Varys

Efficient Coflow Scheduling

Mosharaf Chowdhury,
Yuan Zhong, Ion Stoica
As in-memory systems proliferate, the network is likely to become the primary bottleneck.
Flow

A sequence of packets between two endpoints

Independent unit of allocation, sharing, load balancing, and/or prioritization

Optimizing Communication Performance: Networking Approach

“Let systems figure it out”
## Optimizing Communication Performance: Systems Approach

“Let users figure it out”

<table>
<thead>
<tr>
<th>System</th>
<th># Comm. Params</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spark 1.0.1</td>
<td>6</td>
</tr>
<tr>
<td>Hadoop 1.0.4</td>
<td>10</td>
</tr>
<tr>
<td>YARN 2.3.0</td>
<td>20</td>
</tr>
</tbody>
</table>

*Lower bound. Does not include many parameters that can indirectly impact communication; e.g., number of reducers etc. Also excludes control-plane communication/RPC parameters.*
Optimizing Communication Performance: Systems Approach

“Let users figure it out”

Optimizing Communication Performance: Networking Approach

“Let systems figure it out”
Optimizing Communication Performance: Systems Approach

"Let users figure it out"

Optimizing Communication Performance: Networking Approach

"Let systems figure it out"
Optimizing Communication Performance: Networking Approach

"Let systems figure it out"

"Let users figure it out"

A collection of parallel flows

Distributed endpoints

Each flow is independent

Completion time depends on the last flow to complete

---

Coflow

A collection of parallel flows

Distributed endpoints

Each flow is independent

Completion time depends on the last flow to complete

How to schedule coflows ...

... for faster completion of coflows?

... to meet more deadlines?
Varys

Enables coflows in data-intensive clusters

1. Simpler Frameworks
   Zero user-side configuration using a simple coflow API

2. Better performance
   Faster and more predictable transfers through coflow scheduling
Benefits of Inter-Coflow Scheduling

Coflow 1

Link 2

Link 1

3 Units

Coflow 2

6 Units

3-ε Units

Fair Sharing

Flow-level Prioritization\(^{1,2}\)

The Optimal

Coflow 1 comp. time = 6
Coflow 2 comp. time = 6

Coflow 1 comp. time = 6
Coflow 2 comp. time = 6

Coflow 1 comp. time = 3
Coflow 2 comp. time = 6

Inter-Coflow Scheduling

Concurrent Open Shop Scheduling\(^1\)

- Tasks on independent machines
- Examples include job scheduling and caching blocks
- Use a ordering heuristic

---

Inter-Coflow Scheduling is NP-Hard

<table>
<thead>
<tr>
<th>Link 2</th>
<th>Link 1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>3 Units</strong></td>
</tr>
</tbody>
</table>

with coupled resources

Concurrent Open Shop Scheduling

- Flows on dependent links
- Consider ordering and matching constraints

Characterized COSS-CR

Proved that list scheduling might not result in optimal solution
Varys employs a two-step algorithm to minimize coflow completion times.

1. Ordering heuristic
   - Keeps an ordered list of coflows to be scheduled, preempting if needed

2. Allocation algorithm
   - Allocates minimum required resources to each coflow to finish in minimum time
Ordering Heuristic: **SEBF**

<table>
<thead>
<tr>
<th></th>
<th>$C_1$</th>
<th>$C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Width</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Size</td>
<td>5</td>
<td>12</td>
</tr>
<tr>
<td>Bottleneck</td>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

**Shortest-First**

**Narrowest-First**

**Smallest-First**
A coflow cannot finish before its very last flow.

Finishing flows faster than the bottleneck cannot decrease a coflow’s completion time.

Ensure minimum allocation to each flow for it to finish at the desired duration; for example, at bottleneck’s completion, or at the deadline.
Varys

Enables frameworks to take advantage of coflow scheduling

1. Exposes the coflow API
2. Enforces through a centralized scheduler
1. Does it improve performance?

2. Can it beat non-preemptive solutions?

YES

Evaluation

A 3000-node trace-driven simulation matched against a 100-node EC2 deployment.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Avg.</strong></td>
<td>1.85X</td>
<td>1.25X</td>
</tr>
<tr>
<td><strong>95\text{th}</strong></td>
<td>1.74X</td>
<td>1.15X</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Avg.</strong></td>
<td>3.16X</td>
<td>2.50X</td>
</tr>
<tr>
<td><strong>95th</strong></td>
<td>3.84X</td>
<td>2.94X</td>
</tr>
</tbody>
</table>

1. 26% jobs spend at least 50% of their duration in communication stages.
Better than Non-Preemptive Solutions

<table>
<thead>
<tr>
<th></th>
<th>w.r.t. FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg.</td>
<td>5.65X</td>
</tr>
<tr>
<td>95th</td>
<td>7.70X</td>
</tr>
</tbody>
</table>

NO

Perpetual Starvation?

---

1. Managing Data Transfers in Computer Clusters with Orchestra, SIGCOMM'2011
#1 Coflow Dependencies

#2 Unknown Flow Information

#3 Decentralized Varys

in the Context of *Multipoint-to-Multipoint* Coflows
Theory Behind “Concurrent Open Shop Scheduling with Coupled Resources”
Variys

Greedily schedules coflows without worrying about flow-level metrics

- Consolidates network optimization of data-intensive frameworks
- Improves job performance by addressing the COSS-CR problem
- Increases predictability through informed admission control

http://varys.net/

Mosharaf Chowdhury - @mosharaf