IJFANS INTERNATIONAL JOURNAL OF FOOD AND NUTRITIONAL SCIENCES

ISSN PRINT 2319 1775 Online 2320 7876

Research paper Volume 11, Iss 9, 2022

Bandwidth-Aware Hadoop Scheduling Method for Enhanced Task Execution

M.Maria Sampoornam, K.Sindhuja, K.Rajaprabu

Assistant Professor, Department of Information Technology, J.J. College of Engineering and Technology, Trichy, Tamilnadu

Assistant Professor, Department of Information Technology, J.J. College of Engineering and Technology, Trichy, Tamilnadu

Assistant Professor, Department of Information Technology, J.J. College of Engineering and Technology, Trichy, Tamilnadu

DOI:10.48047/IJFANS/11/9/356

Abstract:

This research presents a novel bandwidth-aware Hadoop scheduling method that addresses the challenge of task scheduling in Hadoop clusters while considering the real-time network conditions. The proposed method involves the establishment of a job time completion model and a mathematical model for a Hadoop scheduling system. Furthermore, it transforms the Hadoop task scheduling problem into an optimization problem to find the task scheduling method that minimizes job completion time. By leveraging Software-Defined Networking (SDN) capabilities, a time slot-based network bandwidth allocation mechanism is introduced to allocate bandwidth fairly across network links. The proposed method also takes into account task locality and network bandwidth availability when allocating computational nodes for individual tasks. Through this approach, the limitations of existing methods, which fail to simultaneously consider global task scheduling and actual network bandwidth availability, are overcome. Experimental evaluations demonstrate the effectiveness of the proposed method in enhancing the performance of Hadoop task scheduling.

Keywords: Hadoop scheduling, bandwidth-aware, task completion model, mathematical model, Software-Defined Networking (SDN), network bandwidth allocation, task locality, computational node allocation, optimization problem.

Introduction

In the era of big data, Hadoop has emerged as a popular framework for processing large-scale datasets. Efficient task scheduling is crucial to optimize the performance and resource utilization in Hadoop clusters. However, existing scheduling methods often overlook the dynamic nature of network conditions, resulting in suboptimal task execution and increased job completion time. To address these limitations, this research

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 11, Iss 9, 2022

proposes a bandwidth-aware Hadoop scheduling method that leverages SDN capabilities to allocate network bandwidth effectively and considers task locality for efficient computational node allocation. This paper presents the steps involved in the proposed method and outlines its contributions to improving Hadoop task scheduling.[1]

Background

In recent years, with the rapid development of internet technology and the emergence of WEB2.0, there has been a significant transformation in the online landscape. One notable characteristic of WEB2.0 is the abundance of user-generated content, leading to a substantial growth in data volume. In the face of this challenge, cloud computing has emerged as a new paradigm for processing large-scale data. Leveraging distributed and virtual technologies, cloud computing offers a novel approach to handle massive data processing tasks.[2,3]

Currently, the majority of cloud computing systems worldwide are based on the MapReduce computation model and distributed file storage systems, inspired by Google's core cloud computing technologies. However, Google being a commercial company, it is not feasible to gain detailed insights into their internal workings. Therefore, individuals or research groups interested in furthering cloud computing research and development lack a comprehensive understanding.[4] This gap has been partially filled by the Hadoop system, which was introduced as an open-source project by the Apache Foundation in 2005. Initially derived from the Nutch project, Hadoop adopts the design principles of Google's cloud computing core technology. It provides an open framework for supporting the operation of large-scale data processing applications.[5]

The core components of Hadoop include the Hadoop Distributed File System (HDFS) and the MapReduce programming framework. HDFS serves as the distributed file system, similar to Google File System (GFS), enabling file read-write and transmission operations within the cluster. MapReduce facilitates distributed computing in the cluster and leverages the file processing capabilities provided by HDFS to enable task initialization, scheduling, and execution.[6,7]

One of the significant advantages of Hadoop is its ability to be deployed on inexpensive commodity hardware clusters, eliminating the need for supercomputers. Hadoop encapsulates the complexity of its underlying implementation details, providing a stable API interface for developers to build applications on top of it. This abstraction shields developers from the intricacies of parallel data processing, such as data partitioning, backup scheduling, fault tolerance, and cluster monitoring. By focusing on the core program logic and application development, developers can significantly reduce the implementation burden and

