Bandwidth-Aware Routing Mechanism to Control Hadoop Shuffle Traffic over Software-Defined Networking

Ming-Syuan Wu
Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 807, Taiwan; Email: i111154101@nkust.edu.tw
Cheng-Han Lin
Department of Health-Business Administration, Fooyin University, Kaohsiung 831, Taiwan
Wen-Shyang Hwang*
Department of Electrical Engineering, National Kaohsiung University of Science and Technology, Kaohsiung 807, Taiwan; Email: wshwang@nkust.edu.tw
Ce-Kuen Shieh
Department of Electrical Engineering, National Cheng Kung University, Tainan 701, Taiwan
Mao-Syun Lin
Institute of Computer and Communication Engineering, National Cheng Kung University, Tainan 701, Taiwan

Abstract

MapReduce is the main program in the Hadoop computing architecture. It involves mapping and reduction processes that require servers to exchange large amounts of data with each other. Data exchange between servers during the shuffle stage can lead to insufficient network bandwidth. In the present study, an algorithm for distributing Hadoop shuffle traffic to all possible paths was developed. Software-defined networking (SDN), a centrally controlled form of network architecture, was used to collect network status parameters to allocate shuffle traffic. During traffic distribution, the Mininet simulator was used to build the network topology and simulate operation. The Ryu controller was used as the SDN controller. The simulation results indicated that the proposed method was superior to the Spanning Tree Protocol (STP) and bandwidth-aware algorithm in reducing Hadoop completion time through effective shuffle traffic distribution.
Keywords: Software-Defined Network, Apache Hadoop, MapReduce, Shuffle

1. Introduction

Data continue to increase in volume and complexity, and Apache Hadoop [1][2] represents an efficient solution to this problem. Hadoop is well known for its MapReduce process, which comprises mapping and reduction components. MapReduce divides data into small blocks, which are then processed individually. Between the mapping and reduction stages, a large volume of intermediate data is produced. The exchange of intermediate data between servers is referred to as the shuffle stage, during which data must be transmitted over a network connection. If servers are on different racks or further from one another, network strength can substantially affect the overall computing performance of the Hadoop framework [3].

To address these problems, the present study investigated the use of software-defined networking (SDN) to improve link utilization. SDN is a centrally controlled network architecture, and it was selected in this study to achieve a better understanding of network usage. Hadoop traffic was allocated according to network on the basis of traffic distribution, which enabled the simultaneous transfer of intermediate data during the shuffle phase through two paths. Thus, this method maximized network bandwidth use. The main contributions of the study are as follows:

1. The study proposes a multipath method for distributing Hadoop traffic.
2. The proposed method exhibited good experimental performance.

The remainder of this paper is organized as follows. In Section 2, we introduce contextual information regarding the Hadoop and SDN architecture, including a review of relevant literature. Section 3 presents our methodology, and Section 4 provides the evaluation and results of the experiment. Section 5 comprises the conclusion and presents recommendations for future research.

2. Background and Literature Review

2.1. SDN

To achieve central control, SDN architecture separates networks into control and data planes, with the control plane including an SDN controller. The controller communicates with switches in the data plane by sending messages according to the OpenFlow protocol [4]. The switches follow these instructions to forward or drop data.

2.2. Apache Hadoop

Apache Hadoop is a framework for distributed storage and big data computing that is typically used on computer clusters. The two main Hadoop components include the storage system and Hadoop distributed file system (HDFS). Additional components include the processing system and the MapReduce programming model.

HDFS uses distributed technology to store big data. Under this system, input data are partitioned into blocks and spread between servers. MapReduce [5] is a programming model designed to distribute data in

parallel across mapping and reduction stages. The intermediate shuffle stage can cause network congestion, which is a notable problem.

2.3. Bandwidth-Aware Methods

A prior study [6] proposed a method to identify the weights on all possible paths from the source to destination node on the basis of the workload per node and the available bandwidth. This approach focuses on the routing algorithm, with the transmission path assigned flow by flow on the basis of network status. The selection of a suitable path with adequate bandwidth improved network utilization relative to that of traditional routing algorithms, such as the Spanning Tree Protocol (STP). However, although bandwidth-aware methods can avoid network congestion, they account for only a fraction of network usage.

