Application-Aware SDN Routing for Big-Data Processing

Evaluation by EstiNet OpenFlow Network Emulator

Director/Prof. Shie-Yuan Wang
Institute of Network Engineering
National Chiao Tung University
Taiwan

TUS 2014, 12/4/2014
Outline

• Introduction and Motivation
• Background of SDN, Hadoop, and EstiNet
• Contribution of this work
• Four Different Routing Schemes Studied
• Design and Implementation
• Performance Comparison Results
• Conclusion
Introduction

- A big-data application processes a huge amount of data on many servers.
- These servers need to exchange their data over a network.
- The current network is hard to understand the needs of applications. Its bandwidth cannot be used very efficiently to meet the run-time bandwidth demands of the application.
- Software-Defined Networking (SDN) provides an opportunity for the network to be application-aware.
- Our work is to use SDN to reduce the data shuffle time of a big-data application over a network.
- This is achieved by dynamically allocate better routing paths to shuffling flows (and reroute them if necessary) in a big-data application.
Background

- SDN
- Hadoop
- EstiNet
SDN

- A new and emergent network technology that centrally controls the network
- Separate the control plane (e.g., routing) from the data plane (e.g., packet forwarding) of a network device
- Implement control logic outside of the network device by a centralized controller
- Use the OpenFlow protocol to control all network devices in a network
- The controller installs flow entries into network devices to control their packet forwarding behavior.
SDN (cont’d)

- Has a global view to find the best routing path for a flow at run time
- Can dynamically change the paths of flows at run time to best utilize the network bandwidth
- In our work, we studied the performance of the Floodlight SDN controller and used it as the platform to implement our application-aware routing scheme.
Software-Defined Network (SDN)

Northbound API

Controller (e.g., NOX/POX, Floodlight, Open Daylight, etc.)

Southbound API (e.g., OpenFlow)

Packet Forwarding

App (e.g., network virtualization)

App (e.g., network management)
Hadoop

- Hadoop is a very popular open source Big-Data application system developed by Apache software foundation.
- The Hadoop MapReduce System splits a job into multiple tasks to process them in parallel on multiple servers.
- The phases of a job is divided into the Map, Shuffle, and Reduce phases.
- During the Shuffle phase, many servers need to exchange their intermediate results over a network.
Hadoop (cont’d)

- Our goal is to reduce the data shuffle time by using the SDN technology to allocate better routing paths to shuffling flows (and reroute the paths if necessary).
- This is achieved by the Hadoop controller telling the SDN controller the transfer traffic volumes of shuffling flows.
Hadoop

Our Focus

Read input data block
Map
Output intermediate file
Shuffle
Receive intermediate data
Reduce
Output the result
EstiNet

- EstiNet is a network simulator and emulator developed by me and EstiNet Technologies, Inc. (www.estinet.com).
- An article describing it was published in IEEE Communication Magazine, September 2013.
- EstiNet uses an innovative methodology to accurately simulate and emulate an OpenFlow (V1.4) network.
- In EstiNet, the real-life TCP/IP protocol in Linux is directly used to generate accurate results.
- In EstiNet, real-life network application programs can be directly run up on hosts in a simulated network to generate realistic traffic.
- It is more accurate, reliable, and scalable than other similar tools such as Mininet. (IEEE ISCC 2014 conference paper)
You Can Download It to Give it a Try
A Book about EstiNet Is Soon To Be Published
EstiNet (cont’d)

• We run EstiNet on a server machine with 16 Ethernet ports to emulate the OpenFlow network. (The server machine is equipped with 4 Ethernet cards, each with four Ethernet ports.)

• The emulated OpenFlow network is a fat-tree network composed of 20 OpenFlow switches.

• 16 Hadoop servers connect to this EstiNet machine to exchange their real data over the emulated network.

• By using EstiNet, we do not need to purchase 20 OpenFlow switches and can thus save a lot of cost.
The Network Emulated by EstiNet

- A fat tree topology

This OpenFlow network is emulated by a machine running EstiNet.

These servers are all real servers. They all connect to the EstiNet machine.
Contribution of This Work

• We study how effective an application-aware routing scheme can reduce the data shuffle time of a big-data job.

• We compared our application-aware routing scheme against 1) the spanning tree protocol, 2) the Floodlight SDN controller, and 3) the ECMP routing scheme. (None of these schemes are application-aware.)