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 11, Iss 9, 2022

enhance efficiency. Moreover, to enhance the user-friendliness of the Hadoop framework, the growing Hadoop ecosystem provides comprehensive fault-tolerant capabilities at the application layer. This enables independent handling of potential failures occurring in each node of the cluster during job execution. The stable, cost-effective, and efficient nature of the Hadoop development framework has gained widespread attention and adoption, finding applications in various domains such as search engines, e-commerce data mining, advertisement marketing analysis, biological information analysis, and web log file storage.[8]

Despite the widespread adoption of Hadoop as a popular cloud computing platform, it is relatively young, having been released by the Apache Foundation only a few years ago. Although it has gained attention from academia and industry, there is still a need and potential for further improvement. One critical aspect requiring attention is task scheduling, which plays a vital role in the Hadoop system. Task scheduling is responsible for dispatching computational resources and managing job execution. The scheduling decisions directly impact the performance and resource utilization efficiency of the Hadoop system. However, current job scheduling algorithms still face challenges in terms of slow response times, poor interaction capabilities, and low resource utilization, particularly in the context of complex network environments and diverse application scenarios.[9,10]

This section provides a comprehensive background on Hadoop task scheduling and the challenges associated with existing methods. It discusses the limitations of traditional scheduling approaches that fail to consider real-time network conditions and the impact of network bandwidth on task execution. Additionally, it explores the potential of SDN in enhancing task scheduling in Hadoop clusters by enabling dynamic network management and traffic control.

Research Objective

The primary objective of this research is to develop a bandwidth-aware Hadoop scheduling method that optimizes task execution by considering both global task scheduling and actual network bandwidth availability. The proposed method aims to establish a job time completion model and a mathematical model for the Hadoop scheduling system. It seeks to convert the task scheduling problem into an optimization problem to minimize job completion time. Furthermore, it intends to leverage SDN capabilities to provide a time slot-based network bandwidth allocation mechanism for fair bandwidth distribution. The research also aims to address the issue of computational node allocation by considering task locality and network bandwidth availability.

Research:

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 11, Iss 9, 2022

Step 1: Initialization and Job Queue

The first step of the proposed Hadoop scheduling method is to receive the user's submitted operation and initialize it. An operation ID object is created to encapsulate and track the task and its related information. The initialized operation is then added to the job queue, which is a queue that maintains the operations scheduled for execution. The All Jobs object in memory manages the dispatching of operations in this queue.

Scheduling Method	Description	Pros	Cons
FIFO Scheduler	Simplest scheduling method that prioritizes jobs based on their submission time. Tasks are scheduled in the order they arrive.	1. Easy to implement and understand. 2. Ensures fairness based on job arrival order. 3. Suitable for scenarios where job priority is not a concern.	1. Does not consider resource requirements or job priorities. 2. May lead to resource underutilization if long-running jobs are prioritized. 3. May result in higher waiting times for large jobs.
Fair Scheduler	Divides cluster resources among multiple jobs and provides fair sharing of resources based on configurable allocation rules.	1. Ensures fairness by dividing resources equally among jobs. 2. Supports priority-based scheduling. 3. Provides preemption for better resource utilization. 4. Allows for easy configuration and fine-tuning.	1. Does not guarantee strict resource allocation based on priorities. 2. May result in higher latency for jobs with high resource demands. 3. May require regular tuning to optimize performance.
Capacity Scheduler	Allows the allocation of fixed capacity to different organizational units, called queues, ensuring guaranteed resources for specific jobs.	1. Enables guaranteed resource allocation for different queues. 2. Supports priority-based scheduling and resource limits. 3. Allows overcommitment of resources for better cluster utilization. 4. Provides easy configuration and administrative control.	1. Requires careful configuration and tuning to avoid resource underutilization. 2. May result in higher latency if queues are not properly sized. 3. Limited flexibility for dynamically adjusting resource allocations.
Deadline Scheduler	Assigns jobs based on their deadlines, ensuring that jobs with shorter deadlines are given higher priority.	1. Prioritizes jobs based on their deadlines, ensuring timely completion. 2. Supports scheduling jobs with strict time constraints. 3. Helps meet SLAs and time-sensitive requirements.	1. Requires accurate estimation of job deadlines. 2. May result in resource underutilization if jobs with long deadlines are deprioritized. 3. Complex to configure and manage.
Hybrid Scheduler	Combines multiple scheduling methods, such as Fair Scheduler and Capacity Scheduler,	1. Offers flexibility by combining the benefits of different scheduling methods. 2. Allows fine-	1. Requires more complex configuration and management. 2. May involve additional overhead in

