www.ijiarec.com

ISSN:2348-2079

Volume-7 Issue-2

International Journal of Intellectual Advancements and Research in Engineering Computations

Designing a dynamic task scheduler in map reduce for hadoop framework

D.Radhika, C.Shalini, T.Uma Maheswari, S.Pooja, G.Subatharani

[1]Assistant Professor, Dept of Computer Science Engineering, Vivekanandha College of Engineering for Women, Tamilnadu, India

[2]U.G Scholar, Dept of computer Science Engineering, Vivekanandha College of Engineering for Women, Tamilnadu, India

ABSTRACT

Network bandwidth is a scarce resource in big data environments, so data locality is a fundamental problem for data-parallel frameworks such as Hadoop and Spark. Existing approaches solve this problem by scheduling computational tasks near the input data and considering the server's free time, data placements, and data transfer costs. However, such approaches usually set identical values for data transfer costs, even though a multicore server's data transfer cost increases with the number of data-remote tasks. Eventually, this hampers data-processing time, by minimizing it ineffectively. As a solution, we propose DynDL (Dynamic Data Locality), a novel data-locality-aware task-scheduling model that handles dynamic data transfer costs for multicore servers.

Keywords: Bigdata, Mapreduce, Hadoop.

INTRODUCTION

Big Data according to McKinsey refers to "datasets whose size are beyond the ability of typical database software tools to capture, store, manage and analyze". Big Data has become very popular in Information Technology, where data is becoming complex and growing day by day. According to IBM Big Data consists of three attributes: 1. Variety: It means that the generated data are different in type. It generally refers to structured, semi-structured or unstructured data in large amount. 2. Volume: It concerns the large quantities of data that is generated continuously; they might reach hundreds or thousands of terabytes. 3. Velocity: Where processing and analyzing data must be done in a fast manner to extract value of data in appropriate time.

Map reduce and hadoop

Map reduce Model

The MapReduce programming model consists of two functions: Map and Reduce. Input data is divided into fixed sized blocks and fed into parallel Map tasks. Then Map task process the data chunks and produces the intermediate output in the form of key-value pair tuples. These tuples are shuffled across different reduce nodes based on key values. Each Reduce task performs three steps:

- Copy - the output of map task is copied to reducer nodes, sort - the collection of outputs is sorted based on their key values and reduce - aggregation is applied to the data.

The Map Reduce architecture consists of:

- One master (Jobtracker) and
- Many workers or slaves (Task trackers)

The Job Tracker receives job submitted from user, divides it into map and reduce tasks. Then it assigns the tasks to Task Trackers, monitors the

Author for correspondence:
Dept of Computer Science Engineering, Vivekanandha College of Engineering for Women, Tamilnadu, India

progress and finally after the completion of the entire job, Job Tracker informs the status of the job to the user. Each Task Tracker has a fixed number of map and reduce task slots which helps to determine how many map and reduce tasks it can run at a time.

Scheduling in hadoop

Hadoop is a multi-user data warehouse which is used to support a variety of different types of processing jobs, with a pluggable scheduler framework providing greater control. The main objective of scheduling is to maximize throughput, minimize the completion time; overhead and available resources must be balanced by allocating jobs to processors. Hadoop uses a default scheduler i.e. FIFO. The jobs submitted earlier gets preference over the jobs submitted later. The Job Tracker pulls oldest jobs first from the job queue. This scheduler is mostly used when execution order of job is not important. Capacity Scheduler is developed by Yahoo that allows multiple tenants to securely share a large cluster. It supports hierarchical queues to ensure that resources are shared among the sub-queues of an organisation before other queues are allowed to use those resources. Fair Scheduler developed by Facebook, the core idea behind fair scheduler is to assign resources to each job such that an average each job gets equal share of available resources. Delay scheduling has an objective to address the conflict between locality and fairness. Delay scheduling temporarily relaxes fairness to improve locality by allowing jobs to wait for scheduling on a node with local data. In this, if data for the task is not present, then a task tracker waits for some time. If there is any request from local node for task assigning, then the scheduler will check the size of the job; if the job is too short, then he scheduler will skip that job and look for any subsequent jobs available to run. Dynamic Proportional Scheduling provides more job sharing and prioritization that result in increasing shares of cluster resources and more differentiation in service levels of different jobs. This improves response time for multi-user Hadoop environments.

task scheduling technology of hadoop

Hadoop implements the Google's MapReduce programming model. MapReduce is a distributed computing model as well as the core technology in Hadoop, which has been widely used into program distributed parallel application. Map and Reduce are two important concepts of MapReduce. Map, written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key I and passes them to the Reduce function.

Reduce task

The Reduce function, also written by the user, accepts an intermediate key I and a set of values for that key. It merges these values to form a possibly smaller set of values. Typically just zero or one output value is produced per Reduce invocation. The intermediate values are supplied to the user's reduce function via an iterator. This allows us to handle lists of values that are too large to fit in memory. Map Reduce is actually the process of "task decomposition and collect the results"

.

