International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2014): 5.611

Survey on Resource Allocation in Phase-Level using MapReduce in Hadoop

Suryakant S. Bhalke[1]

[1]JSPM's Imperial College of Engineering & Research, Wagholi, Pune

Abstract: ***MapReduce is programming tool for Hadoop cluster. While allocating resources, MapReduce has two levels: Task-level and Phase-level. These levels should be used to check performance of each job. In existing system, the scheduling is focus on task level which tasks can have highly varying resource requirements during their lifetime and also its difficult to effectively utilize available resources to reduce job execution time. To address this limitation, this project proposes a PRISM (Phase and Resource Information - aware Scheduler MapReduce) which allocates a fine-grained resource at the phase-level to perform job scheduling. The job scheduling of prism is performed by the master node, which maintains a list of jobs in the system. Each node manager (slave node) periodically sends a heartbeat message to the scheduler. Upon receiving the status message from a node manager running on machine, the scheduler computes the utilization for set of candidate phases for the tasks using the jobs phase-level resource requirement. Then it select the phase with the highest utility for scheduling and update the resource utilization of the machine. This process is continued for until scheduled the phases of map and Reduce task is completed.***

Keywords: *MapReduce, Hadoop, Scheduling, Resource Allocation*

1. Introduction

Hadoop is an open source under the Apache fund account component, and is an open source implementation of Google graphs calculation model. It can easily develop and run large-scale data processing. Two of the most core part are HDFS (Hadoop Distributed File System) and MapReduce.

Businesses today are increasingly reliant on large-scale data analytics to make critical day-to-day business decisions. This shift towards data-driven decision making has fueled the development of MapReduce [10], a parallel programming model that has become synonymous with large scale, data-intensive computation. In MapReduce, a job is a collection of Map and Reduce tasks that can be scheduled concurrently on multiple machines, resulting in significant reduction in job running time. Many large companies, such as Google, Facebook, and Yahoo!, routinely use MapReduce to process large volumes of data on a daily basis. Consequently, the performance and efficiency of MapReduce frameworks have become critical to the success of today's Internet companies.
Motivated by this observation, several recent proposals, such as resource-aware adaptive scheduling (RAS) [15] and Hadoop MapReduce Version 2 (also known as Hadoop NextGen and Hadoop Yarn) [7], have introduced resource aware job schedulers to the MapReduce framework. HoAuthorver, these schedulers specify a fixed size for each task in terms of required resources (e.g. CPU and memory), thus assuming the run-time resource consumption of the task is stable over its life time. HoAuthorver, this is not true for many MapReduce jobs. In particular, it has been reported that the execution of each MapReduce task can be divided into multiple phases of data transfer, processing and storage [12]. A phase is a sub-procedure in the task that has a distinct purpose and can be characterized by the uniform resource consumption over its duration. As Author shall demonstrate in Section 2.2, the phases involved in the same task can have different resource demand in terms of CPU, memory, disk and network usage. Therefore, scheduling tasks based on fixed resource requirements over their durations will often cause either excessive resource contention by scheduling too many simultaneous tasks on a machine, or low utilization by scheduling too few.

1.1 HDFS

The Hadoop distributed file system (HDFS) to store large files with streaming data access patterns, to run with managers-workers mode, that is, there is a Name Node (managers) and multiple Data Nodes (workers). Name Node manages the file system tree and the tree in all of the files and directories. Data Node is usually a Node in the cluster, a record of each file in each block of Data Node information.

1.2 MapReduce

MapReduce work process is divided into two phases: the Map and Reduce phase. A Map function, which is used to put a set of keys for mapping into a new set of key-value pairs. And it points to the Reduce function. MapReduce has four parts: the framework of homework submission and initialization, task allocation, task execution and completion of the homework.

Firstly user program (Job Client) submits a job, and then the job of the information will be sent to the job Tracker. Job Tracker is the center of the Map - reduce framework, which needs to communicate with the cluster machine timing (heartbeat), and need to manage what program should be run on which machines, to manage job failed, restart operation. TaskTracker is a part of each machine in MapReduce. It is designed to surveillance resources of their machines. TaskTracker monitoring tasks run of the current state of the machine. TaskTracker needs sends the information through the heartbeat JobTracker. JobTracker will collect these information to assign new job submitted a run on which machines.

2. The Framework of Hadoop YARN

YARN is the resource management system in the Hadoop 2.0. It splits the JobTracker of MRv1 into two independent service: a global resource manager named Resource Manager and Application Master of each application. The Resource Manager is responsible for the resource management and allocation of the whole system, while ApplicationMaster responsible for the management of a single application.

