Evaluating the Job Performance using DyScale Scheduler and MapReduce in Hadoop framework

Supriya.R, Kantharaju.H.C
Department of Computer Science and Engineering,
Vemana Institute of Technology, Bangalore-560034
rsupriya53@gmail.com, kantharajuhc@gmail.com

ABSTRACT-As there has been a massive growth in the social media in all the aspects from over a span of ten years, the amount of photos being uploaded to the Internet has been increased. Photos are being shared, downloaded and uploaded in surpass quantity through many online services like WhatsApp, Facebook, Instagram to name a few. But the applications making use of this uploaded photos are very few. Hence, in order to make use of the photos in a meaningful way, we have devised an image processing application on Hadoop framework. Image processing in this case, is used to fetch the metadata information about an image and produce the output. The framework consists of a new scheduler named DyScale using which the images are processed and compared with the fair scheduler which is usually used in Hadoop. The scheduler provides a way to use the underlying resources very efficiently and improves the performance by taking less time to process when compared to the fair scheduler.

Keywords – Hadoop, Image processing, MapReduce, Performance, Scheduling.

I. INTRODUCTION

Multi-core processing is a developing industry which replaces the single core processing system rapidly, it has the physical limits of possible complexity and speed. The present SoC (System on Chip) is designed such that it provides variety of choices in the same power envelope and help us to evaluate decision trade off. For example we can select the cores depending on the application what is being used. Hadoop is one of the most scalable and fault tolerant framework for executing bulky data sets [1]. MapReduce jobs are processed on a big cluster of service based machines. These jobs are automatically parallelized, distributed and executed over commodity machines. Hadoop was mainly designed to perform batch oriented processing of huge production jobs. The performance goals of MapReduce workloads are: large throughput oriented batch jobs and smaller response time sensitive interactive jobs. Therefore it is very difficult to take decision whether using MapReduce applications is more beneficial as compared with processors with faster cores. For example Hadoop users process a large MapReduce jobs by applying thumb rule on double size Hadoop cluster, and reduce the jobs completion in half. This thumb rule is applied on jobs, which has multiple tasks and process large data sets. The job completion time can reduce by processing these tasks on a larger number of slots. These tasks are throughput oriented and efficient processing of such jobs can be improved with additional scale out resources.The scalable operational model and data storage in Hadoop allows users to dig out information by using advanced mining technique and machine learning based algorithms to figure out novel data insights in non-traditional, game changing ways. In general, the main use of MapReduce cluster is for hosting different datasets, and multiple applications will share its compute capacity. An interesting design of heterogeneous multi-core processors [2] is to provide both fast and slow cores, for supporting different performance objectives of MapReduce jobs.

DyScale can be abbreviated as dynamic scaling scheduler with scale out or scale up approaches. Scale out approach refers to adding more servers with less RAM and processors, whereas scale up approach refers to adding more RAM and processors and also buying robust and expensive server. If MapReduce jobs are small then scale up approach can be preferred.

The paper is organized as follows. Section II briefs about the background of MapReduce, section III gives the detailed system architecture, section IV describes the image processing steps used, section V provides the results, section VI concludes the paper and gives a way to the future enhancement

II. BACKGROUND OF MAPREDUCE

MapReduce is a programming model for processing a huge amount of parallelizable data. It facilitates distributed processing of data through map and reduce stages. The map stage is partitioned into map tasks and the reduce stage is partitioned into reduce tasks. The map and reduce tasks are executed by map slots and reduce slots. Input data is processed by map function to produce data in the form of tuples and reduce function is used to combine the values along with the key. The split of input data is read by each Map task in the map stage, and applies it to generate set of intermediate key-value pairs. In the next stage map task splits and sorts these data for based on a partition function for different reduce tasks.

In the second stage, every reduce task takes its split of intermediate key-value pairs and combine these data with the same key. This method is call as shuffle or sort phase. After this, value list will be merged by using user defined reduce function to get aggregate results. This is also called as reduce phase. Output obtained from this stage will be written back on distributed file system

The job scheduling in Hadoop is generally performed as shown in the Fig 1. In Hadoop, the master node performs job scheduling, and it is called as job tracker. A Job tracker

takes request from a client and allocates with the task to the task tracker to perform. Periodically, the task tracker connects with the master job tracker and reports the present status with available slots. Based on scheduling policy and the information reported by master job tracker, job tracker decides the next job which needs to be executed. Task tracker accepts map, reduce and shuffle task from the job tracker. Job tracker keeps receiving notification from task tracker to confirm that job tracker is still active.