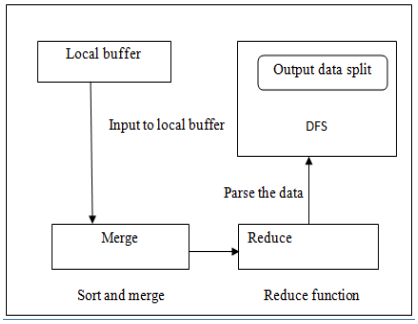

Figure 1.Execution of reduce task

Map Reduce architecture of hadoop

Map decomposes the job into a number of tasks, Reduce sums up the result that multi tasks, and then gets the final results. Besides, the following terms of Hadoop MapReduce are ofter used job. A job refers to every computing request for a user Task. A task refers to the split small parts of a job executing in a server; JobTracker. A JobTracker refers to the Server receives users' jobs. It is also responsible for the various tasks allocation and the management of all tasks servers TaskTracker. A TaskTracker is responsible for executing specific tasks Slot. It is responsible for executing a specific task. A task server may have multiple slots.

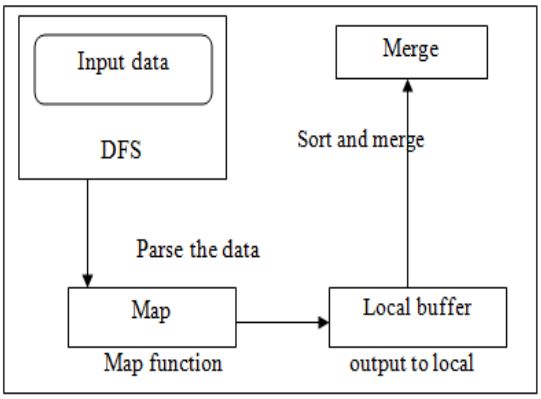

Figure 2.Mapreduce architecture of hadoop

Problem formulation

We use a multiple-input-single-output fuzzy model to predict a job's execution progress based on its input size and resource allocation in each control interval. The fuzzy model is often used to capture the complex relationship between resource allocations and a job's fine-grained execution progress [14]. However, a job's progress can be affected by many factors. First, job progress is not uniform at different execution phases, e.g., map and reduce phases. Second, even within the same phase, data skew among tasks leads to different task execution speed at different intervals. Finally, co-running jobs may unpredictably interfere with a job's execution, making the mapping of resource to job progress variable. Therefore, we design an online self-adaptive fuzzy model based on real-time measurements of job progress.

Fuzzy Model: The job j execution progress in the control interval t is represented as the input-output NARX type (Nonlinear Auto Regressive model with eXogenous inputs),

$y_j(t) = R_j(u(t), d_j, \xi(t))$.

R is the relationship between the input variables and the output variable. The input variables are the current resource allocation u (t), the job input size dj , and the regression vector ξ(t). Here, resource allocation, u(t)=[um(t), ur(t)], includes both map resource allocation um(t) and reduce resource allocation ur(t). The regression vector ξ(t) contains a number of lagged outputs and inputs of the previous control periods. It is represented as

$\xi(t) = [(y(t-1), y(t-2), \cdots, y(t-ny)), (u(t), u(t-1), \cdots, u(t-nu))]^T$

where ny and nu are the number of lagged values for outputs and inputs, respectively. Let ρ denote the number of elements in the regression vector ξ(t), that is,

$\rho = ny + nu$.

R is the rule-based fuzzy model that consists of Takagi-Sugeno rules. A rule Rj is represented as

Problem identification

Existing data-parallel frameworks such as Hadoop and Spark typically use greedy approaches to handle online scheduling. For example, Hadoop's default scheduler tries its best to find a data-local task for each idle core. However, this heuristic and its variations such as delay scheduling do not take the multicore servers into

account, neglecting the fact that data-remote tasks on the same computer compete for limited network bandwidth. The data transfer is dynamic and the scheduling instance is large, it is difficult to design an effective scheduler that has a low scheduling latency.

Proposed Algorithm

FCFS scheduling algorithm

First come, first served (FCFS) is an operating system process scheduling algorithm and a network routing management mechanism that automatically executes queued requests and processes by the order of their arrival. With first come, first served, what comes first is handled first; the next request in line will be executed once the one before it is complete. FCFS is also known as first-in, first-out (FIFO) and first come, first choice (FCFC)

Minmin algorithm

Min-Min scheduling is based on the concept of assigning a task having minimum completion time (MCT) first for execution on the resource, which has the minimum completion time (fastest resource). This algorithm is divided into two steps. In first step, expected completion time of each task in the metatask is calculated on each resource. In second step, the task with minimum expected completion time is selected and assigned to the corresponding resource, and then the selected task is removed from the metatask. This process repeats until all the tasks in metatask get mapped.

