Saath: Speeding up CoFlows by Exploiting the Spatial Dimension

Akshay Jajoo  Rohan Gandhi  Y. Charlie Hu  Chengkok-Koh
Analytics Jobs in Big Data

• Analytics jobs in data-centers
  – Process huge amount of data
  – Distributed in nature
  – Have multiple stages that communicate with each other
Two compute stages: Map and Reduce

Map communicates with reduce in shuffle phase
Impact of communication on job performance

Facebook jobs spend **25%** time in communication![1]

[1] Based on information from full facebook trace used in Aalo. Aalo slides.
CoFlow abstraction

**CoFlow:** Collection of all flows that share same goal

**Implication:** CoFlow finishes when all its flows are over

**CoFlow Completion Time (CCT):** Completion time of its last flow

---

CoFlow Scheduling Problem

Green Job 2 mappers and 2 reducers

Orange Job 4 mappers and 2 reducers

They share datacenter network

Goal: minimize average CoFlow completion time (CCT) of all CoFlows
CoFlow Scheduling Problem

• CoFlow scheduling problem
  • Minimize average CoFlow Completion Time (CCT)
• CoFlows have 2-dimensions
  • Time – Length of individual flows
  • Space – Many flows or ports
• CoFlow scheduling problem is NP Hard [3]

Outline

• Background of Aalo (State-of-the-art CoFlow scheduler)
• Limitations of Aalo
• Design of Saath
• Evaluation
Background of Aalo (State-of-the-art CoFlow scheduler)

– Shortest job first for sequential jobs
– Online approximation of SJF
– Aalo: Online SJF + Spatial dimension (many distributed tasks)
Scheduling 101

Shortest-Job-First (SJF): optimal in minimizing average completion time

Process Scheduling

Last

$P_3$ $P_2$ $P_1$

First

$P_1 < P_2 < P_3$
Outline

• Background of Aalo (State-of-the-art CoFlow scheduler)
  – Shortest job first for sequential jobs
  – Online approximation of SJF
  – Aalo: Online SJF + Spatial dimension (many distributed tasks)
Online Approximation to SJF using Priority queues

Process durations - Unknown

Priority queues
(Higher Priority = more CPU time)

Shorter processes finish in High priority queues
Outline

• Background of Aalo (State-of-the-art CoFlow scheduler)
  – Shortest job first for sequential jobs
  – Online approximation of SJF
  – Aalo: Online SJF + Spatial dimension (many distributed tasks)
Datacenter Network abstraction: Non-blocking switch

• The entire datacenter fabric is one non-blocking switch
  – Makes analysis simple
  – Recent works like CONGA[Sigcomm’ 14], VL2 [Sigcomm’09] make the abstraction practical
• Only source of contention are end-hosts
• Implication: The CoFlow scheduling problem boils down to ordering them at sending hosts/ports

Aalo: Online CoFlow Scheduler

A CoFlow has many flows -- How to approximate SJF?

1. Replicates priority queues at each node
2. A CoFlow moved across priority queues based on total bytes sent at all its ports
3. Different ports send independently
4. Intra-queue: Use FIFO

Outline

• Background of Aalo (State-of-the-art CoFlow scheduler)
• Limitations of Aalo
• Design of Saath
• Evaluation
Aalo Drawback 1: Out-of-Sync

Ports schedule independent of each other

Flows of a CoFlow may get scheduled at different times at different ports
Aalo Drawback 2: Contention Oblivion

Contention of a CoFlow – Number of other CoFlows it blocks

- C1 – 1 – C2
- C2 – 2 – C1 & C3
- C3 – 1 – C2

Average CCT = (2+1+2)/3 = 5/3

Average CCT = (1+2+1)/3 = 4/3
Aalo is not taking arrangement of CoFlows across Space into account
Outline

• State-of-the-art CoFlow scheduler - Aalo
• Limitations of Aalo
• Design of Saath
• Evaluation
Saath:
Speeding up CoFlows by exploiting the Spatial Dimension
Saath