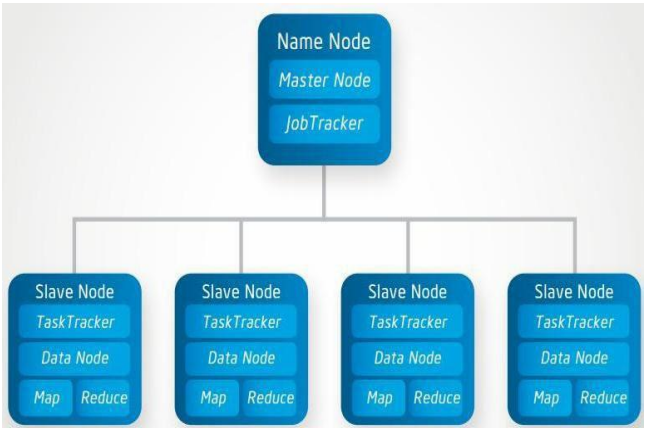

Fig 1: Job scheduling in Hadoop

III. SYSTEM ARCHITECTURE

The DyScale architecture is depicted in the Fig 2. It contains master and slave daemons. The main job of master daemon is to handle slave daemons. The slave daemon runs on every cluster nodes. DyScale executes tasks on the slaves. The master allows resources to be allocated over framework by making them available when needed. The resources like CPU, RAM contains list of Id's as (slave ID, resource1: amount1, resource2, amount2 …).

Based on organizational policy which the master chooses how many to offer in the current framework i.e. fair sharing or strict priority. To manage a diverse set of policies, master performs a plug in mechanism to utilize modular architecture. It also helps us to include new allocation modules.

There are two main components present on the framework which is running on top of DyScale which are scheduler and executer. The scheduler offers the resources and executes a framework task. It is launched on slave nodes. Master tells how many resources need to be offered to each framework and scheduler select which offered resources to be used. A description of task will be passed to DyScale when offered resources will be accepted by framework. In turn, DyScale launches the tasks on the corresponding slaves.

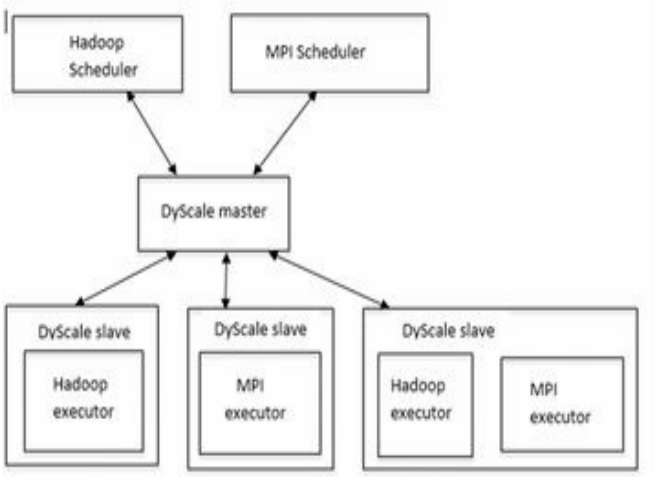

Fig 2: DyScale scheduler architecture and its components

IV. IMPLEMENTATION

In order to show the performance evaluation between the schedulers, we are incorporating image processing in Hadoop, where initially the images are downloaded fromthe internet, processed and the metadata is extracted from the images, then later the images are run on the schedulers in order to compare the performance. The steps in the implementation are as follows:

4.1 Hadoop Image Processing Interface (HIPI)

HIPI program considers HipiImageBundle (HIB) as a primary input object. HIB represents set of images characterized as single file on the HDFS (Hadoop Distributed File System). The distribution of HIPI contains MapReduce program that is designed on HIB from a list of images. It can be downloaded from the internet. The HIPI distribution includes MapReduce program that builds a HIB from a list of images downloaded from the Internet. The culling step is the initial stage of HIPI program. In this stage images are filtered based on different types of user defined conditions i.e. spatial resolution or criteria related to the image metadata.

The images which remain after a culling stage, are allocated to individual map task that tries to maximize data locality, which is a keystone of the Hadoop MapReduce programming model. HIPI also supports for OpenCV. OpenCV is a popular open source computer vision library. Whatever the records comes out from mapper are collected together and passed it on the reducer. Reducer uses shuffle algorithm [6] to reduce traffic on network. As the final point, all user defined reduced jobs are processed in parallel and the result is aggregated and written on HDFS. The working procedure of HIPI program is shown in Fig 3.

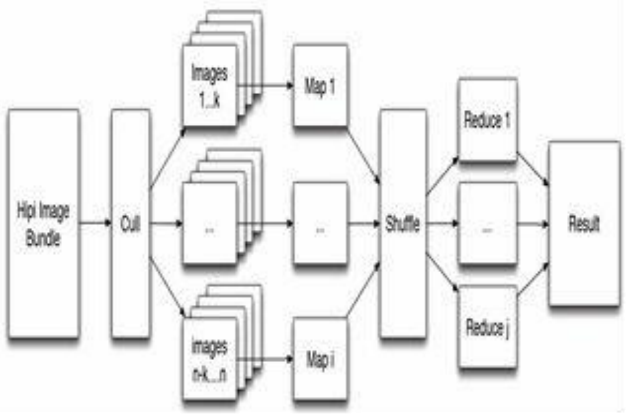

Fig 3: Organization of the HIPI program

Downloading and Storing the Image Data

In order to download and store the images, Hadoop uses the Hadoop Distributed File System (HDFS). First the image URL's list must be stored in a text file, then this image URL list is split across nodes which facilitates parallelization. Then the MapReduce will run in order to download the images from the URL list. Once it is downloaded it is stored in an image bundle known as HIB.

