Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 89 (2016) 203 – 208

Procedia Computer Science

Twelfth International Multi-Conference on Information Processing-2016 (IMCIP-2016)

Tolhit – A Scheduling Algorithm for Hadoop Cluster

M. Brahmwar*, M. Kumar and G. Sikka

Dr. B.R. Ambedkar National Institute of Technology, Jalandhar 144 011, India

Abstract

With the accretion in use of Internet in everything, a prodigious influx of data is being observed. Use of MapReduce as a programming model has become pervasive for processing such wide range of Big Data Applications in cloud computing environment. Apache Hadoop is the most prominent implementation of MapReduce, which is used for processing and analyses of such large scale data intensive applications in a highly scalable and fault tolerant manner. Several scheduling algorithms have been proposed for Hadoop considering various performance goals. In this work, a new scheme is introduced to aid the scheduler in identifying the nodes on which stragglers can be executed. The proposed scheme makes use of resource utilization and network information of cluster nodes in finding the most optimal node for scheduling the speculative copy of a slow task. The performance evaluation of the proposed scheme has been done by series of experiments. From the performance analysis 27% improvement in terms of the overall execution time has been observed over Hadoop Fair Scheduler (HFS).

Peer-review under responsibility of organizing committee of the Organizing Committee of IMCIP-2016

Keywords: Hadoop; MapReduce; Heterogeneous Cluster; Job Scheduling; Big Data.

1. Introduction

In this era of data science, data is considered as a key source for promoting growth and wellbeing of the society. Perusing the affinity in data helps to associate information and formulate strategies efficiently. Every day, more than two quintillion bytes of data is being created in this info-centric digitized world from various sources like scientific instruments, web authoring, telecommunication industry, social media, etc. Therefore, the effective storage and analysis of such tremendous amount of data has become a great challenge for the computing industry. In order to solve this crucial problem of analyzing such large data sets various computing paradigms such as grid computing and cloud computing came into existence. However, these computing paradigms were rendered abortive as their debugging, load balancing and scheduling solutions were found to be inefficient when dealing with such large data. As a result of which various solutions[1–3] were introduced to handle the Big Data Applications efficiently.

MapReduce[1] is the most approved computational framework that utilize adaptive and scalable approaches of distributed computing for processing large data sets. Programs implemented using this functional style are parallelized implicitly. These are processed on a large cluster built from commodity hardware. The system itself is responsible for the partitioning of input data, scheduling of jobs across a set of commodity machines, handling the machine failures,

*Corresponding author. Tel.: +918284095097.
E-mail address: manvibrahmwar@gmail.com

Peer-review under responsibility of organizing committee of the Organizing Committee of IMCIP-2016
doi:10.1016/j.procs.2016.06.043

and managing the required inter-machine communication at run time. This allows naive programmers in the field of parallel and distributed systems to easily make use of such large resources of distributed systems.

Apache Hadoop[4], created by Doug Cutting, is the most universally recommended open source implementation of MapReduce framework. It is used for storing, processing and analyzing large data sets across clusters of commodity hardwares in a reliable and fault tolerant manner. Hadoop version 2.x comprises of the following three components: HDFS[5], YARN[6] & MapReduce. Hadoop Distributed File System (HDFS) implemented by Yahoo is used by Hadoop for storing input and output data. It is also responsible for dividing the input data into fixed sized blocks and then allocating these data blocks to different data nodes. By default Hadoop maintains a replication factor of 3 i.e. it replicates each input block into 3 data nodes. Yet Another Resource Negotiator (YARN) also known as MapReduce version 2 is the foundation of new generation of hadoop. The fundamental idea of YARN is to split up the prominent functionalities of the Job Tracker which are resource management and job scheduling/monitoring, into 2 different daemons. The overall performance of a Hadoop cluster depends upon its scheduler. However, the performance of Hadoop's default scheduler is observed to deprecate in a non homogeneous environment. The scheduler is also not efficient enough in identifying the slow tasks which protract the overall execution time. Many scheduling algorithms[7–13] have been proposed as important extensions of Hadoop's default scheduling algorithm.

In this work, a new scheduling approach has been introduced which assists the Hadoop scheduler in finding the most optimal nodes on which a speculative copy of stragglers can be executed in a heterogeneous hadoop cluster. The scheduling scheme named "Tolhit" makes use of resource utilization and cluster network information in identifying the most appropriate choice for executing the slow tasks so that the delay in overall execution time can be reduced. From the experiments an improvement of approx. 27% in terms of execution time is observed over Hadoop Fair Scheduler (HFS). The remaining paper is scripted as follows. Segment number 2 covers the literature insights of Hadoop MapReduce Scheduling. The innovative approach is described in segment number 3. Segment number 4 explores the performance analysis of the proposed method followed by conclusion in segment number 5.

2. Related Work

By default Hadoop comes with three configurable scheduler policies; these are FIFO scheduler, Capacity scheduler[7] & Hadoop Fair Scheduler (HFS)[8]. First In First out (FIFO) scheduler does not assure fair sharing among users. It also lacks in performance for small jobs in terms of response time. Capacity Scheduler was proposed by Yahoo to make the cluster sharing among organizations possible. This was done by setting the minimum guaranteed capacity of the queues. Facebook also proposed HFS to fairly share the cluster among various users and applications.

Several scheduling algorithms have been proposed to cater the needs of different kind of workloads in different environments. As an important improvement of Hadoop's default scheduler, Zaharia *et al.* proposed LATE[9] for efficiently launching speculative tasks. Longest Approximate Time to End (LATE) scheduling scheme considers cluster heterogeneity for the first time. It also succeeds in improving the locality constraint by compromising fairness slightly. The constraints like deadline of a job are accurately considered in scheduling algorithm[11]. Self Adaptive MapReduce (SAMR)[12] scheduling policy as a supplement to LATE algorithm also considers historical information besides hardware heterogeneity to synchronize the weights of map and reduce stages dynamically. It however lacks behind as it does not consider other factors like jobs with different types and sizes which might also affect the stage weights. To subjugate the shortcomings of SAMR algorithm, Enhanced Self Adaptive MapReduce (ESAMR)[13] scheduling algorithm was introduced by Sun *et al.* The ESAMR algorithm records and reclassifies historical information for each job at every node by adopting K-Means[14] clustering algorithm. It dynamically tunes stage weights and finds slow tasks accurately.

2.1 Motivation

Hadoop is designed to follow Master-Slave architecture model. The master and the slave nodes in a Hadoop cluster communicate by exchanging heartbeat messages with each other at regular intervals. Each heartbeat message is considered as a potential scheduling opportunity for an application to run a container. Whenever a heartbeat message is received from a node about having an empty slot, a job is selected according to the configured scheduling policy.