3. Methodology

3.1. Architecture Overview

In the present study, we examined a multipath distribution architecture for network traffic during the shuffle stage of the Hadoop MapReduce procedure. The proposed method can dynamically allocate transmission paths according to current network status by separating network architecture into control and data planes. The two modules of this multipath distribution method are displayed in Fig. 1. The first module is the switch monitor, which periodically queries every switch for bandwidth usage. After the switch monitor collects data on bandwidth usage, the controller sends these data to be processed by the multipath allocation module. The multipath allocation module then processes these data to identify the best weight for all possible paths from source to destination.

3.2. Bandwidth-Aware Routing Mechanism

The control plane adopted in the present study was based on OpenFlow 1.3 and the Ryu controller [7]. The proposed architecture included the following five steps, as displayed in Fig. 2:

1. User assigns a Hadoop job.
2. The switch monitor periodically queries the network status.
3. Eq. (1) is used to calculate the available bandwidth of each possible path.
4. Eq. (2) is used to calculate the weights of the possible paths.
5. The controller asks the switches to install the new flow entries.

$$Port\_abw = \max_{\text{bandwidth}} - \frac{(rx+tx)_{now} - (rx+tx)_{pre}}{(during\_time)_{now} - (during\_time)_{pre}} \quad (1)$$

$$Weight_i = \frac{abw_i}{\sum_1^t abw_i} \quad (2)$$

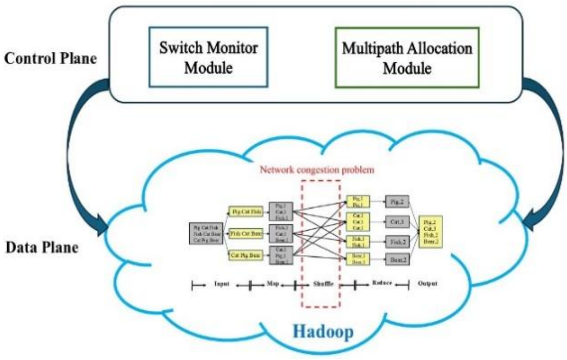

Fig. 1 Architecture Overview

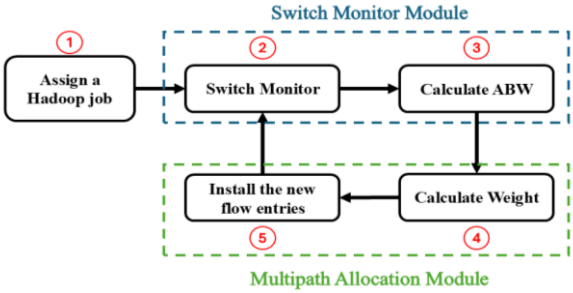

Fig. 2 Distribution Process

At the beginning of the distribution process, the user assigns a Hadoop job on the server. When the network loading changes, the switch monitor begins querying the network status of each switch. After status information, such as the number of transmitted and received packets, is received, the available bandwidth of each possible path is calculated according to Eq. (1). Here, possible paths refer to all available routing paths from the source to the destination node. Eq. (1) subtracts bandwidth consumption from the total bandwidth of the path over the period of the query time interval. The weight of each possible path can be calculated by inputting the result of Eq. (1) into Eq. (2). After calculation, the SDN controller sends messages asking the switch to install a new flow entry, which the switch will follow until the next cycle of the distribution process.

Algorithm 1 presents the process of traffic distribution according to the packet choice of the transmission path. This algorithm is initiated when the switch receives a packet that it does not know where to send from another device. The switch then encapsulates and sends this packet to the SDN controller.

The SDN controller verifies the relationship between the source and destination. If the source and destination are in the same pod, the controller obtains the layer of the switch by checking its identification. The controller asks aggregation layer switches to send the packet back to the edge layer switch. Source and destination servers either do or do not connect to the same edge layer switch. In the first case, our traffic distribution method can be applied to transfer the packet to a remote location. In the second case, packets are sent to the relevant port. Three layers of switches are used for transferring remote packets. The core layer switch is used to communicate among pods, connected by one path each. As the controller receives

remote packets from the aggregation switches, the packets may be forwarded up (out of the pod) or down (into the pod). The traffic distribution method can be used for the up direction.