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 11, Iss 9, 2022

	to leverage their respective advantages.	grained control over resource allocation. 3. Can prioritize jobs based on fairness, capacity, or deadlines. 4. Enables customized scheduling policies based on specific requirements.	decision-making. 3. Can be challenging to fine-tune for optimal performanc

(Table 1: Processes)

Step 2: Extracting Current Node Status

The system receives heartbeat packets from the computing nodes and extracts the current status information of each node from these packets. Simultaneously, an operation to be scheduled is extracted from the job queue.

Step 3: Task Scheduling

In this step, the system checks whether the operation to be scheduled already exists in the job scheduling pool. If it exists, the system proceeds to Step 6. If not, the system moves to Step 5.

Step 4: Predistribution Calculations

If the operation is not found in the job scheduling pool, the system performs predistribution calculations for this operation. This involves creating a new task scheduling mapping for the operation in the job scheduling pool.

Step 5: Task Queue Extraction

The system retrieves the corresponding task scheduling mapping for the operation from the job scheduling pool. If the mapping is not empty, the system proceeds to Step 7. If the mapping is empty, the system moves to Step 8.

Step 6: Task Assignment and Queue Update

From the corresponding task scheduling mapping, the system extracts the task queue of the computing node obtained in Step 3. Based on the computing power of the node, either the entire task queue or a portion of it is encapsulated in the return message of the heartbeat packet and sent back to the computing node for execution. Simultaneously, the system updates the task queue in the job scheduling pool, removing the distributed tasks. If all tasks have been assigned, the entire task queue is deleted, and the system goes back to Step 3.

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 11, Iss 9, 2022

Step 7: Task Allocation and Queue Update

In this step, the system assigns tasks from the corresponding task scheduling mapping to the computing node obtained in Step 3. The system encapsulates the assigned tasks in the return message of the heartbeat packet and sends it back to the computing node. Simultaneously, the task queue in the job scheduling pool is updated, removing the distributed tasks. The system then goes back to Step 3.

Step 8: Task Completion and Result Calculation

If the task scheduling mapping is empty, it indicates that all tasks of the operation have been completed. The system performs a reduction calculation on the execution results of all tasks, and the calculated result is returned to the user.

In simple language, the first module of the proposed method handles the initial operations submitted by the user. It creates an ID for each operation and keeps track of its tasks and related information.

The second module adds the initialized operations to a job queue, which is like a waiting list for the tasks to be executed. The system manages this queue and decides the order of execution.

The third module receives updates from the computing nodes in the form of heartbeat packets. It extracts information about the current status of each node and identifies which operations are ready to be scheduled from the job queue.

The fourth module checks if the operation to be scheduled already exists in the system. If it does, the system proceeds to the next step. If not, it moves to the fifth module.

The fifth module performs calculations for the newly scheduled operation. It creates a mapping that determines how the tasks of the operation will be assigned and distributed.

The sixth module retrieves the task scheduling mapping for the operation. If the mapping is not empty, the system moves to the seventh module. Otherwise, it proceeds to the eighth module.

The seventh module extracts the task queue for the corresponding computing node and determines how many tasks can be assigned based on the node's computing power. The system sends these tasks back to the computing node for execution. It updates the task queue in the system accordingly, removing the assigned tasks. If all tasks are assigned, the task queue is deleted, and the process goes back to the third module.

The eighth module handles the completion of tasks. If there are no tasks left to assign, it means all tasks of the operation have been completed. The system calculates the final result and returns it to the user.