• We found that the traditional spanning tree protocol and the Floodlight SDN controller perform the worst.
Contribution of This Work (cont’d)

- We found that the ECMP scheme performs quite well, although it is not application-aware. It can outperform the spanning tree protocol or the Floodlight by up to 50%.
- We found that an application-aware SDN routing scheme (such as ours) can further outperform ECMP by 20%.
Four Routing Schemes Studied

- The Spanning Tree Protocol
- Floodlight SDN Forwarding Module
- Equal Cost Multiple Path (ECMP) Routing
- Our Application-Aware SDN Routing
Spanning Tree Protocol

- Its goal is to prevent the packet broadcast storm problem, which may happen on a network with loops.
- Select some links on the network topology to construct a spanning tree.
- Unselected links are disabled.
- Packets are forwarded only on the spanning tree, thus no routing loops and packet broadcast storm problem will happen.
- This is the most common approach used today.
- But the network bandwidth is not fully utilized because the links not on the spanning tree are not used.
A Constructed Spanning Tree
Floodlight SDN Controller

- The forwarding module in the Floodlight SDN controller wants to overcome the spanning tree problem.
- It gives each server a different broadcast tree rooted at that server.
- The goal is to use N broadcast trees when there are N servers in the network, hoping that all links and their bandwidth can be used at the same time.
- In theory, these N broadcast trees can use all links at the same time.
- However, we found that in practice, these N broadcast trees are quite similar and they only use a few common links.
- Thus, you will see that the Floodlight SDN forwarding module performs as poorly as the spanning tree protocol.
A Floodlight Broadcast Tree Rooted at Server 9
Equal Cost Multiple Path (ECMP) Routing

- Normally, the routing path between a pair of servers is a single path.
- ECMP tries to use equal-cost (i.e., equal number of hops) multiple paths between a pair of servers at the same time to increase the transfer throughput and reliability.
- ECMP normally works with a layer-3 routing protocol such as the OSPF routing protocol.
- To spread traffic over multiple paths, ECMP can use the round-robin policy (or other policies such as the random policy).
- Due to the use of multiple paths at the same time, you will see that ECMP can outperform the spanning tree protocol and the Floodlight SDN forwarding module by up to 50%.
Two ECMP Paths Used In Parallel between Server 3 and Server 14
Our Application-Aware SDN Routing

- It chooses routing paths with no load (or light load) for shuffling flows to maximize the use of network bandwidth.
- It monitors the load of all links in the network at any time.
- It knows the transfer traffic volumes of each shuffling flow from Hadoop controller and uses the information to predict future link loads.
- It uses the special property of the fat-tree network topology to know the degree of freedom of rerouting a path between two servers.
- It uses the SDN technology to easily set up routing paths and re-route them when necessary.
- It performs the best among the four studied routing schemes.
Interworking between Hadoop, Floodlight, and EstiNet
Design and Implementation
Three Kinds of Connections

- A connection between two servers can be classified into (i) a rack-local connection (they are directly connected by an edge-layer switch), (ii) a pod-local connection, or (iii) a remote connection.
The relationship between two pods can be (i) one-to-one mapping (e.g., pod 0 and pod 2) or (ii) all-to-all mapping (e.g., pod 0 and pod 1)
Implementation on Floodlight Controller

- Hadoop
  - Provide shuffling flows’ information
- MapReduceManager
  - Provide shuffling flows’ information
- RoutingComponent
Implementation of the Routing Component
Routing Path Allocation Priority

- For better path allocation, the RoutingComponent allocate paths for shuffling flows in the following order:
  - One-to-one mapping remote-connection shuffling flows have the highest priority (because they need to use core switches and have less scheduling freedom)
  - All-to-all remote-connection shuffling flows have middle priority (because they need to use core switches and have higher scheduling freedom)
  - The pod-local shuffling flows have the lowest priority (because they need not use core switches)
  - Finally, the rack-local shuffling flows are then scheduled (because their paths are unique and direct paths)
Path Selection from Multiple Equal-Cost Paths

- When iteratively allocating paths for shuffling flows, if a shuffling flow has multiple equal-cost paths to choose from, we choose the path according to the following rule:

![Diagram showing flow transfer volumes between src and dst](image)
Performance Evaluation
Performance Evaluation

- Emulated network topology: a fat tree topology
  - Link bandwidth: 5 Mbps (to make the emulation still real-time)
  - Link propagation delay: 1 ms

- Routing schemes compared:
  - Spanning Tree protocol
  - Floodlight SDN Forwarding module
  - ECMP with Round-Robin policy
  - Our application-aware SDN routing

- Performance metrics:
  - Total shuffle time of each scheme
  - Shuffle-Time-Reduction-Percentage (STRP) of our scheme compared to each of the other schemes
Experimental Setup

- 16 Hadoop servers, 1 Floodlight SDN controller, and 1 EstiNet Emulator

This OpenFlow network is emulated by a machine running EstiNet.

