CIEL : A Universal Execution Engine

G Roshan Lal
What is CIEL?

- Computation Engine
- Distributed Data-Flow
- Universal

- Other Distributed Execution Engines…..Data-Flow
  - MapReduce……..Bi-Partite Graph
  - Dryad…………….DAG
Why CIEL?

• MapReduce/Dryad is

• Good For:
  - Batch-Oriented
  - Eg. Info Retrieval
  - High Throughput

• Bad For:
  - Iterative
  - Eg. ML Training
  - Low Latency, chained events
How CIEL works?

- Dynamic Task Graphs

![Dynamic Task Graph](image)
How CIEL works?

- Task / Object Tables

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Dependencies</th>
<th>Expected outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>{ u }</td>
<td>z</td>
</tr>
<tr>
<td>B</td>
<td>{ v }</td>
<td>x</td>
</tr>
<tr>
<td>C</td>
<td>{ w }</td>
<td>y</td>
</tr>
<tr>
<td>D</td>
<td>{ x, y }</td>
<td>z</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Object ID</th>
<th>Produced by</th>
<th>Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>-</td>
<td>{ host19, host85 }</td>
</tr>
<tr>
<td>v</td>
<td>-</td>
<td>{ host21, host23 }</td>
</tr>
<tr>
<td>w</td>
<td>-</td>
<td>{ host22, host57 }</td>
</tr>
<tr>
<td>x</td>
<td>B</td>
<td>\emptyset</td>
</tr>
<tr>
<td>y</td>
<td>C</td>
<td>\emptyset</td>
</tr>
<tr>
<td>z</td>
<td>A, D</td>
<td>\emptyset</td>
</tr>
</tbody>
</table>

(b) Task and object tables
How CIEL works?

- **Objects**: Named collection of Bytes

- **References**: Obj Name, Physical Location (maybe empty….not yet created)

- **Tasks**
  - Publish output objects (or)
  - Spawn new Tasks depending on output objects
  - Dynamic DAG…. Prevents Deadlock
  - Lazy Evaluation… Easier fault Tolerance
CIEL Architecture

- **Single Master**
  - Keeps record of obj and task tables
  - Dispatches tasks to workers

- **Multiple Workers**
  - HeartBeat messages for availability to Master
  - Update Master with Spawn/Publish
CIEL Fault Tolerance

- **Workers:**
  - HeartBeat from Workers/ Response to Master
  - Re-schedule

- **Master:**
  - Persistent Logging of Tables
CIEL Optimizations

- Long tails of recursion:
  - Assume deterministic behaviour of tasks
  - Memoization of output for given input

- Persistent logging of (Single) Master:
  - Secondary Masters
  - Reconstruct from Workers

- Skywriting:
  - Scripting Language for CIEL
Conclusion

- CIEL performs well on all types of loads

- MapReduce Types: grep

- Iterative Types: k-means

- Compute Intensive Types: Smith-Waterman