International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 03 Issue: 05 | May-2016 www.irjet.net p-ISSN: 2395-0072

An Efficient And Cost Effective URL Indexer For Better Search Engine

Ms. Manasa R[1], Mr. A V Krishna Mohan[2]

[1]*M.Tech student, Dept. of CSE, SIT, Tumakuru, Karnataka, India*
[2]*Assistant Professor, Dept. of CSE, SIT, Tumakuru, Karnataka, India*

Abstract –*The usage of internet is growing enormously these days and the information in the cloud can be extracted using web pages by search engines. Because of rapid development in the web applications, the large numbers of web pages are getting created each day. The web crawlers update information about the web pages into a distributed key-value store in the cloud, keyed by the URL of the page and time stamped by the time of the crawl. For Search Engine Optimization, many search engines adopt Hadoop for maintenance of an index of the various website. The URL data gathered in the cloud is huge and this data can be categorized as Big data. Maintenance of this huge data of URL indexes involves huge cost. To reduce the cost involved, we have to reduce the usage of the cloud. To optimize the computation time, here we propose an efficient URL indexer by incorporating a Hadoop based effective scheduler that reduces the overall computation time of the indexing Big data by allocating idle task slots to the prioritized job, thus reducing the cost.*

Key Words: **Big Data, Cloud computing, Hadoop, Scheduler, Indexer**.

1. INTRODUCTION

The enormous growth in Information technology (IT) industry has led to the growth of virtualization of servers. The Huge cost is involved in building this virtualization of the storage server. The cost not only involves investing in technology but also in maintaining these. This led IT industry to virtualize the storage, computation to cloud and use of Big data technologies for data management in the cloud. Since the cloud service is highly scalable, flexible and platform independent, cloud technologies are used widely by all kind of users. In the recent days, since the cloud computing has the provision of configurable computing resources it is grabbing the attention of many users. Cloud provides the resource to users on demand and user is required to pay for the demanded resources from the cloud.

The cloud solutions extended by the cloud service providers [1], like Google Apps, are giving service to the user online on the internet. The cloud service providers offer the solutions to the user round the clock in the cloud. Authenticated user can access the information in the cloud securely from any ware. To build any application and to make it available to user one has to setup the server and install the hardware physically and run the necessary software. Thus, building and maintaining the actual physical server involves cost and it lacks scalability, and however use of cloud service providers for the same has an advantage of huge computation capacity with quicker and cheaper servers. Cloud service providers are adding flexibility to business which is the starting step and does not have an estimation of storage usage in the future. So such users can use the cloud platform and extend their storage capacity whenever required.

The data that is accumulating in the cloud because of financial trading, advanced web technologies, social networking user generated data and e-commerce details can be categorized as Big data. Big data involves huge volume, high-variety and high-velocity data, thus, quick and simple processing techniques are required to manage this big data. In recent days, Big data technologies are concentrating towards efficient techniques to manage the Big data. Efficient data processing techniques are required to reduce the time of result computation. So by reducing the computational time, cost of cloud usage can be reduced. Thus, to minimize the cost of cloud usage, it is recommended to have fast computational procedures for Big data in the cloud.

For indexing the Big data, there are a few potential challenges. First and foremost challenge is the digital information that is getting accumulated which is too huge to manage by people or normal software. And single computing system cannot store and manage this Big data hence data should be stored and manage in distributed manner. Therefore, big data indexing should be built keeping the distributed system into consideration. Second challenge being, the accumulated data has a complex structure, data is heterogeneous in nature and is high dimensional. Hence, the traditional index techniques with normal data set are insufficient to index the big data. Adding to these challenges we have to consider that the techniques to build the indexing should be fault tolerant.

Here in this paper, we are considering the World Wide Web page information that is stored in the key-value stores [2] to provide the index for that big data. The URL index data that is accumulating for a search engine cloud is growing enormously day by day. The URL indexed data needs to be maintained in an efficient manner to reduce the usage cost of the cloud by any user. Old and obsolete data must be deleted, malicious URL's should be removed from the index and so many maintenance works are involved. So it is need of an hour to have the technique to minimize the computational time for all the maintenance works needs to be done in the cloud, thus by decreasing the cloud usage.

