Orchestra

Managing Data Transfers in Computer Clusters

Mosharaf Chowdhury, Matei Zaharia, Justin Ma, Michael I. Jordan, Ion Stoica
Moving Data is Expensive

Typical MapReduce jobs in Facebook spend 33% of job running time in large data transfers

Application for training a spam classifier on Twitter data spends 40% time in communication
Limits Scalability

Scalability of Netflix-like recommendation system is bottlenecked by communication

Did not scale beyond 60 nodes
» Comm. time increased faster than comp. time decreased
Transfer Patterns

Transfer: set of all flows transporting data between two stages of a job
» Acts as a barrier

Completion time: Time for the last receiver to finish
Contributions

1. Optimize at the level of transfers instead of individual flows

2. Inter-transfer coordination
Orchestra

Inter-Transfer Controller (ITC)

Shuffle Transfer Controller (TC)

Broadcast Transfer Controller (TC)

Broadcast Transfer Controller (TC)

shuffle

broadcast 1

broadcast 2
Cornet: Cooperative broadcast

Broadcast same data to every receiver
  » Fast, scalable, adaptive to bandwidth, and resilient

Peer-to-peer mechanism optimized for cooperative environments

<table>
<thead>
<tr>
<th>Observations</th>
<th>Cornet Design Decisions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. High-bandwidth, low-latency network</td>
<td>✓ Large block size (4-16MB)</td>
</tr>
<tr>
<td>2. No selfish or malicious peers</td>
<td>✓ No need for incentives (e.g., TFT)</td>
</tr>
<tr>
<td></td>
<td>✓ No (un)choking</td>
</tr>
<tr>
<td></td>
<td>✓ Everyone stays till the end</td>
</tr>
<tr>
<td>3. Topology matters</td>
<td>✓ Topology-aware broadcast</td>
</tr>
</tbody>
</table>
Cornet performance

1GB data to 100 receivers on EC2

Status quo

Completion time (s)

HDFS (R=3)  HDFS (R=10)  BitTornado  Tree (D=2)  Chain  Cornet

4.5x to 5x improvement
Topology-aware Cornet

Many data center networks employ tree topologies

Each rack should receive exactly one copy of broadcast
  » Minimize cross-rack communication

Topology information reduces cross-rack data transfer
  » Mixture of spherical Gaussians to infer network topology
Topology-aware Cornet

200MB data to 30 receivers on DETER

~2x faster than vanilla Cornet
Status quo in Shuffle

Links to $r_1$ and $r_2$ are full: 3 time units
Link from $s_3$ is full: 2 time units

Completion time: 5 time units
Weighted Shuffle Scheduling

Allocate rates to each flow using weighted fair sharing, where the weight of a flow between a sender-receiver pair is proportional to the total amount of data to be sent.

Completion time: 4 time units

Up to 1.5X improvement
Inter-Transfer Controller
aka Conductor

Weighted fair sharing
» Each transfer is assigned a weight
» Congested links shared proportionally to transfers’ weights

Implementation: Weighted Flow Assignment (WFA)
» Each transfer gets a number of TCP connections proportional to its weight
» Requires no changes in the network nor in end host OSes
Benefits of the ITC

Shuffle using 30 nodes on EC2

Two priority classes
  » FIFO within each class

Low priority transfer
  » 2GB per reducer

High priority transfers
  » 250MB per reducer

43% reduction in high priority transfers
6% increase of the low priority transfers
End-to-end evaluation

Developed in the context of Spark – an iterative, in-memory MapReduce-like framework

Evaluated using two iterative applications developed by ML researchers at UC Berkeley

» Training spam classifier on Twitter data
» Recommendation system for the Netflix challenge
Faster spam classification

Communication reduced from 42% to 28% of the iteration time

Overall 22% reduction in iteration time
Scalable recommendation system

Before

<table>
<thead>
<tr>
<th>Number of machines</th>
<th>Iteration time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>200</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>60</td>
<td>50</td>
</tr>
<tr>
<td>90</td>
<td>25</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th>Number of machines</th>
<th>Iteration time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>200</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
</tr>
<tr>
<td>60</td>
<td>50</td>
</tr>
<tr>
<td>90</td>
<td>25</td>
</tr>
</tbody>
</table>

1.9x faster at 90 nodes
Related work

DCN architectures (VL2, Fat-tree etc.)
  » Mechanism for faster network, not policy for better sharing

Schedulers for data-intensive applications (Hadoop scheduler, Quincy, Mesos etc.)
  » Schedules CPU, memory, and disk across the cluster

Hedera
  » Transfer-unaware flow scheduling

Seawall
  » Performance isolation among cloud tenants
Summary

Optimize transfers instead of individual flows
  » Utilize knowledge about application semantics

Coordinate transfers
  » Orchestra enables policy-based transfer management
  » Cornet performs up to 4.5x better than the status quo
  » WSS can outperform default solutions by 1.5x

No changes in the network nor in end host OSes
BACKUP SLIDES
MapReduce logs

Weeklong trace of 188,000 MapReduce jobs from a 3000-node cluster

Maximum number of concurrent transfers is several hundreds

33% time in shuffle on average
Monarch (Oakland'11)

Real-time spam classification from 345,000 tweets with urls
  » Logistic Regression
  » Written in Spark

Spends 42% of the iteration time in transfers
  » 30% broadcast
  » 12% shuffle

100 iterations to converge
Collaborative Filtering

Does not scale beyond 60 nodes

385MB data broadcasted in each iteration
Cornet performance

1GB data to 100 receivers on EC2

4.5x to 6.5x improvement
Shuffle bottlenecks

An optimal shuffle schedule must keep at least one link fully utilized throughout the transfer
Current implementations

Shuffle 1GB to 30 reducers on EC2