Postprint: Adaptive Scheduling Algorithm for Hadoop Node Capability in Heterogeneous Resource Environments

Authors: Shen Xueli, Sheng Fangyan

Date: 2018-12-13T00:00:00+00:00

Abstract

To address the deficiencies of existing scheduling and allocation methods inherent in Hadoop clusters operating in heterogeneous resource environments, this paper proposes a node capacity adaptive scheduling algorithm designated as NCAS (node capacity adaptive scheduling). Initially, the NCAS algorithm computes scheduling factors derived from node performance metrics and task characteristics; subsequently, these factors determine the appropriate data volume and task slot allocation for each node; ultimately, data and tasks are preferentially distributed to fast nodes while being conservatively allocated to slow nodes. Experimental results demonstrate that, in comparison with traditional scheduling algorithms, NCAS substantially reduces the number of launched backup tasks, significantly shortens job completion time, and enhances overall task execution efficiency.

Full Text

Preamble

Vol. 37 No. 2
Application Research of Computers
ChinaXiv Partner Journal

Hadoop Node Capability Adaptive Scheduling Algorithm in Heterogeneous Resource Environments

Shen Xueli , Sheng Fangyan
(. School of Electronics & Information Engineering; . School of Graduate Studies, Liaoning Technical University, Huludao, Liaoning 125105, China)

Abstract: To address the shortcomings of current Hadoop clusters' inherent scheduling and distribution methods in heterogeneous resource environments,

this paper proposes a Node Capacity Adaptive Scheduling (NCAS) algorithm based on node capability. First, the NCAS algorithm calculates a scheduling factor based on node performance and task characteristics. Then, this scheduling factor determines the amount of data and number of task slots each node should be assigned. Finally, the algorithm distributes more data and tasks to fast nodes while assigning less to slow nodes. Experimental results demonstrate that compared with traditional scheduling algorithms, NCAS significantly reduces the number of speculative tasks launched, markedly decreases job completion time, and improves task execution efficiency.

Keywords: Hadoop; heterogeneous resources; node capability; adaptive

Classification: TP301.6
doi: 10.19734/j.issn.1001-3695.2018.08.0554

Hadoop Node Capability Adaptive Scheduling Algorithm in Heterogeneous Resource Environments

Shen Xueli, Sheng Fangyan
(a. School of Electronics & Information Engineering, b. School of Graduate Studies, Liaoning Technical University, Huludao Liaoning 125105, China)

Abstract: In order to solve the shortcomings of current Hadoop cluster inherent scheduling distributed methods in heterogeneous resource environments, this paper proposed an adaptive scheduling algorithm NCAS (Node Capacity Adaptive Scheduling) based on the node capability. Firstly, NCAS algorithm calculated the scheduling factor based on node performance and task characteristics; Secondly, the scheduling factor determined the amount of data and the number of task slot that each node should be assigned; Finally, NCAS algorithm dispatched data and tasks more into fast nodes and less into slow nodes. Experimental results show that, compared with the traditional scheduling algorithm, NCAS algorithm can greatly reduce the number of speculative tasks, significantly reduce the job completion time. It also can improve the task execution efficiency.

Key words: Hadoop; Heterogeneous resource; node capability; adaptive

0 Introduction

Hadoop [**?**], as an open-source solution for cloud storage and cloud computing, has gained increasingly widespread adoption in both academia and industry. Job scheduling algorithms and task scheduling algorithms play a crucial role in Hadoop performance [**?**, **?**].