These servers are all real servers. They all connect to the EstiNet machine.
Controlled-Variable Experiments

- We run the “WordCount” job with artificial input data to precisely control the values of variables.
- The goal is to see the impacts of these variables on the performance of the four compared routing schemes.
- Three variables are studied:
  - The number of Reducers (note that the number of “Mapper” are always fixed to 16)
  - The transfer file size ratios (One “Reducer” needs to receive N times the amount of data that other Reducers need to receive.)
  - The number of heavy-weight Reducers (A heavy-weight “Reducer” receives 2 times the amount of data that other Reducer need to receive.)
Experiment 1: Shuffle Data with Different Number ofReducers

Mapper

Mapper

Mapper

Reducer

Reducer

Receive 3.5 MB from each Mapper

Receive 3.5 MB from each Mapper

N = 1 to 16

Receive 3.5 MB from each Mapper
Floodlight Only Uses a Few Links

- Floodlight builds a broadcast tree for each server. But most of them are common.
- The following shows all of the used links in the UP direction. Not many links in the network are used.
ECMP’s Local Decision on Each Switch May Result in Bottlenecks Ahead

- E.g., For the connections from server 3 to 5 and from 4 to 6, the bottleneck is on the link in pod 1, which cannot be foreseen in pod 0.
Application-Aware SDN Routing Can Achieve the Best Performance

- Application-Aware SDN routing has moved the places of performance bottleneck to the links between the edge-layer switches and servers, which is already the best arrangement.
Experiment 2: Shuffle Data with Different File Size Ratios

- Mapper
- Mapper
- Mapper

Reducer
Reducer
Reducer

Receive 3.5 MB from each Mapper
Receive 3.5 MB from each Mapper
Receive 3.5* N MB from each Mapper
Total shuffle time

STRP of our method
Experiment 3: Shuffle Data with Different Number of Heavy-Weight Reducers

Mapper

Mapper

Mapper

Reducer

Reducer

Receive 1.75 MB from each Mapper

Receive 3.5 MB from each Mapper

16 - N

N = 1 to 16
Total shuffle time

![Graph showing total shuffle time for Spanning Tree, ECMP-RR, Floodlight, and Application-Aware SDN.](image)

STRP of our method

![Graph showing STRP (%) relative to Spanning Tree, ECMP-RR, and Floodlight.](image)
Real-World Data Experiment

- Run the “WordCount” job with real-world input data
- We use all (over 6,500) RFC files as the input data.

![Diagram of Mappers and Reducer]

- Mappers: 16 Mappers
- Reducer: Receives data from each Mapper
- N = 1 to 16
- Mappers send data to the Reducer
Total shuffle time

STRP of our method
Conclusion

- Our application-aware SDN routing can reduce the shuffle time by up to 70% when compared with the spanning tree protocol or the Floodlight SDN controller, none of them is application-aware.

- ECMP can perform better than the spanning tree protocol and the Floodlight SDN controller by up to 50% because it uses multiple paths at the same time.
Conclusion (cont’d)

- Our application-aware SDN routing scheme can outperform ECMP by about 20% because ECMP can only make local decisions without a global view.
- An application-aware SDN routing scheme can greatly reduce the data shuffle time of a big-data application and it is useful.
Future Work

- The current path allocation procedure is an iterative method. It allocates paths one-by-one based on the most recent load (traffic volume) condition.
- To save time, it does not try all combinations to search for the best allocation plan. Thus, it may not be optimal and can be improved further.
- Different big-data applications (e.g., medical data, financial data, IoT data, etc.) may exhibit different shuffling traffic volume patterns.
- Instead of using a general-purpose path allocation procedure, using a special-purpose path allocation procedure may generate more performance boost.
Thank you for your attention.

shieyuan@cs.nctu.edu.tw