Processing the image bundle using MapReduce

Once the images downloaded it needs to be processed. Each individual image is processed after retrieving the image data type, after the processing the processed image is attached to the temporary image bundle where the images are stored. Each map task generates the bundles which are scattered across. In reduce phase all the scattered image files are merged into a large processed file. In order to process the images the user can use his own program based on his needs.

He needs to set the key and value as

setOutputKey (k1)

setOutputValue (v1)

Then run the below command in the terminal.

MIP-0.0.1-SNAPSHOT.jar prj.imageDataSet

which means that the user must include the specific jar file in order to process the images and the file path must be specified by the user where the images are processed and converted into the hib file and stored.

Extracting the image bundles

The framework also provides a method to extract and view the images. The extractor module is devised which extracts the images in parallel from all the available nodes. After the images are extracted identifying and arranging them in a final location is a daunting task, but our framework overcomes this difficulty by providing the user simply to extract the images in Hadoop system or in the local system. In order to extract the hib file, again the use the user should run the following command

MIP-0.0.1-SNAPSHOT.jar

MassiveImageProcessing.MIP.ImagePrs which allows the user to extract the images and store it in the path specified by the user. The MapReduce will run in order to extract the images.

2 RESULTS

After running the image dataset on two schedulers, the performance analysis graph can be plotted as bar chart shown in the Fig 4. In order to plot the graph, we should note the CPU time spent while processing the each image dataset on both the schedulers. The x-axis denotes the image dataset considered. The y-axis denotes the time taken in milliseconds. The grey colour shows the time taken by the fair scheduler to process the image whereas the blue colour shows the performance of the DyScale scheduler which takes significantly takes less time when compared to the fair scheduler which is usually used in Hadoop environment.

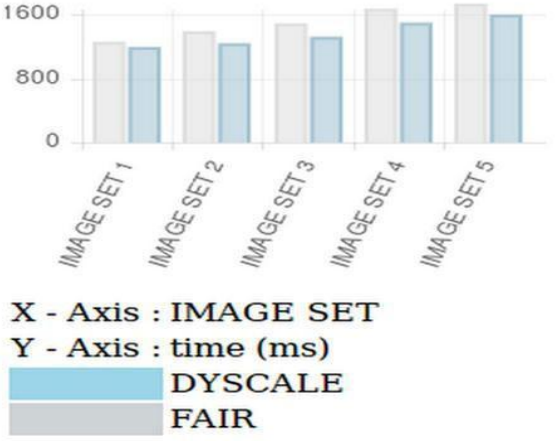

Fig 4: Performance analysis of the DyScale scheduler.

CONCLUSION

The paper can be concluded as follows, it clearly shows that the performance can be improved by using the DyScale scheduler in order to process large form of image data as it lesser time when compared to other schedulers, the dataset can be extended to other formats as well in the future. The image processing can be done considering many other use cases or criteria, also in future the process can be run on high end machines and more number of nodes in cluster while considering varied forms of input like video files and audio files.

REFERENCES

[1] J. Dean and S. Ghemawat, ―Mapreduce: simplified data processing on large clusters,‖ in Proc. of OSDI '04, 2004.

[2] F. Ahmad et al., ―Tarazu: Optimizing MapReduce on Heterogeneous Clusters,‖ in Proceedings of ASPLOS, 2012.

[3] M. Zaharia et al., ―Improving MapReduce performance in Heterogeneous environments,‖ in Proceedings of OSDI, 2008.

[4] J. Xie et al., ―Improving MapReduce performance through data placement in heterogeneous hadoop

clusters,‖ in Proceedings of the IPDPS Workshops: Heterogeneity in Computing, 2010.

[5] G. Lee, B.-G. Chun, and R. H. Katz, ―Heterogeneity-aware resource allocation and scheduling in the cloud,‖ in Proceedings of the 3rd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud, 2011.

[6] J. Polo et al., ―Performance management of accelerated MapReduce workloads in heterogeneous clusters,‖ in Proceedings of the 41st Intl. Conf. on Parallel Processing, 2010.

[7] W. Jiang and G. Agrawal, ―Mate-cg: A map reduce-like framework for accelerating data-intensive computations on Heterogeneous clusters,‖ in Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International, May 2012, pp. 644–655.

[8] Q. Chen, D. Zhang, M. Guo, Q. Deng, and S. Guo, ―Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment,‖ in IEEE 10th International Conference on Computer and Information Technology (CIT), 2010.

Biographies and photographs

Ms Supriya R has received her BE degree in Computer Science from VTU, in 2014. She is currently pursuing her M.Tech degree in Computer Science in VIT, Bangalore. She has authored one journal paper and published in international conference. Her research interests include Big Data, Image processing.

Mr Kantharaju H C has received his BE degree in Information Science from VTU in 2009 and ME degree in Computer Science from Anna University in 2011. He is currently working as an Assistant Professor in VIT, Bangalore since 2013. He has authored and co-authored 5 research papers in national and international proceedings. His research interests include networking and operating systems.