Algorithm 1: Bandwidth-Aware Routing Algorithm

```
When packet_in event occurs in controller:
if Src and Dst are pod-local:
    if switch ∈ aggregate:
        return: Send back to pod
    if switch ∈ edge:
        if Src and Dst are not rack-local:
            multipath allocation
        else:
            return: Send to particular port
else:
if switch ∈ core:
        return: Send to particular pod
    if switch ∈ aggregate:
        if Dst & switch are in the same pod:
            return: Send to particular port
        else:
            multipath allocation
    if switch ∈ edge:
        if Dst is connected to the switch:
            return: Send to the particular port
        else:
            multipath allocation
```

4. Evaluation and Experimental Results

4.1. Environment

In the experiment, 12 Docker containers were used as servers to run the Hadoop WordCount application through Open vSwitch. The bandwidth was set to 20 Mbps between the core layer and aggregation switches and to 10 Mbps between the aggregation and edge layer switches. The environmental configuration is presented in Table 1. The experiment included general and federated Hadoop architectures. The general Hadoop architecture involves only one cluster, which is controlled by a single Name Node. The federated Hadoop architecture involves multiple clusters, each of which is controlled by its own Name Node. The federated Hadoop architecture is used to process iterative jobs, resulting in a high volume of MapReduce processes. We used this architecture to verify whether our traffic distribution method could handle a high network load. The topology of the federated Hadoop architecture is showed as Fig. 3.

The general Hadoop architecture was used for WordCount jobs with 250 Mbytes of input data. Additionally, we compared the STP method with the bandwidth-aware routing mechanism that was built into the topology of our traffic distribution architecture. Iperf was used to generate background traffic at 10 Mbps to congest the path. Before beginning the experiment, we verified whether the flow and port status matched the input statistics of our traffic distribution method.

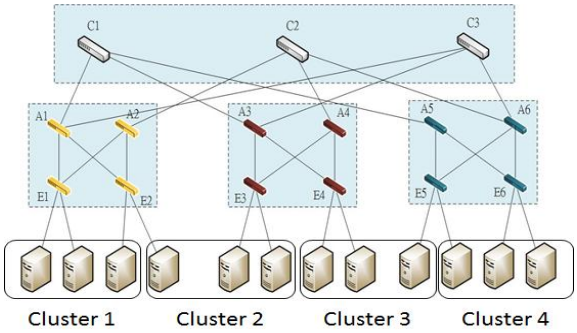

Fig. 3 Federated Hadoop Topology

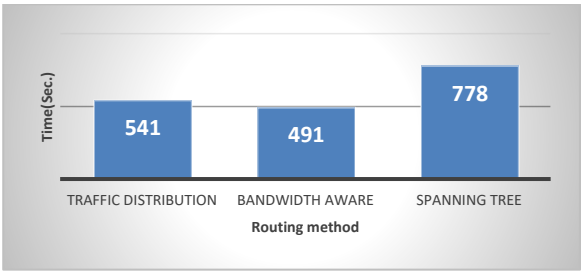

Fig. 4 Data Transfer Time Between Networks

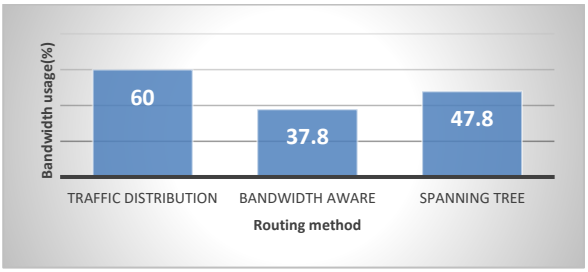

Fig. 5 Bandwidth Usage

Table 1. Environmental Configuration

Node	**12**
Input size	250Mbytes
Bandwidth	Core←→Aggregate:20Mbps Aggregate←→Edge:10Mbps
Name node	1
Data node	12

4.2. Experimental Results

Background traffic for the simulation was generated using Iperf, which set this traffic to be transmitted through a single path. As indicated in Fig. 4, 250 Mbytes of data were transferred between networks through different routing methods. Background traffic lasting for 300 seconds was transmitted from server 1 to server 4. Then, the 250 Mbytes of data were transmitted from server 1 to server 3. As displayed in Fig. 4, the bandwidth-aware and traffic distribution methods had nearly equal traffic shares due to dynamic path changes. Fig. 5 indicates that the traffic distribution method maximized the use of available bandwidth, resulting in bandwidth usage twice that of the STP method.

