Traffic Optimization for Hadoop based on SDN

Siyuan Liu, Xinyi Wang, Jiaxu Zhu
Computer Science and Engineering Department
University of California, San Diego

***Abstract*—**

Keywords*—*Hadoop, SDN, Virtualization

I. Introduction - Siyuan

Hadoop is a open-source software framework for storing and running computing applications on a cluster of computers or hosts. Nowadays, many tasks of companies and research associations are accomplished through taking use of Hadoop. For the sake of efficiency and economy, people request a Hadoop cluster system with better performance. The performance of Hadoop relies on many factors. The basic factor is the algorithm of Hadoop, like how the mapper and reducer work and how the Hadoop controller distributes tasks to mappers/reducers. Since Hadoop is deployed on a cluster of computers, the network for the cluster obviously affect the trasmission of data and control information. In intuition, even though the mappers and reducers can work very fast, the network with high latency and drop rate which cause a long trasmission time will lead a Hadoop system with bad performance.

Our concern is focused on the network of Hadoop system. According to some basic experiment, the time of maps and reduces only occupy about 30% of the completion time for one Hadoop job. It means network transmmission takes lots of time in a Hadoop job. The fact gives us a oppotunity to improve the performance of Hadoop system.

Network for Hadoop system is much different from the network we use for daily life. The features of Hadoop systems and applications make the network maintain some properties. We talk about the features of Hadoop and the network in the Section V. If we can catch and take use of the properties and information, there will be a more efficient network for Hadoop jobs whose completion time mainly comes from maps and reduces. This is why we consider about using SDN for the Hadoop cluster. Software-Defined Networking (SDN) is a approach to allow a network controller to manage the network scheduling in order to take fully use of the resources and improve the performance of the softwares running on the network. SDN decouples the decisions where the traffic is sent (control plane) from how to forward the traffic to the destination (data plane). As to Hadoop applications, SDN aims at a better traffic scheduling by considering how the mappers and reducers work and how the tasks are distributed.

Our idea of traffic optimization for Hadoop based on SDN derives from $FlowComb$ and $Pythia$. They build a SDN control module for the Hadoop cluster system which collect the information from running mappers and reducers and then adjust flow scheduling for the routers and switches in the network. Our system architechture is designed according to the same idea, but we made some modification and adjustment for some algorithm details and implementation of the system. It is mostly concerned about SDN for Hadoop that what kinds of information should be collected and how the network and flow are scheduled. On the whole, we collect information like the start/end time of hadoop map/recude tasks on each node and the activities of HDFS. Then the SDN controller analyses the information and updates a $Load\ Graph$ held in the controller and deploy new flow scheduling rules to the switches and routers regularly. The $Load\ Graph$ reflects the load level for the facilities in the network. Our goal is to schedule flows avoiding the nodes with large load.

We implement our system with several useful softwares which help us to simulate a Hadoop cluster system and SDN control module. The used softwares are depicted in the Section VI. We take $completion\ time$ for Hadoop jobs as the only critical metric for the performance of Hadoop on SDN. But we also analyse some variables, like the rate of time of maps and reduces in the total completion time. We made series of experiment on our system. We test our system with different data size and Hadoop jobs in order to observe how is the performance on different Hadoop applications. Also we compared our system with $Pythia$ and baseline (normal network without SDN). Actually we got almost same performance with $Pythia$ and better than the baseline.

The rest of this paper is structured as follows. We review and put our work in the context of related work in Section II. Architect of Hadoop on SDN for our system is depicted in Section III. We describe more details of the design of Hadoop cluster system (Section IV) and traffic optimization by SDN (Section V). At last, we show the implementation of our entire system and the experiments in Section VI. And conclusion and further discussion are in Section VII.

II. Related Work - Xinyi

Watering

(Refer to papers found: flowcomb, pythia)

III. Architecture of Hadoop on SDN - Siyuan

At a high level of abstraction, our system included three components shown in Figure 1: 1) Hadoop cluster with SDN. This part is mainly deployment of Hadoop master/slave nodes. For the sake of SDN, we replace normal routers and switches with open source switches. 2)Instrument Process (we call it Auxiliary Process) runs on each node who collects information of the Hadoop node and sends it to the SDN control system. 3)SDN control system. This module is the core of our system architecture. It consists of two parts: routing computation and flow allocation. After collecting the information from auxiliary processes, the control module executes routing computation to check if the topology of the network changes, then the flow allocation part will update flow allocation rules according to the $LoadGraph$ which is gotten from the analysis of information collected.

The logical relations of the three components pf the system architecture are described as follows. The Hadoop cluster can work without SDN control. We deploy Hadoop nodes on a set of virtual machines. And we create a $Fat\ Tree$ network topology (ex. Figure 2)for the Hadoop nodes. The Fat Tree topology eliminates the influence from bottleneck links and take use of more available resources in the network. By choosing the fatness of links, the network can efficiently use any bandwidth made available by packaging and communications technology. After the building of Hadoop cluster system, we can deploy a auxiliary process on each Hadoop slave node which monitors the activities here, like the tasks distributed and finished and the HDFS activities. The auxiliary processes send information collected on the node to the SDN control system and keep monitoring. When the Hadoop cluster system with open source switches and auxiliary processes ready, the SDN control system can work to schedule more efficient network transmission. First, the event collector collect all the information sent from the auxiliary processes on the Hadoop nodes. The event collector parses the information as different inputs, such as $< inf\_type, node\_n, task\_header, timestamp >$ and $< inf\_type, hdfsaccess, timestamp >$, for the routing computation and flow allocation. The routing algorithm for our system is k-shortest paths. Unless the network topology changes, the routing computation happens once at the start of the whole system. With k-shortest paths available, the goal of flow allocation is to distribute packets on the paths with low load which avoid high latency and drop rate. Because finding optimal flow distribution on a graph a NP-complete problem, our system computes a set of next hops for one packet with the help of $Load\ Graph$. More details about network scheduling will be depickted in Section V.

