International Journal of Research in Engineering, Science and Management
Volume-3, Issue-2, February-2020
www.ijresm.com | ISSN (Online): 2581-5792

New Scheduling Algorithms for Improving Performance and Resource Utilization in Hadoop YARN Clusters

P. Priyadharshika[1], S. Sajithabanu[2], N. Balasubramanian[3]
[1]*Student, Department of MCA, Mohamed Sathak Engineering College, Ramanathapuram, India*
[2,3]*Assistant Professor, Department of MCA, Mohamed Sathak Engineering College, Ramanathapuram, India*

***Abstract*: Bigdata is the large amount of data just beyond technology's capability to store, manage and process efficiently. MapReduce is a framework for processing and managing large scale data sets in a distributed cluster, which has been used for applications such as generating search indexes, document clustering, access log analysis, and various other forms of data analytics. In existing system, a hash function is used to partition intermediate data among reduce tasks. In this project the system proposed a decomposition-based distributed algorithm to deal with the large-scale optimization problem for big data application and an online algorithm is also designed to adjust data partition and aggregation in a dynamic manner.**

***Keywords*: Yarn, Scheduling algorithms, Data processing, Map Reduce.**

1. Domain Introduction

Big data can be structured, unstructured or semi-structured, resulting in incapability of conventional data management methods. Data is generated from various different sources and can arrive in the system at various rates. In order to process these large amounts of data in an inexpensive and efficient way, parallelism is used. Big Data is a data whose scale, diversity, and complexity require new architecture, techniques, algorithms, and analytics to manage it and extract value and hidden knowledge from it. Hadoop is the core platform for structuring Big Data, and solves the problem of making it useful for analytics purposes. Hadoop is an open source software project that enables the distributed processing of large data sets across clusters of commodity servers. It is designed to scale up from a single server to thousands of machines, with a very high degree of fault tolerance.

Big data is a term that refers to data sets or combinations of data sets whose size (volume), complexity (variability), and rate of growth (velocity) make them difficult to be captured, managed, processed or analyzed by conventional technologies and tools, such as relational databases and desktop statistics or visualization packages, within the time necessary to make them useful. While the size used to determine whether a particular data set is considered big data is not firmly defined and continues to change over time, most analysts and practitioners currently refer to data sets from 30-50 terabytes (1012 or 1000 gigabytes per terabyte) to multiple petabytes (1015 or 1000 terabytes per petabyte) as big data.

2. Hadoop yarn cluster

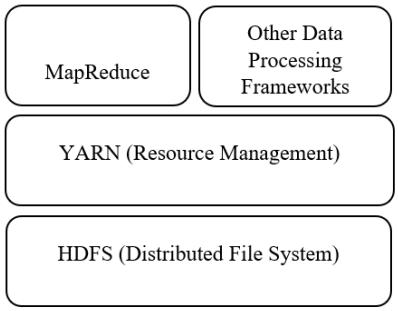

Fig. 1. Hadoop 2.0 Architecture

In Hadoop YARN stands for "Yet Another Resource Negotiator". It was introduced in Hadoop 2.0 to remove the bottleneck on Job Tracker which was present in Hadoop 1.0. The Scheduler in Resource manager of YARN architecture allows Hadoop to extend and manage thousands of nodes and clusters. Apache Hadoop YARN sits between HDFS and the processing engines being used to run applications. It combines a central resource manager with containers, application coordinators and node-level agents that monitor processing operations in individual cluster nodes. YARN offers clear advantages in scalability, efficiency and flexibility compared to the classical MapReduce engine in the first version of Hadoop.

3. Introduction

In this project, I proposed a distributed algorithm for big data applications by decomposing the original large-scale problem into several sub problems that can be solved in parallel. The system designs an online algorithm whose basic idea is to postpone the migration operation until the cumulative traffic cost exceeds a threshold. The network topology is based on three-tier architectures: an access tier, an aggregation tier and a core tier. The system investigates network traffic reduction

within MapReduce jobs by jointly exploiting traffic-aware intermediate data partition and data aggregation among multiple map tasks.

MapReduce and MapReduce-like models are widely used to process “Big Data”. This paper targets at provisioning a virtual cluster according to the position relationship between VMs so as to decrease the network traffic and improve the performance of MapReduce and MapReduce-like applications rather than modifying the job scheduling strategies or VM configurations.

4. Scheduling algorithms

There are many types of process scheduling algorithms. we see some of them as below:

A. First Come First Serve (FCFS)

The process which enters the queue first is executed first. It is non-preemptive. It is a troublesome algorithm for time sharing systems. Performance is highly dependent on the order in which jobs arrive.

B. Shortest-Job-First (SJF)

Associates to each job/process a unit of time to complete (completion time). Implemented with non-preemptive policy. Useful for batch-type processing where waiting for jobs to complete is not critical. This can potentially improve job throughput by ensuring shorter jobs are executed first and thus potentially save shorter turnaround time.