YARN is still the Master/Slave structure. In the resource management framework, the Resource Manager is Master, NodeManager is a Slave, and the ResourceManager is responsible for all the resources on the NodeManager for unified management and scheduling. YARN is mainly composed of the ResourceManager, NodeManager, ApplicationMaster and several Container components.

- **ResourceManager (RM):** RM is a global resource manager, is responsible for the resource management and allocation of the whole system. It is mainly made up of two components: the Scheduler (Scheduler) and the application Manager (Applications Manager, ASM);
- **ApplicationMaster (AM):** Each application contains 1 AM. There are the main features: Negotiate with RM scheduler for resources, Tasks within the task assigned to further, Communicate with NM to start/stop the task, the Monitor all tasks running state;
- **NodeManager (NM):** NM is on each node of resources and task manager. On the one hand, it will report regularly to the RM this node on the resource usage and the running state of every Container. On the other hand, it receives and deal with the Container from AM start/stop and other requests;
- **Container:** Container is resource abstraction of the YARN. It encapsulates the multi-dimensional resources on a node, such as memory, CPU, disk, network and so on.

3. System Architecture

A fine grained, phase-level scheduling scheme that allocates resources according to the phase that each task is currently executing. The job scheduling in PRISM is performed by the resource manager in the master node, which maintains a list of jobs in the system. The phase level scheduler will use the provided information to make scheduling decisions. When a task needs to be scheduled, the scheduler replies to the heartbeat message with a task scheduling request. The node manager then launches the task. Each time a task finishes executing a particular phase, the task asks the node manager for a permission to start the next phase . The task of each pase is scheduled based on the utility of that phase. The scheduler assigns a utility value to each phase which indicates the benefit of scheduling the phase. The utility function is calculated based on the fairness and job performance of the particular phase. Then it select the phase with the highest utility for scheduling and update the resource utilization of the machine.

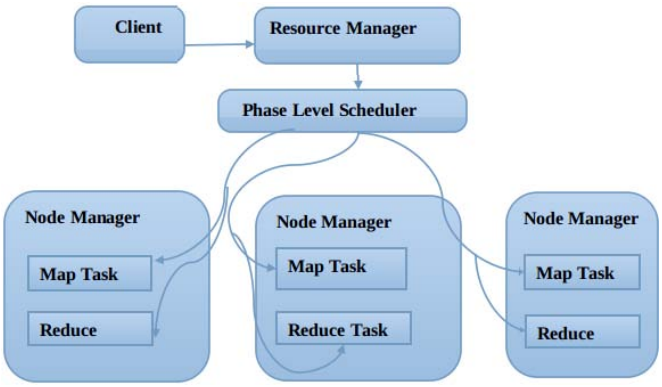

Figure 1: System Architecture

4. System Block Diagram

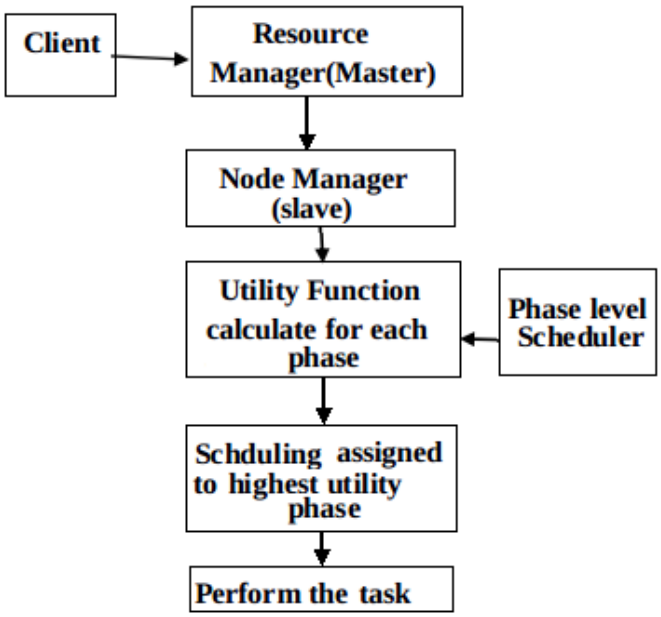

Figure 2: System Block Diagram

5. Challenges

- Varying resources at the task-level offer author performance.
- It is difficult for task-level scheduler to utilize the run-time resources. So that it reduces job execution time while executing

Finally, even though the flexibility of phase-based scheduling should allow the scheduler to improve both resource utilization and job performance over existing MapReduce schedulers, realizing such a potential is still a challenging problem. This is because pausing the task execution at run-time may delay the completion of the current and subsequent tasks, which may increase the job completion time (these delayed tasks are commonly referred to as stragglers [10]). Thus, the scheduler must avoid introducing stragglers when switching bet Author phases. In the following sections, Author will describe how PRISM overcomes this challenge

