ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology
(IJARTET) Vol. 5, Special Issue 14, April 2018

BIG DATA PROCESSING FOR A DILIGENT APPLICATION WITH MULTI LAYER CLUSTERS IN SECURED STORAGE SYSTEM

M.SHIVA PRIYA, Assistant Professor, Bandari Srinivas Institute of Technology, Hyderabad
shiva6priya@gmail.com
N.ANOHU, Assistant Professor, Bandari Srinivas Institute of Technology, Hyderabad,
P.SUNITHA, Assistant Professor, Bandari Srinivas Institute of Technology, Hyderabad
A.VENKATESH, Assistant Professor, Bandari Srinivas Institute of Technology, Hyderabad,

ABSTRACT

Hadoop frame work has been designed to processing large amount of data in Big data Processing. Much task is involved for the assignment of resources and executing order of tasks to progress an output in a homogeneous cluster. In this paper we explore the storage design in a multi layer system by the account of bundled jobs.Where input data and related application jobs are comply in a bundled.Our main objective is to remove the railing between resource management and latent storage layer and enhance data locality , salient presentation characteristic for resource management from the cast of storage system. We progress sampling based randomized algorithm for network file system to find out placement of input data blocks. The theme is to query a required set of candidate nodes and evaluate their efficiency at runtime union of consolidated and per node information . The node with the minimum work load is choose to capture the data blocks. The consequences of this paper proved outstanding performance enhanced in terms of execution time and data locality.

INTRODUCTION

The current big data processing platforms adopt the 'scale- out' solution, where a cluster of servers collaborate to ac- complish big data applications. The application job and the corresponding input data are split into small tasks and asso- ciated data blocks respectively. Each task can be executed on any of the cluster nodes that can allocate the task's demanded computing resources. In such platforms, the storage layer is usually managed by a network file system where data blocks are distributed across the cluster with replicas. Data locality is extremely important to the system performance when executing the big data application because the current platforms only prefer 'move-compute-to-data' paradigm, but do not enforce it. It is possible that a task is executed on a server that does not host the task's input data block. In this case, the input data block will have to be transferred from another cluster node incurring a transmission overhead into the taskexecution.The data locality issue has been explored in a lot of prior work, and new enhancement schemes have been proposed to increase the chance of executing a task on the server hosting its input data. However, the performance improvement from these existing solutions is limited by dynamic run-timefactors in practice, especially in a heterogeneous cluster where each server has different hardware and software configurations. The fundamental cause that hinders the optimization on data locality is the separation of job execution and input data uploading processes. In all the prior work, data locality is only considered during the job execution by the job/task scheduler assuming that the input data have been uploaded into the network file system in a separate process. Without considering the initial data block placement during the uploading process, the existing solutions cannot optimize the data locality with various job workloads and cluster configurations. In addition, the prior work considers that the system performance of executiontimeormakespanreferstothejobexecutionprocess excluding

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology
(IJARTET) Vol. 5, Special Issue 14, April 2018

the time spent in uploading the input data. In practice, however, there are many *bundled jobs* submitted to a processing cluster that include both application jobs and the reference of the input data that needs to be transferred to the network file system. For this type of bundled jobs, the performance should include the time spent in both uploading theinputfileandexecutingtheassociatedjobs.

PROPOSED WORK

In this paper, we target on the execution of bundled jobs in a heterogeneous cluster, and present a new storage layer that efficiently places data blocks across the cluster to improve the data locality. When dispatching data blocks of the input data, our solution considers the associated application jobs and estimates the future executions. We apply randomized algorithms to simplify and speed up the solution. Essentially, our algorithm queries a small set of randomly selected candidate nodes for hosting a data block to estimate their run-time workload. The data block will be uploaded to the node with the smallest workload aiming to balance the workload of all the nodes and thus improve the data locality. We implement our solution in Hadoop YARN platform and evaluate it with experiments and simulation. The results are highly supportive with significant performance improvements.

BACKGROUND ANDMOTIVATION

In this section, we present our target environment settings and the background information about the representative Hadoop YARN platforms. We also introduce the policies and issues in the existing storage layer of Hadoop, and bring up ourmotivationontheefficientstoragedesign.

A. DataLocality

The focus of this paper is to improve the data locality which is an important performance factor in a Hadoop system. This subsection briefly reviews the background and presents the issues of the traditional system.