• Saath is an online scheduler.
• Takes spatial dimension into account while scheduling CoFlows.
• Spatial dimension: Arrangement of flows of CoFlows across ports
Saath Key Ideas

• All-or-none
• Least-Contention-First within a queue
• Faster CoFlow-queue transition
Key idea 1: All-or-none

• Either schedule all flows of a CoFlow or schedule none.
  – Not scheduling a CoFlow for which a subset of flow was being scheduled has no effect on CCT.
  – By freeing up some ports we potentially improve CCT for others
Challenges in All-or-none

CCT:
\[ C_1 = t, \quad C_2 = 2t, \quad C_3 = C_4 = t \]

Saath handles low port utilization by carefully designed work conservation
Key idea 2: Least-Contention-First within a queue

- *Contention* of a CoFlow – Number of other CoFlows it blocks
- Saath sorts CoFlows in each queue in increasing order of *Contention*
- Allows more CoFlows to be scheduled in parallel.
Key idea 2: Least-Contention-First within a queue

*Contention* of a CoFlow – Number of other CoFlows it blocks

- C1 – 1 – C2
- C2 – 2 – C1 and C3
- C3 – 1 – C2

Average CCT = (1+2+2)/3 = 5/3

Average CCT = (1+2+1)/3 = 4/3
Key idea 3: Faster CoFlow-queue transition

- Both Aalo and Saath use priority-queue structure to move CoFlows across queues
- Aalo uses total bytes by all flows
- Saath uses bytes per flow
- Saath has fast transition of longer CoFlows to lower priority queue
Recap: Saath Scheduling Ideas

• All-or-none
• Least-Contention-First within a queue
• Faster CoFlow-queue transition
Outline

• State-of-the-art CoFlow scheduler - Aalo
• Limitations of Aalo
• Design of Saath
• Evaluation
Evaluation Methodology

1. Large scale trace driven simulations
2. Large scale testbed evaluation - 150 nodes
3. Implemented Saath in 5.2 KLoC in C++
Trace

1. FB Trace[7]
   1. Collected from Facebook’s cluster.
   2. 526 CoFlows, 150 ports

2. OSP
   1. Collected from Microsoft’s cluster.
   2. $O(1000)$ CoFlows, $O(100)$ ports

[7]: https://github.com/coflow/coflow-benchmark
Overall CCT improvement

- Saath approaches offline SEBF
- 1.53x for FB and 1.42x for OSP median speedup as compared to Aalo
• Each design component has considerable contribution in CCT improvement.
Things are In-Sync now

- Most of the equal flow coflow now have very small deviation in FCTs
Testbed

CCT Speedup 1.88x on Average and 1.43x P50
# Scheduling Overhead

<table>
<thead>
<tr>
<th></th>
<th>SAATH</th>
<th>Aalo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>P90</td>
</tr>
<tr>
<td><strong>Global Coordinator</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU %</td>
<td>37.8</td>
<td>42.7</td>
</tr>
<tr>
<td>Memory(MB)</td>
<td>229</td>
<td>284</td>
</tr>
<tr>
<td>Total time (msec)</td>
<td>0.57</td>
<td>2.85</td>
</tr>
<tr>
<td>(LCoF/All-or-none)</td>
<td>(0.02/0.24)</td>
<td>(0.03/0.7)</td>
</tr>
<tr>
<td><strong>Local Node</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU %</td>
<td>5.6</td>
<td>5.7</td>
</tr>
<tr>
<td>Memory(MB)</td>
<td>1.68</td>
<td>1.7</td>
</tr>
</tbody>
</table>
Conclusion

• CoFlow scheduling holds promise to optimize communication in Big Data jobs

• Limitation of prior-art Aalo:
  – Ignores spatial arrangement
  – Has no coordination across ports
    • Flows can be out of sync
    • CoFlow contention oblivious

• Saath:
  – Fuses spatial dimension in CoFlow scheduling
  – Coordination across ports
  – Evaluation: CCT improvement: 1.53x (P50) and 4.5x (P90) for FB trace and 1.42x (P50) and 37x (P90)
Thank you!