Currently, several common Hadoop job scheduling algorithms include the First-In-First-Out (FIFO) scheduler [**?**], the Fair Scheduler [**?**], and the Capacity Scheduler [**?**]. These Hadoop job scheduling algorithms are designed for homogeneous cluster resources and perform inefficiently in heterogeneous resource environments [**?**]. To address the limitations of Hadoop' s default speculative task scheduling algorithm (Speculative Task) [**?**] in heterogeneous environments,

Zaharia et al. [?] proposed the LATE scheduler, which improves Hadoop task execution efficiency by launching backup tasks for tasks with the longest remaining completion time, though the estimation of task remaining completion time needs improvement. Tao Yongcai et al. [?] proposed an adaptive MapReduce scheduler that addresses load distribution imbalance in heterogeneous resource environments to some extent, but determining the expected task completion time is challenging. Zheng Xiaowei et al. [?] proposed a task adaptive scheduling method based on node capability that fully considers the heterogeneity of node resources in clusters, but it can easily cause frequent data transmission between nodes, consuming network bandwidth and reducing system performance.

To solve the problem of low operational efficiency that still exists in Hadoop scheduling algorithms in heterogeneous environments, this paper proposes the Node Capability Adaptive Scheduling (NCAS) algorithm. This algorithm calculates a scheduling factor representing each node' s computing capacity and uses this factor to determine the number of task slots and data volume allocated to fast or slow nodes, thereby improving overall system execution efficiency.

1 Research Background

The Hadoop core consists of two main components: the distributed file system HDFS (Hadoop Distributed File System) and the MapReduce implementation system. The Hadoop cluster architecture comprises one master node and a group of slave nodes. As shown in [Figure 1: see original paper], in HDFS, the master and slave nodes run a NameNode and DataNode process, respectively. The NameNode stores metadata in HDFS and is responsible for responding to user access requests to the HDFS file system. The DataNode is responsible for storing data, executing NameNode commands, and creating, replicating, and deleting data blocks. Clients can directly perform read and write operations on DataNodes [?].

After receiving a file, HDFS splits it into equal-sized data blocks [?] and replicates them three times, storing them on different nodes. [Figure 2: see original paper] illustrates the MapReduce framework. User-submitted jobs are divided into several map tasks and reduce tasks [?]. Each map task processes one data block and outputs corresponding key-value pairs, while reduce tasks process these key-value pairs to produce the final output results [?]. Each slave node has a certain number of map task slots and reduce task slots, with one slot executing one corresponding type of task. Slot configuration plays a vital role in Hadoop performance [?]. To manage job execution, the master and slave nodes run a JobTracker and TaskTracker process, respectively. The JobTracker is responsible for scheduling map and reduce tasks to appropriate TaskTrackers, monitoring task execution, and managing fault tolerance. The TaskTracker executes assigned map and reduce tasks, collects and stores the results, and reports the node' s current status and resource utilization to the JobTracker through periodic heartbeats. The system launches backup tasks for straggler tasks and reschedules failed tasks.

2 NCAS Scheduling Algorithm

Upon receiving a user file, the NCAS algorithm first splits it, then distributes the resulting data blocks to HDFS for storage, followed by Map task scheduling and Reduce task scheduling. To ensure clarity, we first explain two relevant concepts before presenting the adaptive data distribution algorithm and adaptive task scheduling algorithm that constitute the NCAS algorithm.

2.1.1 Scheduling Factor

The scheduling factor relates to two factors: node performance and task characteristics, representing the strength of a node' s computing capacity. Node performance can be determined by hardware parameters, where larger hardware parameters indicate better node performance. In heterogeneous environments, the same slot exhibits different processing speeds when handling different jobs [**?**]. We define task characteristics to represent the different performance exhibited by the same node when executing different tasks.

Hadoop clusters primarily process three types of jobs: CPU-bound, I/O-bound, and a hybrid of the two (class sway) [**?**]. This paper selects CPU-bound and I/O-bound tasks, using the time spent by nodes executing different types of tasks previously as parameters to calculate task characteristic values. As tasks continue to execute, task characteristic parameters may change, so we set a certain period for updating and recalculating these parameters. This paper uses scripts to read and calculate various node information values, storing the results in files.