A Hadoop cluster is built upon a network file system (HDFS) that manages the placement of all the data blocks (with replicas). Hadoop defines three types of data locality for map tasks: node-local, rack-local, and off-switch, referring to different cases of the location of the resource container and input data block of a task. 'Node-local' indicates that the container for executing the task is allocated on a node that hosts the corresponding input data block. In the case of 'rack-local', the node that is about to execute the map task does not host the input data block, but there is a copy of the data in the same rack, and can be transferred to the executing node. In 'off-switch', the data block has to be transferred from another rack for the execution. In either 'rack-local' or 'off- switch', a network overhead is incurred for transferring the data block to the node that allocates a container to execute the task. It is not negligible considering the size of a data block, shortexecutiontimeofamaptask,andheavytrafficloadsin

B. Bundled jobs:

WeconsidertheHadoopclusterisrunningatypeofbundled jobs,whereitheinputdataanditsprocessingjobsaresubmitted together as a bundle. The input data can be uploaded by the user, or from external storage sites such as Amazon S3. In this setting, the user considers the cluster as a computing site, but not for long-term or permanent data storage. After the jobs are finished, the input data will be deleted from the HDFS. Fig. illustrates the basic process. The main difference compared to the traditional Hadoop is that the HDFS is aware of the jobs associated with the input data, and thus could use this informationtobetterdispatchthedatablocksinthecluster.

ISSN2394-3777 (Print)
ISSN2394-3785 (Online)
Available online atwww.ijartet.com
International Journal of Advanced Research Trends in Engineering and Technology
(IJARTET) Vol. 5, Special Issue 14, April 2018

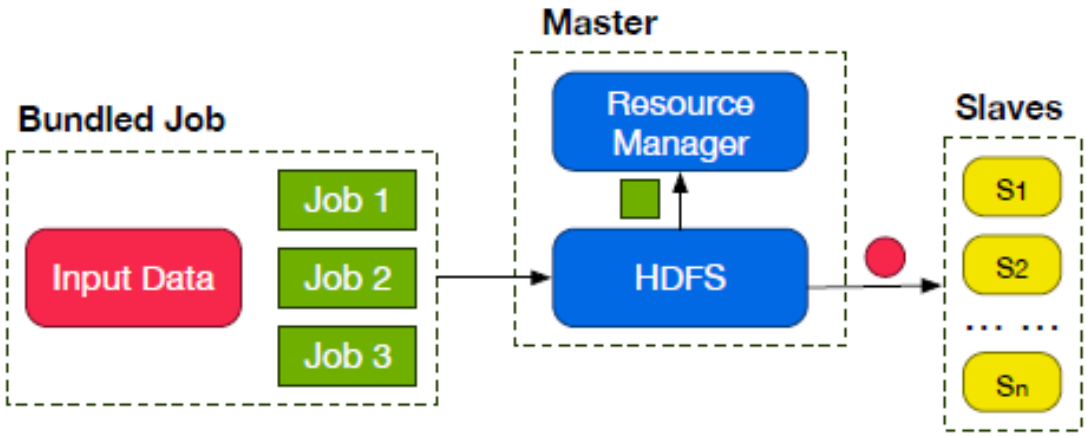

Fig.: Illustration of processing a bundled job

CONCLUSION

Inthispaper,wedevelopanewstoragesystemtoefficiently servebundledjobsinlargescalebigdataprocessingplatforms. Our system improves the data block placement when uploading thembyconsideringtheassociatedapplicationjobs.Sampling- based randomized algorithm is adopted to select the candidate node with the minimum workload to host new data blocks. In addition, we develop algorithm to efficiently and accurately estimate the workload of each candidate node. Our solution is evaluated with experiments and simulation. The results show that our solution is significantly superior to the native Hadoop storagesystem.

REFERENCES

[1] Apache Hadoop NextGen MapReduce (YARN). http://hadoop.apache.org/docs/current/hadoop-yarn/ hadoop-yarn-site/YARN.html.
[2] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, et al. Job scheduling for multi-user mapreduce clusters. Technical report, EECS Department, University of Cali- fornia, Berkeley, Apr2009.
[3] Capacity Scheduler. http://hadoop.apache.org/common/ docs/r1.0.0/capacityscheduler.html.
[4] Ali Ghodsi, Zaharia, Benjamin Hindman, and etc. Dom- inant resource fairness: Fair allocation of multiple re- source types. In *8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 323–336, Berkeley, CA, USA, 2011. USENIX Associ- ation.
[5] Abhishek Verma, Ludmila Cherkasova, and Roy H. Campbell. Aria: Automatic resource inference and al- location for mapreduce environments. In *ICAC*, pages 235–244,2011.
[6] Kamal Kc and Kemafor Anyanwu. Scheduling hadoop jobs to meet deadlines. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second Interna- tional Conference on*, pages 388–392.IEEE, 2010.