Min max algorithm

This algorithm overcomes the drawback of the min-min algorithm (larger makespan when numbers of large size tasks are greater than the small size task) [14]. Max-Min algorithm performs the same steps as the Min-Min algorithm but the main difference comes in the second phase, where a task ti is selected which has the maximum completion time instead of minimum completion time as in min-min and assigned to resource Rj 15 , which gives the minimum completion time. Hence it named as the Max-Min algorithm. This process is repeated until the metatask get empty or all the tasks are mapped.

Particle swarm optimization algorithm

Particle swarm optimization (PSO) based heuristic to schedule applications to cloud resources that takes into account both computation cost and data transmission cost. It is used for workflow application by varying its computation and communication costs. The experimental results show that PSO can achieve cost savings and good distribution of workload onto resources. PSO approach can solve the task scheduling problems. Therefore, we list other approaches to solve scheduling problems such as GA, Simulated annealing, tabu search, and ant colony. The work studied the comparison of particle swarm optimization and the genetic algorithm in the improvement of power system and stability

CONCLUSION

Task scheduling is one of the most important issues in big data analytics. In this paper, we propose a genetic algorithm-based approach, which uses a performance estimation module we put forward, for obtaining optimized jobs scheduling scheme that have the optimized time and cost consumption. The optimized solutions can be used to enable effective scheduling strategies, and then in the actual running, system can make use of the chosen scheduling scheme to execute data processing jobs. The whole process is evaluated and the results show that our approach is feasible with acceptable performance and accuracy. Due to the limitations in our performance estimation module, currently, the evaluation is simplified and in the future, the approach will be extended to be more complete and precise.

REFERENCES

[1]. A Hadoop, Apache Software Foundation (2016).http://hadoop.apache.org
[2]. J Dean, S Ghemawat, MapReduce: simplified data processing on largeclusters. Commun. ACM. 51(1), 107–113 (2008)
[3]. M Mitchell, An introduction to genetic algorithms. Journal of Computing Sciences in Colleges. 20(1), 2004.
[4]. A Konak, DW Coit, AE Smith, Multi-objective optimization using genetic algorithms: a tutorial. Reliab. Eng. Syst. Saf. 91(9), 2006, 992–1007.
[5]. K Deb, An introduction to genetic algorithms. Sadhana. 24(4–5), 1999, 293–315.
[6]. SM Thede, An introduction to genetic algorithms. J. Comput. Sci. Coll. 20(1), 2004, 115–123.
[7]. E Amazon, Amazon elastic compute cloud (amazon ec2). Amazon Elastic Compute Cloud (Amazon EC2) (2015). https://aws.amazon.com/ec2/
[8]. J Berli ´ nska, M Drozdowski, Scheduling divisible mapreduce computations. J. Parallel Distrib. Comput. 71(3), 2011, 450–459.
[9]. T Nykiel, M Potamias, C Mishra, G Kollios, N Koudas, MRShare: sharing across multiple queries in MapReduce. Proc. VLDB Endowment. 3(1–2), 2010, 494–505.
[10]. M Zaharia, A Konwinski, A Joseph, R Katz, I Stoica, in Proceedings of OSDI 201. Improving MapReduce performance in heterogeneous environments, 2008, 29–42.
[11]. L Xu, in Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International. MapReduce framework optimization via performance modeling (Shanghai, China, 2012. 2506–2509
[12]. J Han, M Ishii, H Makino, in Computer Science and Information Technology (CSIT), 2013 5th International Conference on. A Hadoop performance model for multi-rack clusters, (Amman, Jordan, 2013), pp. 265–274
[13]. S Krishan Veer, R Zahid, in Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in. A GA based job scheduling strategy for computational grid, Ghaziabad, India, 2015. 29–34
[14]. A Syed Hasan, R Kamran, in 2015 International Conference on Open Source Systems & Technologies (ICOSST). Cloud task scheduling using nature inspired meta-heuristic algorithm, 2 (IEEE, 2015), 158–164
[15]. K Shvachko, H Kuang, S Radia, R Chansler, in 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). The hadoop distributed file system, (Nevada State, Molossia, 2010, 1–10.
[16]. T White, Hadoop: The definitive guide. (O'Reilly Media, 2015)
[17]. Bonnie,Bonnie++(2012). http://sourceforge.net/projects/bonnie/. Accessed 19, 2015.
[18]. .Netperf,Netperf(2012). http://www.netperf.org/netperf/.Accessed 19, 2015.
[19]. . DY Chen, TR Chuang, SC Tsai, JGAP: a Java-based graph algorithms platform. Softw. Pract. Experience. 31(7), 2001, 615–635.
[20]. Site, site (2015). https://cs.gmu.edu/~eclab/projects/ecj/. Accessed 19, 2015.
[21]. JJ Durillo, AJ Nebro, jMetal: a Java framework for multi-objective optimization. Adv. Eng. Softw. 42(10), 2011, 760–771.
[22]. W Zhang, KM Hansen, in Engineering of Complex Computer Systems, 2009 14th IEEE International Conference on, IEEE. An evaluation of the NSGA-II and MOCell genetic algorithms for self-management planning in a pervasive service middleware, 2009, 192–201.