International Journal on Applications in Information and Communication Engineering
Volume 1: Issue 6: June 2015, pp 55-59. www.aetsjournal.com ISSN (Online) : 2394-6237

Big Data Analytics in Cloud Platform for Load Rebalancing With Block Chain Algorithm

M.Vinothkumar, B.Sowmiya

Abstract— Distributed file systems are key building blocks for cloud computing applications based on the Map Reduce programming paradigm. In Map Reduce functionality, nodes simultaneously serve computing and storage functions; a file is partitioned into a number of chunks allocated in distinct nodes so that Map Reduce tasks can be performed in parallel. However, in a cloud computing environment, the files in a cloud can be arbitrarily created, deleted, and appended, and nodes can be upgraded, replaced and added in the file system, the file chunks are not distributed as uniformly as possible among the nodes. This results in load imbalance in a distributed file system. Emerging distributed file systems in production systems strongly depend on a central node for chunk reallocation. This is clearly inadequate in a large-scale, because the central load balancer is put under considerable workload that is linearly scaled with the system size, and may thus become the performance bottleneck and the failure may occur in file systems. . In this paper, the concept of "Data Skew" it refers to the imbalance in the amount of data assigned to each task, or the imbalance in the amount of work required to process such data. This is presented to scope the under loaded and over loaded imbalance problem. Implement Block Chain Algorithm to find the neighboring Nodes to reduce the movement cast in the data servers. The performance of our proposal implemented in the Hadoop distributed file system with a cluster environment.

Index Terms— Load balances, Distributed file systems, Map reduce, Data skew, Partitioning, Clouds.

I. INTRODUCTION

The past decade has witnessed the explosive growth of data for processing. Large Internet companies routinely generate hundreds of tera-bytes of logs and operation records. Map Reduce has proven itself to be an effective tool to process such large data sets. It divides a file system into multiple small tasks and assigns them to a large number of nodes for parallel processing. Due to its remarkable simplicity and fault tolerance, Map Reduce has been widely used in various applications, including web indexing, log analysis, data mining, scientific simulations, machine translation, etc. There are several parallel computing frameworks that support Map Reduce, such as Apache Hadoop, Google Map Reduce, and Microsoft Dryad, of which Hadoop is open-source and widely used. The job completion time in Map Reduce depends on the slowest running task in the file system. If one task takes significantly longer to finish than others (the so-called straggler) it can delay the progress of the entire job. Stragglers can occur due to various reasons, among which data skew is an important one. Data skew refers to the imbalance in the amount of data assigned to each task, or the imbalance in the amount of work required to process such data. The fundamental reason of data skew is that data sets in the real world are often skewed and that we do not know the distribution of the data beforehand. Note that this problem cannot be solved by the speculative execution strategy in Map Reduce. Implement the Block Chain Algorithm for reduce the movement cost in the various large data sets in the file system environment.

A System called Skew Reduce which optimizes the data partition for the spatial feature extraction application by operating preprocessing extracting and sampling procedures. Although these solutions can mitigate data skew to some extent, they have significant overhead due to the pre-run jobs and are applicable only to certain applications. Implement the Block Chain Algorithm for reduce the movement cost in the various large data sets in the file system environment.The performance of the proposal implemented in the hadoop distributed file system in further investigated in a cluster environment.

Hadoop cluster is a special type of computational cluster designed specifically for storing and analyzing huge amounts of unstructured data in a distributed computing environment. Typically one machine in the cluster is designated as the Name Node and another machine the as JobTracker; these are the masters. The rest of the machines in the cluster act as both Data Node and TaskTracker; these are the slaves. When a file is placed in HDFS it is broken down into blocks. These blocks are then replicated across the different nodes (Data Nodes) in the cluster. Whenever a file is placed in the cluster a corresponding entry of it location is maintained by the NameNode.

A file is partitioned into a number of chunks allocated in distinct nodes so that Map Reduce Tasks can be performed in parallel over the nodes. The load of a node is typically proportional to the number of file chunks the node possesses. Because the files in a cloud can be arbitrarily created, deleted, and appended, and nodes can be upgraded, replaced and added in the file system, the file chunks are not distributed as uniformly as possible among the nodes.

