Naiad: A Timely Dataflow System

Indigo Orton – R244
Motivation

- High throughput
- Low latency
- Interactive querying
Example – Analytics dashboard

• Constant metric streams – *stream*

• Automated insights – *stream + batch*

• Interactive user queries – *interactive*
Key idea

• Records traveling through a graph
• “Timely dataflow”
• Timestamps - *progressive record ids*
• Timestamps - *loop counters*
Graph model

• Graph based computation model
• Enable loops within graph
• Highly parallel stream processing
Data integrity

• Process records in epoch order

• Notifications to vertices – *i.e. flushing*

• Calculation of possible records
Limitation - Micro-stragglers

• Micro-stragglers – outsized performance impact
• Mutable shared state for low latency
• In-memory datasets
Results

Throughput

Latency

Twitter

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>PDW</th>
<th>DryadLINQ</th>
<th>SHS</th>
<th>Naiad</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageRank</td>
<td>156,982</td>
<td>68,791</td>
<td>836,455</td>
<td>4,656</td>
</tr>
<tr>
<td>SCC</td>
<td>7,306</td>
<td>6,294</td>
<td>15,903</td>
<td>729</td>
</tr>
<tr>
<td>WCC</td>
<td>214,479</td>
<td>160,168</td>
<td>26,210</td>
<td>268</td>
</tr>
<tr>
<td>ASP</td>
<td>671,142</td>
<td>749,016</td>
<td>2,381,278</td>
<td>1,131</td>
</tr>
</tbody>
</table>

Context

- Vertex centric computation models - Pregel [2]
- Straggler mitigation a higher priority in some systems – RDD [5], D-Streams [6] (based on RDD).
- Later systems decouple processing and coordination for faster cluster adaption – Drizzle [7]
- Updates to Naiad – last public commit in 2014 [3]
- Industry projects – Apache Flink™ [8]
Review
Encouraging highlights

• Graphs as a computational dependency model
• Modulization of computations
• Streaming, batch, and interactive support
Concerns

• Micro-stragglers – inability to mitigate
• Unsuitable for memory intensive computations
• Addressed via implementation optimisation
• Implementation approach and allocation of research resources
• Unnecessary complexity – timestamps/notifications
The paper

- Unnecessary complexity
  - Timestamps – *progressive ids*
  - Notifications – *flushing*
- Focus on implementation optimisations
The space – further discussion

- Nothing solves specifically for our target
- Collaboration between frameworks
- New framework that will not collaborate
- Generic protocol
- Jack of all trades, master of none
Conclusion

• Interesting model
• Modulization – global coordination
• Risks with micro-stragglers
• Unnecessary complexity
• Time spent on implementation optimisations
• Young field - or fundamentally unsolvable?
References


3. Naiad open source repository – Accessed 15/10/18 – [https://github.com/MicrosoftResearch/Naiad](https://github.com/MicrosoftResearch/Naiad)