The usage of Hadoop [3] on a cloud computing environment which contains data acute operations speed up the processing and it is necessary to have techniques for enhancing throughput. The Hadoop inbuilt scheduler is originated on a First in First out (FIFO) mechanism. The final throughput is reduced for multiple jobs submission when FIFO is used as a scheduler. Here we propose a scheduler which executes the shorter jobs in priority for multiple users in parallel without any waiting period. This helps in quicker execution of shorter jobs without any waiting period and thus, accelerates the system performance. And we also propose an effective URL indexer, which provide a positioning mechanism for web pages saved in the cloud.

2. LITTERATURE SURVEY

MapReduce [4, 5] is a programming model; it is a framework for managing the huge data sets with a coordinated, distributed method and a framework for implementation to process the larger data sets in the cluster. A Map operation categorizes the input (like, sorting employees by their employee ID into a list and one queue for each ID) and filtering and a Reduce operation performs an analysis process (like the statistics of employees in every queue is counted, recognizing its frequencies).

MapReduce is a framework which transforms the memory representation of data in the distributed servers to a data format which is apt for storage and transform between various servers, runs the diverse tasks in coordinated manner, the data transfer and the communication of data between the diverse components of the system is also managed, and provides redundancy and flaw tolerances [5].

The functional programming models also use the map and reduce functions and the MapReduce framework is inspired by that, however, the purpose achieved in MapReduce is different from their primitive forms. The prime offerings of the MapReduce programming technique are the scalability and the fault tolerance achieved by Map and Reduce functions. The optimization of an execution engine is done to achieve the scalability and the fault-tolerance for the different applications. Such as a single-threaded implementation of MapReduce is almost same as a traditional implementation technique, only the multithreaded implementations prove the faster execution. Good MapReduce algorithm can only be achieved by optimizing the communication cost.

3. SCHEDULING TECHNIQUES

The scheduling of available jobs in Hadoop[6] is performed by a master node. Master node with TaskManager distributes work to all of the slaves. Every few seconds, slaves used to send heartbeat messages to the master, and then tasks are assigned in depending on the received heartbeats. The exact numbers of slots are maintained by slave node for each Map and Reduce functions. Since Hadoop slots have a single thread for single node they are considered as a single threaded task manager. The slot model can sometimes underutilized system resources; however this model manages the memory and CPU utilization in a better manner. The FIFO scheduling schedules the larger jobs if they come first, then the shorter jobs have to wait for a long time resulting in poor response times. This disadvantage makes the interactive applications which submit a shorter job for execution has to wait for a long time resulting in poor performance.

3.1 Fair scheduler

One of the scheduler called as fair scheduler [7] which is a scheduler for Hadoop at slot-level granularity.

Features of fair scheduler are:

- Isolation: Isolation feel is given to the each job in the queue by providing the illusion of running a private cluster.
- Statistical Multiplexing: If the capacity given to some slots is unused by some users then redistribute that unused capacity to other slots.

The fair scheduler uses a two-level hierarchy. The given capacity is divided into multiple pools. At the first level, this scheduler allocates task slots across many pools, and at another level, for multiple jobs in the pool, slots are allocated in a fair manner. Each pool is guaranteed with a minimum share of slots until there is a demand for the resource. The constraint over here is the total sum of the minimum shares of all pools should not exceed the total system capacity. The pools if they required more than the minimum share given to them they can use the other pools slots when those slots are not being used by the pools.

In fair scheduling, for large jobs or production jobs special pools are allocated and for normal user jobs or short jobs, one pool per each are allocated. This practice makes the shorter jobs independent of longer jobs thus providing good response time. However, fair scheduling has the disadvantage of jobs getting starved because the reduce task has to starve till all dependent map tasks to be completed and if there is a dependent log running map task then reduce function must be on hold till the complete execution of the map task.

3.2 Capacity Scheduler

Capacity scheduler [8] was designed by Yahoo!.

Features of capacity scheduler are:

- Locality-aware: It automatically allocates the resources to the nodes which are nearby. Hence, programmers need not worry about the locality of the resources.
- Access Control Lists (ACL): It uses ACLs to provide access to the users and administrators.
- User limit: This scheduler can set the minimum resource usage for the each user in the queue

All available jobs are organized into different queues. The hierarchical queue is used to manage the jobs. The Certain

capacity of the available resources is allocated to each queue. The FIFO scheduling is followed within each queue. Idle resources of one queue can be allocated to another queue which is in need. The job submission limits are determined by the scheduler and scheduler decides if the job can be executed or not. A job which takes too much of time can be preempted. In this scheduler single user cannot monopolize resource usage because capacity scheduler has a mechanism to limit the percentage of resources assigned to the user. Jobs can specify the higher memory requirement for the execution and capacity scheduler can run these kinds of jobs in the clusters with higher available memory. The resource capacity can be configured by the job. This scheduler ensures that certain amount of available capacity is shared among users rather than sharing among jobs.