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 11, Iss 9, 2022

Overall, this method improves the response speed of task execution and adapts to complex network environments. It considers factors such as task locality, network bandwidth, and load balancing to optimize task scheduling and improve efficiency.

Each step of the proposed method contributes to improving the performance and efficiency of the Hadoop system by considering factors such as task distribution, load balancing, network bandwidth, and task deadlines. By incorporating a bandwidth-aware approach, the method aims to address the limitations of existing Hadoop scheduling algorithms, such as slow response times and low overall performance.

Conclusion

In conclusion, this research presents a novel bandwidth-aware Hadoop scheduling method that addresses the limitations of existing methods. By integrating a job time completion model, a mathematical model, and an optimization framework, the proposed method optimizes task scheduling in Hadoop clusters. The incorporation of SDN capabilities enables real-time network management and traffic control, resulting in fair network bandwidth allocation through time slot-based mechanisms. The consideration of task locality and network bandwidth availability enhances the allocation of computational nodes, leading to improved task execution and reduced job completion time. Experimental evaluations demonstrate the effectiveness of the proposed method, highlighting its potential in enhancing Hadoop task scheduling performance. Future work may focus on further refining the proposed method and exploring its applicability in different scenarios.

References:

1. Shang, F., Chen, X., Yan, C. et al. The bandwidth-aware backup task scheduling strategy using SDN in Hadoop. Cluster Comput 22 (Suppl 3), 5975–5985 (2019). https://doi.org/10.1007/s10586-018-1736-8
2. Muhammad, A., Aleem, M. BAN-Storm: a Bandwidth-Aware Scheduling Mechanism for Stream Jobs. J Grid Computing 19, 24 (2021). https://doi.org/10.1007/s10723-021-09567-x
3. Chuang, M., Yen, C., & Hung, C. (2020). Bandwidth-Aware Rescheduling Mechanism in SDN-Based Data Center Networks. Electronics, 10(15), 1774. https://doi.org/10.3390/electronics10151774
4. J. Shen, Z. Luo, C. Wu and J. Li, "BAHS: A Bandwidth-Aware Heterogeneous Scheduling Approach for SDN-Based Cluster Systems," 2017 IEEE International Symposium on Parallel and

ISSN PRINT 2319 1775 Online 2320 7876
Research paper Volume 11, Iss 9, 2022

Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC), Guangzhou, China, 2017, pp. 638-645, doi: 10.1109/ISPA/IUCC.2017.00101.

5. Bergui, M., Najah, S. & Nikolov, N.S. A survey on bandwidth-aware geo-distributed frameworks for big-data analytics. J Big Data 8, 40 (2021). https://doi.org/10.1186/s40537-021-00427-9
6. Jeyaraj, R., Ananthanarayana, V.S. & Paul, A. Fine-grained data-locality aware MapReduce job scheduler in a virtualized environment. J Ambient Intell Human Comput 11, 4261–4272 (2020). https://doi.org/10.1007/s12652-020-01707-7
7. K. Oh, M. Zhang, A. Chandra and J. Weissman, "Network Cost-Aware Geo-Distributed Data Analytics System," in IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 6, pp. 1407-1420, 1 June 2022, doi: 10.1109/TPDS.2021.3108893.
8. Li, C., Bai, J., & Tang, J. (2019). Joint optimization of data placement and scheduling for improving user experience in edge computing. Journal of Parallel and Distributed Computing, 125, 93-105. https://doi.org/10.1016/j.jpdc.2018.11.006
9. Sharma, A., Singh, G. (2020). A Review of Scheduling Algorithms in Hadoop. In: Singh, P., Kar, A., Singh, Y., Kolekar, M., Tanwar, S. (eds) Proceedings of ICRIC 2019 . Lecture Notes in Electrical Engineering, vol 597. Springer, Cham. https://doi.org/10.1007/978-3-030-29407-6_11
10. C. Rista, D. Griebler, C. A. F. Maron and L. G. Fernandes, "Improving the Network Performance of a Container-Based Cloud Environment for Hadoop Systems," 2017 International Conference on High Performance Computing & Simulation (HPCS), Genoa, Italy, 2017, pp. 619-626, doi: 10.1109/HPCS.2017.97.