M.Vinothkumar is PG Scholar, Department of Computer Science and Engineering,, SRM University, India (Email: mvinothkumar1987@gmail.com)

B.Sowmiya is Assistant Professor, Department of Computer Science and Engineering, SRM University, India (Email: sowmiya.b@ktr.srmuniv.ac.in)

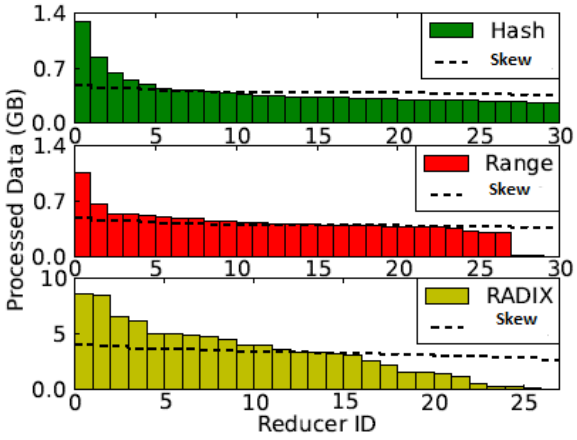

Fig 1: Data Processed By Reducer

The objective is to allocate the chunks of files as uniformly as possible among the nodes such that no node manages an excessive number of chunks.

II. RELATED WORK

The major part of the project development sector considers and fully survey all the required needs for developing the project. Before developing the tools and the associated designing it is necessary to determine and survey the time factor, resource requirement, man power, economy, and company strength. Once these things are satisfied and fully surveyed, then the next step is to determine about the software specifications in the respective system such as what type of operating system the project would require, and what are all the necessary software are needed to proceed with the next step such as developing the tools, and the associated operations.

Distributed file systems in clouds rely on central nodes to manage the metadata information of the file systems and to balance the loads of storage nodes based on that metadata. The centralized approach simplifies the design and implementation of a distributed file system. When the number of storage nodes, the number of files and the number of accesses to files increase linearly, the central nodes become a performance bottleneck. This results in load imbalance in a distributed file system. So the file chunks are not distributed as uniformly as possible among the nodes. They are unable to accommodate a large number of file accesses due to clients and Map Reduce applications. The existing solutions are designed without considering both movement cost and node heterogeneity and may introduce significant maintenance network traffic.

Hung-Chang Hsia et al., [1] has presented fully distributed load rebalancing algorithm is presented to cope with the load imbalance problem. This algorithm is compared against a centralized approach in a production system and also outperforms the prior distributed algorithm in terms of load imbalance factor, movement cost, and algorithmic overhead. To deal with the load rebalancing problem in large-scale, dynamic, and distributed file systems in clouds. The nodes take more loads and also maintain the consistency and speed.

J. Dean and S. Ghemawat et al., [2] has done an implementation of Map Reduce that scales to large clusters of machines comprising thousands of machines. The implementation makes efficient use of these machine resources and therefore is suitable for use on many of the large computational problems encountered at Google. Easy to use scalable programming model for large-scale data processing on clusters. It achieves efficiency through disk-locality and also achieves fault-tolerance through replication.

Gobioff et al., [3] has presented the Google File System demonstrates the qualities essential for supporting large-scale data processing workloads on commodity hardware. The system provides fault tolerance by constant monitoring, replicating crucial data, and fast and automatic recovery. Chunk replication allows us to tolerate chunk server failures. Additionally, the file system use check summing to detect data corruption at the disk or IDE subsystem level, which becomes all too common given the number of disks in the system.

I. Stoica et al., [4] has implemented about Chord that improves the scalability of consistent hashing by avoiding the requirement that every node know about every other node. A Chord node needs only a small amount of "routing" information about other nodes. Chord will be a valuable component for peer-to-peer, large-scale distributed applications and also adapts efficiently as nodes join and leave the system, and can answer queries even if the system is continuously changing. Chord acts as a distributed hash function, spreading keys evenly over the nodes. This provides a degree of natural load balance.

D. Karger et al., [5] has presented about Load balancing with critical issue for the efficient operation of peer-to-peer networks. A core problem in peer to peer systems is the distribution of items to be stored or computations to be carried out to the nodes that make up the system. An important issue in DHTs is load-balance, even distribution of items, or other load measures to nodes in the DHT. All DHTs make some effort to load balance; generally by (i) randomizing the DHT address associated with each item with a "good enough" hash function and (ii) making each DHT node responsible for a balanced portion of the DHT address space.