4. PROPOSED SYSTEM

In this paper, we propose a computational model which is an extension to the existing MapReduce scheduler which is used in the index ranker to give positioning to the URL indexes in the cloud. Here the URL index data present in the cloud can be positioned based on the importance of incoming links and outgoing links using the graph representation. During execution of URL index position algorithm, one can implement the efficient scheduler to minimize the computation time of indexer execution. Fig-1 shows the multiple queues allocated with the CPU resources in priority in the efficient scheduler.

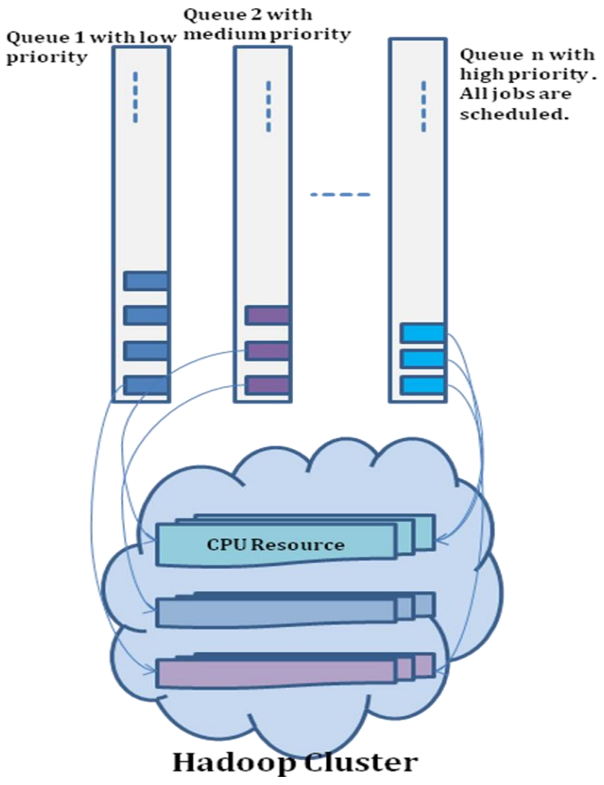

Fig-1: Efficient scheduler

4.1 Efficient Scheduler

The proposed efficient scheduler is an add-on feature to MapReduce scheduler, which creates an interface for sharing the CPU resource in single computing cluster to multiple job submissions in the uniform manner and queues are given with priority to have the CPU share. The priority to a queue is given by considering the job execution cost model. The entire cluster pools are allocated to an individual job, which is running at the present time when there is no other job in the queue, but the new jobs with a high priority when arrived at the same cluster; new jobs are allocated with the idle task slots present in the pool. If the new job comes with low priority than the one being executed, the new job is scheduled only if idle slots are available otherwise the new job is scheduled only after completion of the running job. This enables idle tasks to aid the parallelizing of job execution.

The proposed efficient job scheduler allocates the usable resources to the jobs with the highest priority and schedules the priority jobs into free pools. This scheduler uses the quick message exchange communication system between JobTracker [9] and TaskTracker [10]. The system performance is improved due to this optimization in the planning of job execution since jobs are executed in parallel and smaller jobs are finished in short or no waiting time without looking forward to the execution of heavy jobs. Since short jobs are given with the separate slots to execute, they can be executed as soon as they appear in the queue. And usage of quick messaging improves the task scheduling mechanism speed for computations which involves intensive data. The FIFO mechanism is used as the default scheduler in Hadoop which is inefficient to manage the multiple jobs. Hadoop with FIFO is not efficient when compared to the proposed efficient scheduler when couples of jobs are accommodated into a single cluster.

In the proposed scheduler for jobs, few of parameters like preemption of low priority jobs, giving the number of jobs to be executed in the pool, to avoid the reduce tasks till completion of all map tasks etc can be configured. This feature makes the applications to have the customized execution of the job and make the execution fast. Whenever high priority jobs are submitted, they can be configured in the customized way to speed up the execution.

4.2 Effective URL indexer

Here we show the URL Index ranker by using MapReduce approach to parallelism. The internet has a huge collection of web pages so that it is the tedious job to find the URL Index position. There may exist a web page which contains the spam, to find out this we can use the number of incoming links to the web page. Here we are not searching the web page by the content alone. Thus, the URL Index ranker algorithm is the best way of using the collective intelligence to resolve the importance of web pages.