5. Conclusion

In the present study, a bandwidth-aware routing mechanism was used to distribute Hadoop shuffle traffic. The SDN architecture calculated the weight of each possible path from source to destination and then redirected traffic accordingly. The results of our experimental evaluation demonstrated that although the network status was congested for some paths, the proposed mechanism achieved a lower Hadoop completion time than did the STP method and bandwidth-aware routing algorithm due to maximal use of available bandwidth. In future studies, we will focus on allocation algorithms and the application of iterative jobs using our allocation method.

Acknowledgments

This work was supported by the National Science and Technology Council (NSTC), Taiwan, under Contract No. NSTC 113-2221-E-242-001 and NSTC 113-2221-E-992-001.

References

1. N. Ahmed, Andre L. C. Barczak, Teo Susnjak and Mohammed A. Rashid, “A comprehensive performance analysis of Apache Hadoop and Apache Spark for large scale data sets using HiBench,” Journal of Big Data, vol 7, 110, 2020.
2. R. Machova, J. Komarkova and M. Lnenicka, “Processing of big educational data in the cloud using Apache Hadoop,” in: 2016 International Conference on Information Society (i-Society), 2016.
3. N. M. Ahmad, A. H. Yaacob and A. H. M. Amin, “Performance analysis of MapReduce on OpenStack-based hadoop virtual cluster” in: 2014 IEEE International Symposium on Telecommunication Technologies (ISTT), 2014.
4. Jose Miguel-Alonso, “A Research Review of OpenFlow for Datacenter Networking,” IEEE Access, vol. 11, pp. 770-786, 2022.
5. J. Dean S. Ghemawat, “MapReduce: Simplified data processing on large clusters”, in: Communications of the ACM, vol. 51, pp. 107-113, 2008.
6. W. Shi, Y. Wang, J. P. Corriveau, B. Niu, W. L. Croft and M. Peng, “Smart Shuffling in MapReduce: A Solution to Balance Network Traffic and Workloads,” in: 2015 IEEE International Conference on Utility and Cloud Computing (UCC), 2015.
7. Anil Ram, Manash Pratim Dutta and Swarnendu Kumar Chakraborty, “A Flow-Based Performance Evaluation on RYU SDN Controller,” Journal of The Institution of Engineers (India): Series B, vol 105, pp203-215, 2024.

Author Introductions

Ming-Syuan Wu

Ming-Syuan Wu received his M.S. degree (2021) from the Department of Electrical Engineering, National Kaohsiung University of Science and Technology, where he is currently a doctoral student.

Cheng-Han Lin

Cheng-Han Lin received a Ph.D. degree from the Department of Electrical Engineering, National Cheng Kung University, in 2010. He is currently an associate professor in the Department of Health-Business Administration at Fooyin University. His current research interests include wireless and SDN networks.

Wen-Shyang Hwang

Wen-Shyang Hwang received B.S. (1984), M.S. (1990), and Ph.D. (1996) degrees from the Electrical Engineering Department of National Cheng Kung University, Taiwan. He is currently a professor in the Electrical Engineering Department at National Kaohsiung University of Science and Technology. His current research interests include multimedia wireless communication, wireless mesh networks, and Internet QoS.

Ce-Kuen Shieh

Ce-Kuen Shieh is currently a professor in the Department of Electrical Engineering at National Cheng Kung University, Tainan, Taiwan, where he also obtained his B.S., M.S., and Ph.D. degrees. His current research interests include distributed and parallel processing systems and computer networking.

Mao-Syun Lin

Mao-Syun Lin received his M.S. degree (2017) from the Institute of Computer and Communication Engineering at National Cheng Kung University. His current research interests include wireless networks and multimedia communication.