Y. Zhu and Y. Hu et al., [6] has presented a DHT based P2P systems offer a distributed hash table (DHT) abstraction for object storage and retrieval. Many solutions have been proposed to tackle the load balancing issue However, all these solutions either ignore the heterogeneity nature of the system, or reassign loads among nodes without considering proximity relationships, or both. The goal is to ensure fair load distribution over nodes proportional to their capacities, but also to minimize the load-balancing cost (e.g., bandwidth consumption due to load movement) by transferring virtual servers between heavily loaded nodes and lightly loaded nodes in a proximity-aware fashion.

M. Rinard et al., [7] has presented a new framework, called Histogram-based Global Load Balancing (HiGLOB) to facilitate global load balancing in structured P2P systems. Each node P in HiGLOB has two key components. The first

component is a histogram manager that maintains a histogram that reflects a global view of the distribution of the load in the system. The histogram stores statistical information that characterizes the average load of no overlapping groups of nodes in the P2P network. It is used to determine if a node is normally loaded, overloaded, or under loaded.

S. Surana et al., [8] has presented most P2P systems that provide a DHT abstraction distribute objects randomly among "peer nodes". Additionally, a node's load may vary greatly over time since the system can be expected to experience continuous insertions and deletions of objects, skewed object arrival patterns, and continuous arrival and departure of nodes. Consider a system in which , data items are continuously inserted and deleted, , nodes join and depart the system continuously, and , the distribution of data item IDs and item sizes can be skewed.

III. PROPOSED SCHEME

Map Reduce is an effective tool for parallel data processing. In cloud environment the data's are upgraded, replaced, and added in the system this load imbalance occur. The amount of imbalance data assigned to each task. Introduce data skew concept in Map Reduce applications to solve the unevenness in the data processing environment. It uses an innovative sampling method which can achieve a highly accurate approximation to the distribution of the intermediate data by sampling only a small fraction of intermediate data during the normal map processing. It allows reducing tasks to start copying as soon as the chosen sample map tasks complete. It supports the split of large keys when application semantics permit and the total order of the output data. It considers the heterogeneity of the computing resources when balancing the load among the reduce tasks appropriately.

Implement Load rebalancing in Hadoop and find out the neighboring nodes using Block Chain algorithm to apply the chain functionality in the Big Data environment. Our Experiment show that Load Rebalance has negligible overhead and can speed up the execution of applications

1. Psudeocode

A. Partitioning load data:

Number of elements in the Map and Reduce phase
$(A_1, B1), (A_2, B_2), (A_3, B_3)$……., (A_n, B_n)
A_i = Distinct Key in Map output.
B_i = Represent number of elements in the cluster.
$A_i < A_i+1$: Allow Range Partition to minimize the load of the largest number of tasks.
Range Partition Expression as (r – reduce tasks)
$O = ct_o < ct_1 < ct_2$……$< ct_r = n$.
Define Cost $(B_i) = B_i$

Minimize Task using Cost:

$$\max_{i=1,2,\ldots,r}\{\sum_{j=ct_{i-1}+1}^{pt_i} Cost(B_j)\}$$

B. Sampling load data:

Light weight sampling procedure
(A_i^j, B_i^j) - A_i^j output of the Task
$E_{sample} = E_{largest} U E_{normal}$
$E_{largest}$- Contains number of elements in the largest $B_i^j = p$
E_{normal} - Randomly selected for the rest task for uniformly Distribution except $E_{largest} = q$
Calculate degree of Data Skew – Average Coefficient (p/q).

$$avg\ COV_{p/q} = \frac{\sum_{\sigma \in E} COV_{\sigma}^{p/q}}{|E|}$$

Calculate Accuracy of Tasks

$$\sqrt{\frac{\sum_{i=1}^{n}(x_i^{appro} - x_i^{real})^2}{n}}$$

2. Block Chain Algorithm

Input searching for each and every node of the cloud environment.