Here we describe a single-machine implementation which can easily handle millions of web pages. URL Index ranker uses the adjacency matrix and calculates the position score for each web page and this is an algorithm to assign a rank for

each web page based on importance and efficiency [11]. It is the link analysis algorithm, here a numerical cost is assigned to each set of hyperlink documents and the importance of the set of hyperlink document set is discovered.

The algorithm works by using MapReduce technique, in map phase each and every vertex sends a value of its outgoing link to all of its neighbors and in reduce phase, positioning is calculated by adding all the values the vertex received from its neighbors.

The MapReduce phase can be executed for more than one time to have the accurate results. And the incremental big data processing is adopted over here, where results of the previous computations are used for further analysis. This incremental big data processing saves huge computational cost as computation can be done only for the small portion of the data which is affected after previous computation. It is redefined with the probability distribution, to represent the random click on links and hyperlinks to arrive at any instant web page [12].

Once the positioning of the web page is calculated we can use the same value to find out the spam pages present in the cloud. We can set the minimum positioning threshold value and the web pages with the position value less than the threshold value can be garbage collected from the cloud. The web pages which are not referred to any other web pages will get the very little positioning value, thus, they can be removed from the cloud.

Thus, using this effective URL indexer, the search engine can return the result for the user query with most relevant web pages. Thus, here we can provide the high position to the page with a number of incoming links, the page with high importance. The positioning is assigned to the all web pages thus positioning can be considered as an index to that web page and data is stored in the key-value store. Whenever a new search was given to the search engine, the search engine can refer to the indexed value of the web page and return the web page with relevant information with high positioning value among the others. While calculating the index position value we are using the MapReduce with the efficient scheduler which speeds up the job execution. Usage of an add-on scheduler improves the throughput of the entire process.

5. CONCLUSION

Here we proposed a model based on MapReduce for processing the Big data in the cloud present in an incremental manner. By adopting the incremental computational model it reduces the huge computational cost. Hence, the big data involves a huge amount of data, to perform the computation of that data repeatedly involves more computational cost. The proposed efficient MapReduce scheduler combines an iterative model which is general-purpose with incremental analysis engine and the iterative computational techniques for effective incremental computation. This algorithm is very efficient which is useful for parallelism in map reducing. Here we present big data concepts and characteristics to manage data effectively in the cloud. And also explain the URL Index ranker algorithm for map reducing concept in big data. Thus, overall this framework reduces the time taken for computational work by adopting effective scheduler and other techniques so the usage cost of the cloud is greatly reduced to the cloud users. This is resulting in the better search engines to manage huge data of a large number of web pages in the cloud.

Using the positioned indexes we can manage to find the best URL and use them for future and the URL with less position value can be removed from the cloud. Thus, make this work as the maintenance of the storage and cost in an effective manner.

ACKNOWLEDGEMENT

My Sincere thanks are due to my supervisor A V Krishna Mohan, for the valuable guidance offered during every stage of my project work. I thank Principal, Siddaganga Institute of Technology and the management of Siddaganga Institute of Technology for the support, encouragement and facilities provided at the institute.

REFERENCES

[1] Amazon Elastic Compute Cloud, http://www.amazon.com/ec2/, accessed on March 2014.

[2] https://en.wikipedia.org/wiki/Key-value_database

[3] Apache Hadoop, http://hadoop.apache.org., accessed on March 2014.

[4] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters", Comm. of the ACM, Vol. 51, No. 1, 2008, pp. 107-113.

[5] MapReduce Algorithms for Big Data Analysis, DNIS 2013

[6] "Job Scheduling for Multi-User MapReduce Clusters" by Matei Zaharia,Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, Ion Stoica.

[7] Apache Fair Scheduler, http://hadoop.apache.org/docs/r2.3.0/hadoopyarn/ hadoop-yarn-site/FairScheduler.html, accessed on March 2014.

[8] Hadoop Capacity scheduler https://svn.apache.org/repos/asf/hadoop/common/tags/release-0.19.2/docs/capacity_scheduler.pdf

[9] https://wiki.apache.org/hadoop/JobTracker

[10] https://wiki.apache.org/hadoop/TaskTracker

[11] "The PageRank Citation Ranking: Bringing Order to the Web" by Lawrence Page, Sergey Brin, Rajeev Motwani.

[12] http://www.webworkshop.net/pagerank.html.