Naiad: A Timely Dataflow System
Daniel Vlasits
User queries are received

Low-latency query responses are delivered

Queries are joined with processed data

Updates to data arrive

Complex processing incrementally re-executes to reflect changed data
What is it?

- High throughput
- Low latency
- Iterative + Incremental

Examples
- MapReduce
- Time Stream
- Giraffe
Timely Dataflow

• Developed to handle continuous data to be processed fast
\( \nu.\text{ONRECV}(e : \text{Edge}, \ m : \text{Message}, \ t : \text{Timestamp}) \)
\( \nu.\text{ONNOTIFY}(t : \text{Timestamp}). \)

A vertex may invoke two system-provided methods in the context of these callbacks:

\( \text{this.SENDBY}(e : \text{Edge}, \ m : \text{Message}, \ t : \text{Timestamp}) \)
\( \text{this.NOTIFYAT}(t : \text{Timestamp}). \)
Example Program – Incrementally updatable MapReduce

// 1a. Define input stages for the dataflow.
var input = controller.NewInput<string>();

// 1b. Define the timely dataflow graph.
// Here, we use LINQ to implement MapReduce.
var result = input.SelectMany(y => map(y))
   .GroupBy(y => key(y),
            (k, vs) => reduce(k, vs));

// 1c. Define output callbacks for each epoch
result.Subscribe(result => { ... });

// 2. Supply input data to the query.
input.onNext(/* 1st epoch data */);
input.onNext(/* 2nd epoch data */);
input.onNext(/* 3rd epoch data */);
input.onCompleted();

• Constructing the graph
Metrics Considered

- Throughput
- Latency
- Scaling

- Very impressive

<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>

Table 1: Running times in seconds of several graph algorithms on the Category A web graph. Non-Naiad measurements are due to Najork et al. [34].

- Example given is computing the most popular hashtag.
The Diversity of Applications

• WordCount – Computing word frequencies of Twitter Corpus
• WCC – weakly connected components

• Does a reasonable job of scaling
• Batch computation
• Streaming Computation
• Graph Computation
• All can be expressed at high-level using the Naiad framework
Takeaways

• Decoupling high-level ideas from low-level implementations
• Very fast
• Useful for an incredibly wide range of applications
• Just because we’ve just covered garbage collection in PL – specific effort made to make the life of collector as easy as possible
• Tradeoffs hardly mentioned, but they prefer performance over restoring from crash – less logging more updating – spark is better