For a resource set $R = \{r_1, r_2, \dots, r_m\}$, let N_{r_j} and T_{t_j} denote the node performance and task characteristics of node r_j $(1 \le j \le m)$, respectively, and W_j be its scheduling factor. The CPU frequency, memory capacity, network bandwidth, and maximum disk read/write rate of node r_j are ttc_j, ttm_j, ttn_j, and ttd_j, respectively, while the cluster average CPU frequency, memory capacity, network bandwidth, and disk read/write rate are $avgc$, $avgm$, $avgn$, and $avgd$, respectively. Node performance is calculated as:

$$N_{r_j} = w_1 \cdot \frac{ttc_j}{avgc} + w_2 \cdot \frac{ttm_j}{avgm} + w_3 \cdot \frac{ttn_j}{avgn} + w_4 \cdot \frac{ttd_j}{avgd}$$

where w_1, w_2, w_3, and w_4 are the weight values of CPU frequency, memory capacity, network bandwidth, and maximum disk read/write rate in node performance, respectively, with the sum of weights satisfying:

$$\sum_{i=1}^{4} w_i = 1$$

Historical time records of completed tasks can be found in local user logs. The average completion times for node r_j running CPU-bound and I/O-bound Map

tasks are $ttcm_j$ and $ttiom_j$, respectively, while the average completion times for CPU-bound and I/O-bound Reduce tasks are $ttcr_j$ and $ttior_j$, respectively. The cluster average completion times for CPU-bound and I/O-bound Map tasks are $avgcm$ and $avgiom$, respectively, and for CPU-bound and I/O-bound Reduce tasks are $avgcr$ and $avgior$, respectively. The task characteristic T_{t_j} is calculated as:

$$T_{t_j} = \begin{cases} \frac{ttcm_j}{avgcm} + \frac{ttiom_j}{avgiom}, & \text{Map phase} \\ \frac{ttcr_j}{avgcr} + \frac{ttior_j}{avgior}, & \text{Reduce phase} \end{cases}$$

The scheduling factor W_j is calculated as:

$$W_j = w_5 \cdot N_{r_j} + w_6 \cdot T_{t_j}$$

where w_5 and w_6 are the weight values of node performance and task characteristics in the scheduling factor, respectively, with the sum of weights satisfying:

$$w_5 + w_6 = 1$$

2.1.2 Node Heterogeneity

We use the heterogeneity factor HF to represent the computational capability heterogeneity of cluster resources. Define HF as the average heterogeneity factor of cluster resource set $R = \{r_1, r_2, \ldots, r_m\}$. Since different jobs utilize CPU and I/O resources to varying degrees, HF differs after running different types of tasks. The expression for HF is:

$$HF = D(X) = E[(X - E(X))^2]$$

where x_j is the scheduling factor of r_j, as shown in equation (7):

$$x_j = W_j$$

$E(X)$ is the mean of $X = \{x_1, x_2, \ldots, x_m\}$, calculated through equation (8):

$$E(X) = \sum_{j=1}^{m} P_j \cdot x_j$$

where P_j is the probability of node r_j being assigned a task. For a cluster with m nodes, P_j is $1/m$, and $D(X)$ is the mean square deviation of X.

2.2 Adaptive Data Distribution Algorithm

In heterogeneous resource environments, node computing capabilities vary significantly. When each node receives the same amount of data, fast nodes that quickly process their local data must handle data from slow nodes, incurring substantial costs for transmitting large amounts of data between distant fast and slow nodes. To avoid this situation [**?**], we ensure that the amount of data stored on each node is proportional to its computing capacity, which significantly reduces the volume of data requiring remote transmission [**?**]. The adaptive data distribution algorithm allocates large amounts of data to fast nodes and small amounts to slow nodes based on the scheduling factor, avoiding remote transmission of large data volumes and conserving network bandwidth.

shows the experimental resource node configuration. In heterogeneous environments, the data allocation amount $ND(r_j)$ for node r_j is calculated as:

$$ND(r_j) = \frac{x_j}{E(X)} \cdot ND$$

where ND is the user job data volume (here T_{t_j} in x_j takes the Map phase value).

