjustment is performed according to an heuristic ad-dustmen algorithm. CRMS batches 4 adjustment steps together to keep node storage usage in balance and stops adjustment when satisfactory internal traffic reduction is reached. From the experimental results, we can see that CRMS meets our expectation and reduces internal traffic greatly comparing tithe unadjusted situation in a real world cluster from XunleiInc.. In the future, we plan to improve CRMS by dynamically adjusting the number of replicas as well, so popular data blocks can have more replicas at the server nodes to further reduce the internal traffic

6.REFERENCES

[1] “What is apache hadoop?” https://hadoop.apache.org/, accessed: 2015-08-13.
[2] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, “Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling,” in Proceedings of the 5th European conference on Computer systems. ACM, 2010, pp. 265–278.
[3] K. S. Esmaili, L. Pamies-Juarez, and A. Datta, “The core storage primitive: Cross-object redundancy for efficient data repair & access in erasure coded storage,” arXiv preprint arXiv:1302.5192, 2013.
[4] G. Ananthanarayanan, S. Agarwal, S. Kandula, A. Greenberg, I. Stoica, D. Harlan, and E. Harris, “Scarlett: Coping with skewed content popularity in mapreduce clusters.” in Proceedings of the Sixth Conference on Computer Systems, ser. EuroSys ’11. New York, NY, USA: ACM, 2011, pp. 287–300. [Online]. Available: http://doi.acm.org/10.1145/1966445.1966472
[5] G. Kousiouris, G. Vafiadis, and T. Varvarigou, “Enabling proactive data management in virtualized hadoop clusters based on predicted data activity patterns.” in P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference
on, Oct 2013, pp. 1–8.
[6] A. Papoulis, Signal analysis. McGraw-Hill, 1977, vol. 191.