6. Conclusion

MapReduce is programming model for cluster to perform a data-intensive computing. In this paper Author mainly demonstrate that, if the resources focus on task-level, execution of each task may divide into many phases. While executing these phases, many breaking- down of map and reduce tasks will takes place and execute them in a parallel across a large number of machine, so that it will reduce running time of data-intensive jobs. So they will perform resource allocation at the phase-level. Author will introduce PRISM at the phase-level. PRISM demonstrates that, how run-time resources can be used and how it varies over the long life time. PRISM improves job execution algorithm and performance of resources without introducing stragglers.

References

[1] Qi Zhang, Student Member, IEEE, Mohamed Faten Zhani, "PRISM: Fine-Grained Resource-Aware Scheduling for MapReduce" IEEE Transactions On Cloud Computing, Vol. 3, No. 2, April/June 2015.
[2] Hadoop MapReduce distribution [Online]. Available: http://hadoop.apache.org, 2015.
[3] Hadoop Capacity Scheduler [Online]. Available: http://hadoop.apache.org/docs/stable/capacity_scheduler.html/, 2015.
[4] Hadoop Fair Scheduler [Online]. Available: http://hadoop.apache.org/docs/r0.20.2/fair_scheduler.html, 2015.
[5] R. Boutaba, L. Cheng, and Q. Zhang, "On cloud computationalmodels and the heterogeneity challenge," J. Internet Serv. Appl.,vol. 3, no. 1, pp. 1–10, 2012.
[6] T. Condie, N. Conway, P. Alvaro, J. Hellerstein, K. Elmeleegy, andR. Sears, "MapReduce online," in Proc. USENIX Symp. Netw. Syst.Des. Implementation, 2010, p. 21.
[7] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processingon large clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.
[8] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker,and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types," in Proc. USENIX Symp. Netw. Syst. Des. Implementation, 2011, pp. 323–336.
[9] H. Herodotou, H. Lim, G. Luo, N. Borisov, L. Dong, F. Cetin, andS. Babu, "Starfish: A self-tuning system for big data analytics,"in Proc. Conf. Innovative Data Syst. Res., 2011, pp. 261–272.
[10] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, and K. Talwar,"Quincy: Fair scheduling for distributed computing clusters,"in Proc. ACMSIGOPS Symp. Oper. Syst. Principles, 2009, pp. 261–276.
[11] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang. "Multi-resource allocation: Flexible tradeoffs in a unifying framework," in Proc. IEEEInt. Conf.Comput. Commun., 2012, pp. 1206–1214.
[12] J. Polo, C. Castillo, D. Carrera, Y. Becerra, I. Whalley, M. Steinder,J. Torres, and E. Ayguad_e, "Resource-aware adaptive schedulingfor MapReduce clusters," in Proc. ACM/IFIP/USENIX Int. Conf.Middleware, 2011, pp. 187–207.
[13] A. Rasmussen, M. Conley, R. Kapoor, V. T. Lam, G. Porter, and A.Vahdat, "ThemisMR: An I/O-Efficient MapReduce," in Proc.ACM Symp. Cloud Comput., 2012, p. 13.
[14] A. Verma, L. Cherkasova, and R. Campbell, "Resource provisioningframework for MapReduce jobs with performancegoals," in Proc. ACM/IFIP/USENIX Int. Conf. Middleware, 2011,pp. 165–186.
[15] D. Xie, N. Ding, Y. Hu, and R. Kompella, "The only constant ischange: Incorporating time-varying network reservations in datacenters," in Proc. ACM SIGCOMM, 2012, pp. 199–210.
[16] Y. Yu, M. Isard, D. Fetterly, M. Budiu, _ U. Erlingsson, P. Gunda,and J. Currey, "DryadLINQ: A system for general-purpose distributeddata-parallel computing using a high-level language,"in Proc. USENIX Symp. Oper. Syst. Des. Implementation, 2008,pp. 1–14.
[17] M. Zaharia, D. Borthakur, J. SenSarma, K. Elmeleegy, S. Shenker,and I. Stoica, "Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in Proc. Eur. Conf.Comput. Syst., 2010, pp. 265–278.
[18] M. Zaharia, A. Konwinski, A. D. Joseph, R. H. Katz, and I. Stoica, "Improving MapReduce performance in heterogeneous environments, "in Proc. USENIX Symp. Oper. Syst. Des. Implementation,2008, vol. 8, pp. 29–42.