The adaptive DFS data distribution algorithm is as follows:

a) For each node r_j in resource node set R, calculate W_j through equations (1)-(5) and set $x_j = W_j$;

b) Calculate $E(X)$ through equation (8), and for each node r_j in R, calculate HF through equation (6);

c) Compare HF with the set heterogeneity factor threshold $HF_{threshold}$. If $HF < HF_{threshold}$, the cluster resources are homogeneous and each node equally shares ND; otherwise, the cluster resources are heterogeneous, and each node allocates data volume according to the result calculated by equation (9).

2.3 Adaptive Task Scheduling Algorithm

Current scheduling methods produce uneven load distribution when facing heterogeneous clusters. To balance the load to the greatest extent possible, we propose the adaptive task scheduling algorithm, which calculates task slot numbers matching each node' s computing capacity based on the scheduling factor. The adaptive task scheduling algorithm consists of two parts. We first introduce the adaptive Map task scheduling:

The test programs selected for experiments are grep-count (CPU-bound), TeraSort (I/O-bound), and WordCount (class sway). Based on cluster operation conditions, we set a certain period to update task characteristic parameters and

retrieve corrections for task characteristics from files. The cluster runs grep-count and TeraSort programs 10 times each under the speculative task scheduling algorithm, with each program processing 3.0GB of data. By recording the map and reduce task execution information during operation and performing calculations, we obtain initial task characteristic values. The average historical time data for each node is recorded in .

In heterogeneous environments, the number of task slots $N_{slot}(r_j)$ allocated to node r_j is calculated as:

$$N_{slot}(r_j) = \frac{x_j}{E(X)} \cdot N_{slot}$$

where N_{slot} is the system-preset evenly distributed number of task slots (here T_{t_j} in x_j takes the Map phase value).

The adaptive Map task scheduling algorithm is as follows:

a) For each node r_j in resource node set R, calculate W_j through equations (1)-(5) and set $x_j = W_j$;

b) Calculate $E(X)$ through equation (8), and for each node r_j in R, calculate HF through equation (6);

c) Compare HF with the set heterogeneity factor threshold $HF_{threshold}$. If $HF < HF_{threshold}$, the cluster resources are homogeneous and each node' s task slots are set to N_{slot}; otherwise, the cluster resources are heterogeneous, and each node sets its task slots to $N_{slot}(j)$ calculated through equation (10);

d) If a TaskTracker has idle map task slots, the JobTracker assigns unscheduled map tasks to it;

e) TaskTrackers periodically send heartbeat signals to the JobTracker to report node operation status;

f) When a map task completes, the TaskTracker sends heartbeat information to the JobTracker, which then sets the task slots to $N_{slot}(r_j)$ calculated through equation (10).

Adaptive reduce task scheduling is similar to adaptive map task scheduling, with the difference that T_{t_j} in x_j takes the Reduce phase value. To avoid redundancy, we omit detailed description here.

3 Validation and Analysis

To validate the algorithm, we built a cluster consisting of three racks. Each rack comprises computer nodes with different configurations, with switches connect-

ing nodes within and between racks. The parameters of each computer node are shown in .

[Figure 3: see original paper] compares the number of backup tasks launched by Hadoop' s default speculative task scheduling algorithm and the NCAS algorithm when running WordCount with job data volumes of 2GB, 4GB, and 8GB. The results show that NCAS reduces the number of backup tasks by an average of 70.0% compared to the speculative task scheduling algorithm. The analysis is as follows: existing Hadoop defaults assume homogeneous cluster resources, evenly distributing task slots and data volumes among nodes, causing many tasks on slow nodes to fall behind and triggering numerous backup tasks. In contrast, NCAS determines task slots and data volumes matching each node' s computing capacity based on the scheduling factor, making the progress across nodes relatively balanced during cluster operation, reducing backup task launches and optimizing cluster performance.

