Using SDN to Visualize the Network Behavior of MapReduce

William Clay Moody wcm@clemson.edu
Jason Anderson jwa2@clemson.edu
Dylan Wagenseller twagens@clemson.edu
K.-C. Wang kwang@clemson.edu
Amy Apon aapon@clemson.edu

Motivation

The shuffle phase of a Hadoop MapReduce job consists of transfers of data from maps to reduce nodes. In the literature, a network bottleneck is often mentioned during this phase negatively impacting the job's overall execution time.

Hadoop system administrators do not possess tools to monitor the network performance of running MapReduce jobs and understand whether network optimizations could improve job run time.

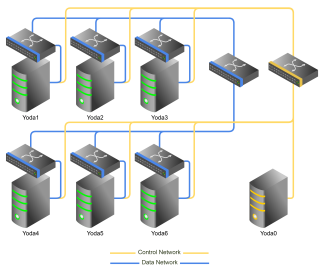
Objectives

Using Software Defined Networking (SDN) and the Hadoop system, build a distributed set of tools for a HPC environment to monitor, record, and visualize network performance of submitted jobs during and after execution.

Materials and methods

We assembled a small Hadoop cluster using the following equipment:

- (1) Dell Precision 330 workstation
- (6) Sun Ultra20-M2 workstations
- (2) Pica8 Pronto 3290 OpenFlow switches



The machines were configured as a single master with six slave nodes, connected via two independent networks. On each slave node, Open vSwitch was used to control and monitor traffic on the data network. The virtual and physical switches were controlled by Floodlight on the head node.

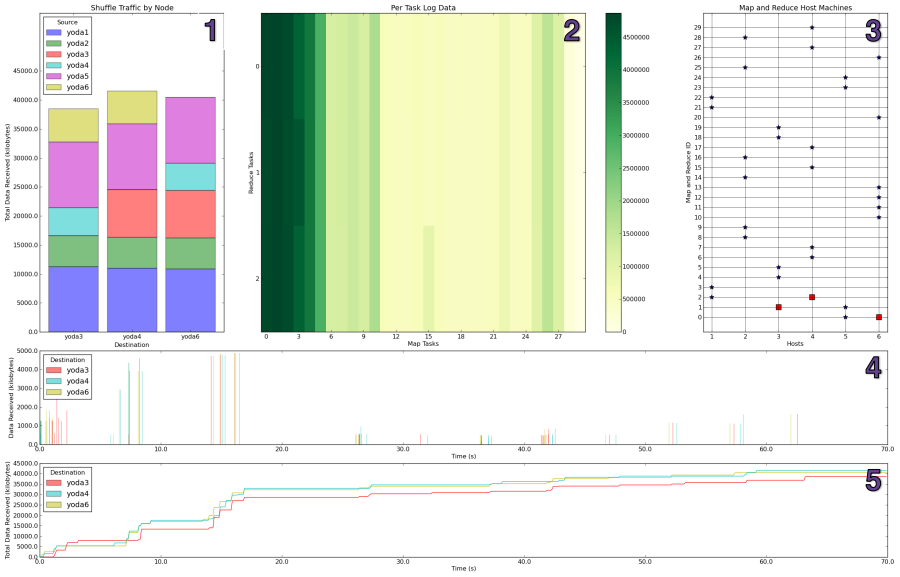
Results

We developed software to interface with different components of the SDN enabled Hadoop cluster to build a visualization of the network performance after job execution. The visualizations will enable a user or system administrator to understand the characteristics of the shuffle phase during the job. The individual graphs are numbered and described below.

① In this graph, the total data received by reducers on each node is categorized by the source node. Large bar segments indicate greater map output, which would ideally be sent to reducers on the same node. Statistics for this graph were gathered by measuring network traffic at each node with OpenFlow.

② A heat map presents the size of the data transferred from each map to each reduce regardless of the assigned task tracker. This information is gathered from the individual task tracker logs within Hadoop. Discovering reduce and map tasks with imbalance shuffle traffic is quickly accomplished in this visualization.

③ The grid shows map and reduce tasks assignments to task trackers. Blue stars represent map tasks and red squares represent reducers and both are listed along the y-axis. This data was gathered from the Hadoop JobTracker via its REST API. This graph helps visualize how the maps and reducers were distributed in a certain Hadoop job.



④ A timeline displays the fetching of map task output by each reduce and the quantity of the data. The information to build this visualization is collected from the individual task tracker logs within the Hadoop cluster. This view allows a user to understand the distribution of the network traffic over the lifetime of the job and understand when the load is maximized on the network.

⑤ Querying the flow statistics from each virtual switch at specificed intervals builds a cumulative distribution view of the network traffic for each destination node. This visualization can reveal if any reducers were underutilized during job execution.

Conclusions

Using Software Defined Network and the Hadoop System, we developed a set of tools allowing users to monitor and visualize the network behavior of a Hadoop MapReduce job. This framework can be used in further research on Hadoop network performance.

Our environment enables system administrators and users to quickly understand the network behavior of a MapReduce job with node-to-node, task-to-task, task-to-node, and timeline visualizations.

Future Work

The next step in our project is to use our tools to discover under which network conditions a network bottleneck occurs and whether these conditions are likely in a production Hadoop cluster. Then, we will use SDN to change the network configuration to try and alleviate the performance degradation.

To do this, we will use SDN to manipulate the topology such that shuffle traffic travels the least amount of hops possible. To that end, we have started on Phase II of our SDN testbed:

In preparation, we have added six more nodes to the cluster and equipped each node with a total of six network interfaces. Four of the interfaces on each node connect to one of the OpenFlow switches, which allows us to programmatically configure the network topology to simulate different computing environments.

Citations

Guohui Wang, T.S. Eugene Ng, and Anees Shaikh. **Programming Your Network at Run-Time for Big Data Applications**. *In Proceedings of the first workshop on Hot topics in Software Defined Networks* (HotSDN '12). ACM, New York, NY, USA, 103-108.

N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. **OpenFlow: enabling innovation in campus networks**, *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, April 2008.

Jeffrey Dean and Sanjay Ghemawat. **MapReduce: Simplified Data Processing on Large Clusters**. *In Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation* - Volume 6 (OSDI'04), Vol. 6. USENIX Association, Berkeley, CA, USA, 10-10