C. Priority Scheduling

Priority scheduling is a method of scheduling processes based on priority. Priority scheduling involves priority assignment to every process, and processes with higher priorities are carried out first, whereas tasks with equal priorities are carried out on a first-come-first-served or round robin basis.

5. Online algorithm

An online algorithm is one that can process its input piece-by-piece in a serial fashion, i.e., in the order that the input is fed to the algorithm, without having the entire input available from the beginning. Eg: Insertion sort. Insertion sort is an online algorithm. It produces the optimum result, i.e., a correctly sorted list.

6. Relevant works to the scheduling

This section presents the relevant works related to using large data sets in our system for implementing new scheduling algorithm. This new scheduling algorithm finds weight, arrival rate, execution time, performance for our datasets.

7. System architecture

In this system, I Increase the amount of data and the availability of high performance. Database systems have been extended and parallelized to run on multiple hardware platforms to manage scalability. simplify the parallel processing using a distributed computing platform that offers only two interfaces. Reduce network traffic within a MapReduce job, we consider to aggregate data with the same keys before sending them to remote reduce tasks.

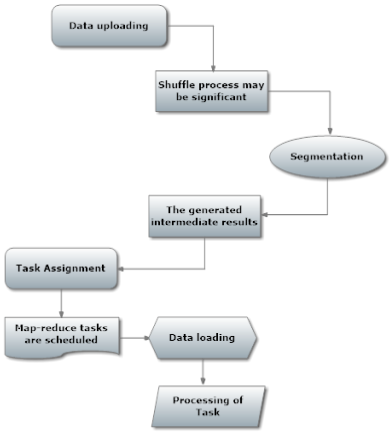

Fig. 2. Flowchart

A. Data uploading

The master schedules map tasks in the workers by taking into account of data locality. The output of the map tasks is divided into as many partitions as the number of reducers for the job. Entries with the same intermediate key should be assigned to the same partition to guarantee the correctness of the execution. All the intermediate key/value pairs of a given partition are sorted and sent to the worker with the corresponding reduce task to be executed.

B. Segmentation

The system considers a typical MapReduce job on a large cluster consisting of a set N of machines. We let xy denote the distance between two machines x and y, which represents the cost of delivering a unit data. When the job is executed, two types of tasks, i.e., map and reduce, are created. The sets of map and reduce tasks are denoted by M and R, respectively, which are already placed on machines.

C. Task assignment

The access tier is made up of cost-effective Ethernet switches connecting rack VMs. The access switches are connected via Ethernet to a set of aggregation switches which in turn are connected to a layer of core switches. An inter-rack link is the most contentious resource as all the VMs hosted on a rack transfer data across the link to the VMs on other racks. Our VMs are distributed in three different racks, and the map-reduce tasks are scheduled.

D. Processing of task

The system divides the execution of a MapReduce job into several time slots with a length of several minutes or an hour. We let $m^{p}_{j}(t)$and α (t) denote the parameters collected at time slot t with no assumption about their distributions. As the job is

running, an existing data partition and aggregation scheme may not be optimal anymore under current.

$$m^{p}{}_{j}(t) \text{and } \alpha(t)$$

8. Evaluation process

The system evaluates the performance of proposed algorithm under online cases by comparing it with other two schemes: OHRA and OHNA, which are online extension of HRA and HNA, respectively. The default number of mappers is 20 and the number of reducers is 5. The maximum number of aggregators is set to 4 and we also vary it to examine its impact. The key/value pairs with random data size within [1-100] are generated randomly in different slots. The total number of physical machines is set to 10 and the distance between any two machines is randomly choose within

9. Conclusion

In this project, we upload our large data set into a system that segments our data into a multiple job. Our job results are store in a separate data centre. This helps us to analyze and our data easily and also we find weight, arrival rate, performance and execution time to our data.

References

[1] M. Yazdani, M. Amiri, and M. Zandieh, “Flexible job-shop scheduling with parallel variable neighborhood search algorithm,” Expert Systems with Applications, vol. 37, no. 1, pp. 678–687, 2010.
[2] A. Verma, Ludmila Cherkasova, and R. H. Campbell, “Aria: Automatic resource inference and allocation for mapreduce environments,” in ICAC’11, 2011, pp. 235–244.
[3] J. Polo, D. Carrera, Y. Becerra, J. Torres, E. Ayguad´e, M. Steinder, and I. Whalley, “Performance-driven task co-scheduling for mapreduce environments,” in Network Operations and Management Symposium (NOMS), IEEE, 2010, pp. 373–380.
[4] V. V. Vazirani, “Approximation algorithms.” springer, 2001.
[5] R. Grandl, G. Ananthanarayanan, S. Kandula, S. Rao, and A. Akella, “Multi-resource packing for cluster schedulers,” ACM SIGCOMM Computer Communication Review, vol. 44, no. 4, pp. 455–466, 2015.
[6] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” Communications of the ACM, vol. 51, no. 1, pp. 107–113, 2008.