Boosting Distributed Machine Learning System with Application-aware Network Scheduling

Zheng Luo

April 10, 2018

Background & Previous Research

Big Data With SDN

Big Data applications[1]:

- Require high bandwidth
- Have well-defined computational/traffic patterns
- Have a centralized management structure

Therefore, it is possible to leverage application-aware information to improve network scheduling, especially in a software-defined network.

[1] Programming Your Network at Run-time for Big Data Applications, HotSDN, Guohui Wang et. al., 2013

Compared with Our Last Research

In our last paper *BAHS: A Bandwidth-Aware Heterogeneous Scheduling Approach for SDN-based Cluster Systems*, we used *network information* to support *task scheduling in Hadoop System*.

What about the reverse?

> *Big data can also benefit SDN, including traffic engineering, cross-layer design, defeating security attacks, and SDN-based intra- and inter-data-center networks.*[1]

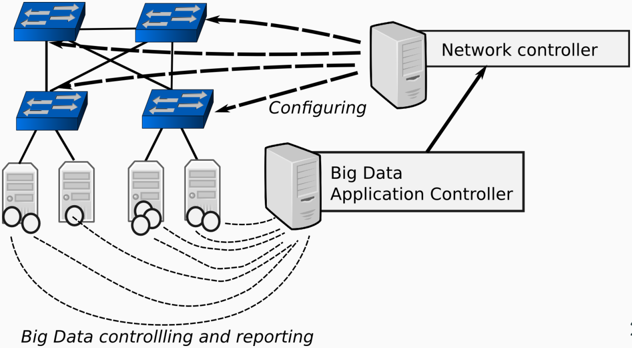
Why the reverse is better:

> *Today, application schedules tasks in a network-agnostic way*[2]

[1] When big data meets software-defined networking: SDN for big data and big data for SDN, Laizhong Cui et. al., IEEE Journal of Networks, 2016

[2] Cross-Layer Scheduling in Cloud Systems, Hilfi Alkaff et. al., IEEE International Conference on Cloud Engineering, 2015

Application-aware scheduling

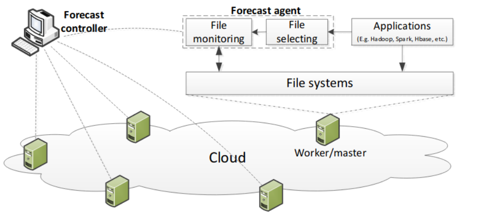


[2]

[2] Network Configuration and Flow Scheduling for Big Data Applications, Lautaro Dolberg et. al., Networking for Big Data, 2015

A typical example: HadoopWatch

HadoopWatch[3] is a typical example to perform such cross-layer network scheduling:



- Predict task status by monitoring file system
- Calculate task & flow dependencies
- Assign higher priority to slow/blocking flows

[3] HadoopWatch: A first step towards comprehensive traffic forecasting in cloud computing, Yang Peng et. al., INFOCOM, 2014

After HadoopWatch

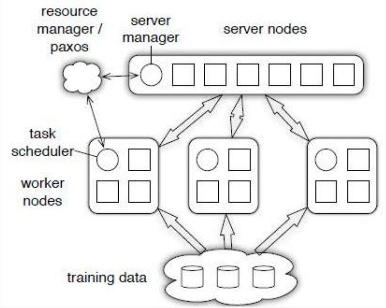
A lot of research on application-aware scheduling conducted on Hadoop, Storm and Flink.

Basically consisting three parts:

- Predict flows & Aggregate multiple flows into an application-level task
- Calculate priorities of flows based on application-level information
- Change routing configuration on switches to achieve QoS scheduling

Incorporate application-aware scheduling to distributed machine learning

Parameter server architecture



- Require high bandwidth? YES. Real-world model contains 10^9 to 10^{12} parameters. [4]
- Well-defined patterns? YES. Worker pull parameters from server, and push gradients to server in a batch.
- Centralized architecture? YES. Most use a single scheduler.

[4] Scaling Distributed Machine Learning with the Parameter Server, Mu Li et. al., OSDI, 2014

Proposed methods

Expect to conduct experiments on Apache MXNet[5], a widely-used distributed machine learning framework.

Typical workflow for a worker:

- (Batch 1) Pull parameter
- Calculation
- Push gradients to parameter server
- (Wait for all other worker to finish batch 1)
- (Server updating parameters)
- (Batch 2) Pull parameter...

[5] MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems, Tianqi Chen et. al., NIPS LearningSys, 2016

Proposed methods (2)

Easy to categorize flows:

- pull flow = server to worker
- push flow = worker to server
- Batch number are explicitly output to worker's logs Late batch (=smaller batch number) has higher priority. Pull flows > Push flows in the same batch

Easy to determine the priority of flows:

- Smaller batch number -> higher priority
- Pull flows > push flows in the same batch

Proposed architecture