- Identify input available cloud specific location for every node (L_n)
- To form a chain environment for this specific input available locations
- Lo = L_1-> L_2-> L_3……L_n forming chain environment.
- Next time searching the same word again then the chain rule apply for produce the output
- Reduce Time constraint and Movement Cost.

IV. EXPERIMENTAL RESULTS

The ratio of $p=q$ is positively related to the degree of the data skew: the heavier the skew, the larger the ratio should be. To select a good ratio, we generate 10GB synthetic data sets following Zipf distributions with varying σ parameters (from 0.2 to 1.4) to control the degree of the skew. Larger σ value means heavier skew That shows the coefficient of variation in data size across reduce tasks with different $p=q$ ratio and skew degree σ .

$$avg\ COV_{p/q} = \frac{\sum_{\sigma \in E} COV_{\sigma}^{p/q}}{|E|}$$

From the result, we can find that the optimal $p=q$ ratio which can work well in both skew and non-skew cases is 0.10. Therefore, we set the default $p=q$ ratio to 0.10.

1. Average Coefficient of task variation

p/q ratio	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40	0.45	0.50
avg COV	0.433	0.405	0.552	0.589	0.659	0.762	0.839	0.86	0.961	1.005

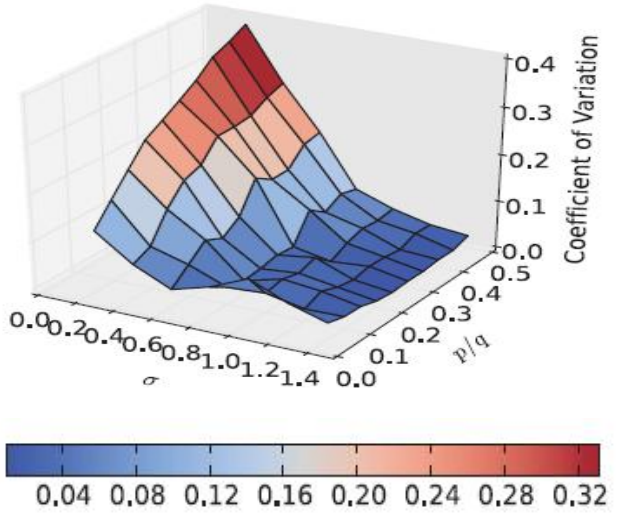

Fig 1: Co-efficient variation of data skew

2. Task Execution time comparison sorting in load rebalancing

A major motivation for data skew mitigation is to improve the job execution time. As we can see from the below figure, the improvement is dramatic: the execution speed in system is 80% faster than that in Hadoop hash. When compared to Hadoop range (which supports the total order), our improvement jumps to 167%. In fact, by the time Hadoop finishes its pre-run sample of the input data; our system has already completed its entire execution. The figure also shows that the overhead of our sampling method is negligible: our combined sample/map phase is about the same length as the map phase in Hadoop hash and Hadoop range (which perform no sample). This demonstrates the efficiency of our carefully designed algorithm. Our much improved execution time in the reduce phase is because Data Skew can partition the intermediate data much more evenly (as evidenced in figure). The coefficient of variation among all reduce tasks in Data Skew is only 0.07, while in Hadoop range and Hadoop hash it reaches 0.47 and 0.51, respectively.