The experimental program data volume is set to 4GB. The Hadoop cluster runs under four algorithms: Hadoop scheduler without speculative task scheduling, speculative task scheduling algorithm, LATE algorithm, and NCAS algorithm. Each type of task program executes three times under each algorithm, and the average values are obtained. The runtime information is shown in [Figure 4: see original paper].

[Figure 4: see original paper] reveals that for all test program types, the system using the Hadoop scheduler without speculative task scheduling takes significantly longer than the other three algorithms. Compared to the speculative task scheduling algorithm, NCAS reduces runtime by an average of 32.2%; compared to the LATE algorithm, NCAS reduces runtime by an average of 16.6%. The main reasons are: the speculative task scheduling algorithm assumes homogeneous resources in Hadoop' s default cluster, causing slow nodes to launch numerous backup tasks; the LATE algorithm provides more accurate identification of straggler tasks but does not fundamentally solve the problems encountered by the speculative task scheduling algorithm; NCAS determines task slots and data volumes proportional to each node' s computing capacity, significantly reducing runtime.

4 Conclusion

This paper studied Hadoop' s main architecture and operational principles, proposing the Node Capability Adaptive Scheduling (NCAS) algorithm for heterogeneous resource environments. The algorithm calculates a scheduling factor based on relevant node computing capacity metrics and uses the proportion of this factor in the total cluster scheduling factor to determine the data volume and task slots each node should receive, allocating more to fast nodes and less to slow nodes to maintain relatively balanced cluster load. Experimental results demonstrate that NCAS improves Hadoop' s resource utilization and system performance in heterogeneous resource environments.

Future work will further refine and optimize the algorithm regarding weight value settings in node performance and scheduling factors, potential difficulties in estimating task remaining completion time, and possible excessive network bandwidth consumption that may reduce system performance. As Hadoop clusters continue to evolve and factors affecting node performance constantly change, we will conduct more in-depth research on task scheduling algorithms.

References

[1] Apache Hadoop. What is Apache Hadoop [EB/OL]. (2018-06-13) [2018-07-03]. http://hadoop.apache.org/
[2] Han Jiazhen, Yuan Zhengheng, Han Yiheng, et al. An adaptive scheduling and load balancing in computing clusters [J]. Journal of Network and Computer Applications, 2017, 83(1): 213-220.
[3] Liu Yyu, Zhang Changjie, Li Bo, et al. DeMS: a hybrid scheme of task scheduling and load balancing in computing clusters [J]. Journal of Network and Computer Applications, 2017, 83(1): 213-220.
[4] Wang Jiaqi. Research on scheduling algorithms and improvement strategies under Hadoop platform [D]. Beijing: Beijing University of Posts and Telecommunications, 2016.
[5] The Apache Software Foundation. Fair Scheduler [EB/OL]. (2013) [2018-07-03]. https://hadoop.apache.org/docs/r1.2.1/fair_{scheduler}.html.
[6] Zhang Cong. Research on Hadoop distributed system scheduling algorithms [D]. Beijing: Beijing Jiaotong University, 2016.
[7] Liu Ying, Luo Xingyu, Wang Ning, et al. Task scheduling scheme for heterogeneous Hadoop cloud platform based on task progress awareness [J]. Application Research of Computers, 2017, 34(10): 3139-3143. (Liu Ying, Luo Xingyu, Wang Ning, et al. Task scheduling scheme for heterogeneous Hadoop cloud platform based on task progress awareness [J]. Application Research of Computers, 2017, 34(10): 3139-3143.)
[8] Liu Kui, Liu Xiangdong, Ma Baolai, et al. Research on speculative Hadoop task scheduling algorithm based on data locality [J]. Application Research of Computers, 2014, 31(1): 182-187. (Liu Kui, Liu Xiangdong, Ma Baolai, et al. Research on speculative Hadoop task scheduling algorithm based on data locality [J]. Application Research of Computers, 2014, 31(1): 182-187.)
[9] Zaharia M, Konwinski A, Joseph A D, et al. Improving MapReduce performance in heterogeneous environments [C]// Proc of the 8th USENIX Conference on Operating Systems Design and Implementation. New York: ACM Press, 2008: 29-42.
[10] Tao Yongcai, Shi Lei. Optimization of MapReduce performance in heterogeneous environments [J]. Journal of Chinese Computer Systems, 2013, 34(2): 287-292. (Tao Yongcai, Shi Lei. Optimization of MapReduce performance in heterogeneous environments [J]. Journal of Chinese Computer Systems, 2013, 34(2): 287-292.)
[11] Zheng Xiaowei, Xiang Ming, Zhang Dawei, et al. Hadoop cluster task adaptive scheduling method based on node capability [J]. Journal of Computer Re-