The goal of using SDN is to get fast and efficient transimission. The most important thing is to monitor the status of the current networking and generate feedback to control the following network flows. Our system obeys the intuition which tries to collect information from the running Hadoop nodes and compute new flow allocation rules. Our experiment shows that it is a positive cycle between collection and adjustment.

IV. Virtual Hadoop Cluster

A. Introduction to Hadoop - Jiaxu

Watering

B. Traffic Problems from Hadoop - Xinyi

Relate Hadoop with Network.

V. Optimize Shuffle Traffic using SDN

A. Introduction to SDN - Xinyi

Watering

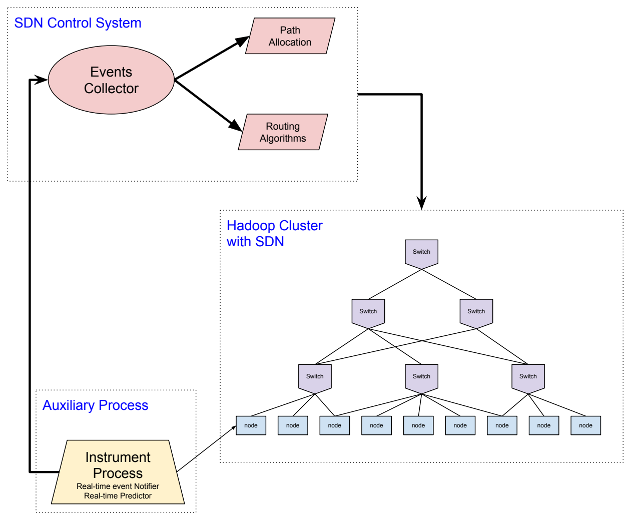

Fig. 1. Architecture of our system design

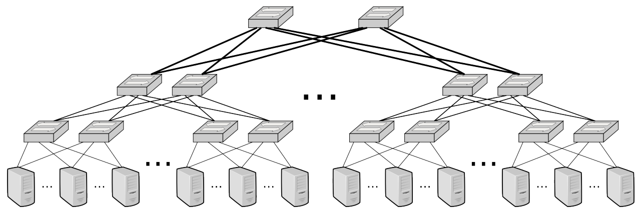

Fig. 2. Fat Tree Topology

B. Network Scheduling - Siyuan

In this section, I will introduce the details of network scheduling in SDN control module. In general, the network scheduling consists of two parts: routing and flow allocation algorithms. During startup, the scheduler gets a fat-tree network topology of the Hadoop cluster and compute a routing graph based on k-shortest paths algorithm (Dijkstra shortest-path algorithm). We choose to use k-shortest paths algorithm because we try to implement a multipath routing system which take use of more links and bandwidth in the network. The routing part defines how the packets or flows are transfered to the destination which is also called data plane. The k-shortest paths routing algorithm is hop-based and executed again to generate a updated routing graph only when the network topology has some changes.

In order to introduce the flow allocation algorithm we designed for our system. I need to explain two things: 1) the way to generate a $Load\ Graph$ with the information collected from each Hadoop nodes; 2) the details of flow allocation algorithm about how to utilize the $Load\ Graph$. Some formats of information are mentioned above. For example, the task activity $< inf\_type, node\_n, task\_header, timestamp >$ denotes that one node was distributed a new task or finished a task, where inf_type denotes the information type of this item, $node\_n$ denotes the Hadoop node number, $task\_header$ contains the task details like task type and data size, $timestamp$ denotes the time when the information item is sent. With these information, I designed a $load\ weight\ algorithm$ to add/delete some load weight on switches. If the information shows that there is a new task distributed on a nodes, then the upwards switches are added some load weight. Formula are abstracted below: define a load weight w for this round of update, and w is related with the data size of this task

$$First - layer\ upwards\ swithces:\ weight(switch) + = w$$
$$Second - layer\ upwards\ swithces:\ weight(switch) + = w/2$$
$$Third - layer\ upwards\ swithces:\ weight(switch) + = w/4$$

Because of the fat-tree topology, one layer may have lots of switches I just update first several switches on the routing table (based on k-shortest paths algorithm mentioned above). The strategy of load weight deletion is similar with the addition.

With the load graph, next step our network scheduler will generate new flow allocation rules for each switches and deploy them on switches regularly. For each switch, we contain a routing table which is computed by k-shortest paths algorithm. For each path, the flow or bandwidth available are in inverse proportion of the $load$ for the next hop. Therefore, I maintain a set of next hops for one destination, and allocate flow according to the $load$ of the next hops.

Thus, we design a network scheduling framework of routing computation and flow allocation. The last step it to deploy the new flow allocation rules on open source switches. We decide to deploy it regularly. More specifically, we deploy it after the $Load\ Graph$ is updated for several times.

Currently, all of the SDN network control logic of our system is implemented in the form of modular components within an OpenFlow controller POX. We will describe it in Section VI.

VI. Evaluation

A. *Experimental Setup - Jiaxu*

System Building (Softwares)

B. *Experiments - Jiaxu*

Groups of Experiments, results and analysis.

The rate of Computing time compared to Networking time

VII. Conclusion

Watering

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.