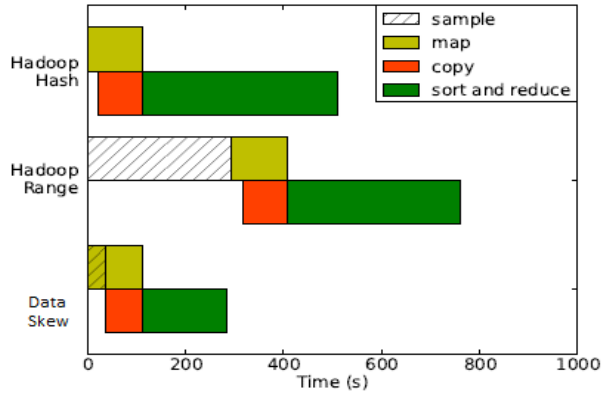

Fig 2: Task Time comparison sort

3. Degrees of Data Skew in Map Reduce concept using Hadoop in Cloud environment

The input data exhibits different degrees of skew, we repeat the previous experiment but with σ varying from 0.2 to 1.5. Figure shows how the job execution time and the coefficient of variation change when the skew increases. For the two strategies in Hadoop(marked 'Hadoop hash' and 'Hadoop range') and Skew Tune (We use Hadoop hash as the original practitioner), both metrics increase substantially once the degree of the skew reaches a certain threshold. As we can see from the figure, this optimization has a profound impact on partitioning the data evenly. With this optimization, both the job execution time and the coefficient of variation remain very low as σ increases. (We have continued the experiments for σ up to 2.0 and find that the curves remain flat.) Without this optimization, the curves begin to climb up after σ reaches a certain threshold. Even so, our system performs better than Hadoop hash and much better than Hadoop range and Skew Tune. The reason that Skew Tune performs worse than Hadoop hash is that Skew Tune does not detect or split large keys and hence cannot make a better partition decision.

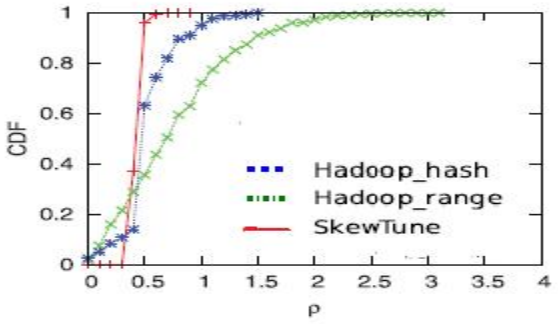

Fig 3: Degree of Data Skew Tune

4. Block Chain Algorithm based Time difference Job Performance

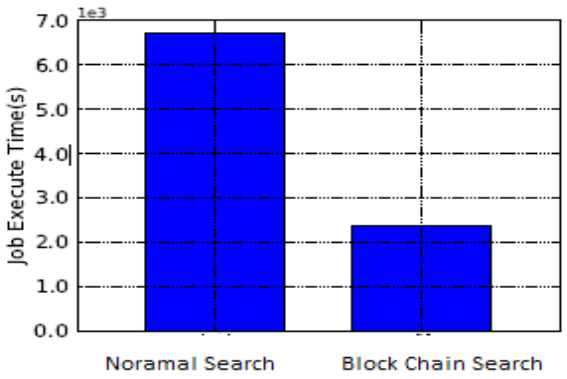

Fig 4: Block Time Constraint

Using Block Chain Algorithm reduced the searching time for the cloud environment. Compare to the Normal Searching time the Block chain based searching time only searching the available location of the input data. The figure shows the time difference.

V. CONCLUSION

An efficient Data Skew Processing to deal with the load imbalance problem in large-scale, dynamic and distributed file systems in clouds has been presented. The implementation is demonstrated through a small-scale cluster environment consisting of a single, dedicated name node and data nodes. The proposal strives to balance the loads of data nodes and task nodes efficiently. Then only can able to distribute the file chunks as uniformly as possible. The proposed Block Chain algorithm using to identify the neighbor nodes and operates in a distributed manner in which nodes perform their load-balancing tasks independently without synchronization or global knowledge regarding the system. In a load-balanced cloud, the resources can be well utilized and provisioned, maximizing the performance of Map Reduce-based applications.

VI. FUTURE WORK

Our proposal utilizes the resources very efficiently. It will be very effective if it also concentrate on the processing time and processing speed. When compared to existing system our proposal can make an acceptable change in the processing time. In future, Response time is also need to be considered effectively.

REFERENCES

[1] Hung-Chang Hsiao, Member, ieee Computer Society, Hsueh-Yi Chung,Haiyingshen, Member, ieee, And Yu-Chang Chao, "Load Rebalancing For Distributed File Systems In Clouds" Ieee Transactions on Parallel and Distributed Systems, Vol. 24, No. 5, May 2013

[2] J.Dean and S. Ghemawat, proposed a "Map Reduce: Simplified Data Processing on Large Clusters," in Proc. 6th Symp. Operating System Design and Implementation (OSDI'04), Dec. 2004, pp. 137–150.

[3] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," Proc. 19th ACM Symp. Operating Systems Principles (SOSP '03), pp. 29-43, Oct. 2003.

[4] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan,"Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," IEEE/ACM Trans. Networking, vol. 11, no. 1, pp. 17-21, Feb. 2003.