search and Development, 2014, 51(3): 618-626. (Zheng Xiaowei, Xiang Ming, Zhang Dawei, et al. Hadoop cluster task adaptive scheduling method based on node capability [J]. Journal of Computer Research and Development, 2014, 51(3): 618-626.)
[12] Chen Wenlong. Research on job scheduling algorithms based on Hadoop [D]. Nanjing: Nanjing University of Science and Technology, 2015. (Chen Wenlong. Research on job scheduling algorithms based on Hadoop [D]. Nanjing: Nanjing University of Science and Technology, 2015.)
[13] Guo Zhenhua, Pierce M, Fox G, et al. Automatic task re-organization in MapReduce [C]// Proc of IEEE International Conference on Cluster Computing. Piscataway, NJ: IEEE Press, 2011: 335-343.
[14] Chen Quan, Zhang Daqiang, Guo Minyi, et al. SAMR: a self-adaptive MapReduce scheduling algorithm in heterogeneous environment [C]// Proc of the 10th IEEE International Conference on Computer and Information Technology. Piscataway, NJ: IEEE Press, 2010: 2736-2743.
[15] Lim B, Shim Y, Chung Y D. 2PTS: a two-phase task scheduling algorithm for MapReduce [J]. IEICE Trans on Information & Systems, 2016, 99 (9): 2377-2380.
[16] Wang Jiayin, Yao Yi, Mao Ying, et al. FRESH: fair and efficient slot configuration and scheduling for Hadoop clusters [C]// Proc of the 7th IEEE International Conference on Cloud Computing. Piscataway, NJ: IEEE Press, 2014: 761-768.
[17] Tang Zhuo, Liu Min, Ammar A, et al. An optimized MapReduce workflow scheduling algorithm for heterogeneous computing [J]. Journal of Supercomputing, 2014, 72(6): 1-21.
[18] Tian Chao, Zhou Haojie, He Yongqiang, et al. A dynamic MapReduce scheduler for heterogeneous workloads [C]// Proc of the 8th International Conference on Grid and Cooperative Computing. Piscataway, NJ: IEEE Press, 2009: 218-224.
[19] Lin Changhang, Guo Wenzhong, Chen Huangning. Data allocation strategy for Hadoop heterogeneous cluster node performance [J]. Journal of Chinese Computer Systems, 2015, 36(1): 83-88. (Lin Changhang, Guo Wenzhong, Chen Huangning. Data allocation strategy for Hadoop heterogeneous cluster node performance [J]. Journal of Chinese Computer Systems, 2015, 36(1): 83-88.)
[20] He Xiang, Li Renfa, Tang Zhuo. An improved mechanism based on MapReduce task scheduling in heterogeneous environments [J]. Application Research of Computers, 2013, 30(11): 3370-3373. (He Xiang, Li Renfa, Tang Zhuo. An improved mechanism based on MapReduce task scheduling in heterogeneous environments [J]. Application Research of Computers, 2013, 30(11): 3370-3373.)

Note: Figure translations are in progress. See original paper for figures.