Nebula: Distributed Edge Cloud for Data-Intensive Computing

Mathew Ryden, Kwangsung Oh, Abhishek Chandra and Jon Weissman

Department of Computer Science and Engineering
University of Minnesota

Scheduling for Large Scale Systems Workshop, Lyon 2014
Geo-Distributed Data

• Across several domains:
  – Social. e.g.: User tweets, blog postings
  – Mobile. e.g.: Phone pics, sensor data
  – Access logs. e.g.: CDNs
  – Scientific. e.g.: Weather, seismic data
  – Commercial. e.g.: Customer, inventory data

• Data analysis is key:
  – Trend analysis, network monitoring, scientific data analysis, accounting/billing, ...
Traditional Compute Model

• Centralized:
  – Push all data to a central location

• Challenges:
  – Data upload can be slow
  – Central point of failure
  – Cost of data upload and processing may be too high

• Observation: Not all analysis need be centralized!
Idea: Use the Edge

• **Capacity**
  – Folding@home: 350K CPUs, 4 Teraflops
  – Gnutella: 3M CPUs, several Petabytes
  – Akamai: ~140K servers

• **Locality**
  – Closer to data
  – Low latency to end-user

• **Resilience**
  – No single point of failure
  – Robust to network failures
Nebula: Decentralized Edge Cloud

• Exploit edge computing and storage capacity
  – Uses volunteer edge nodes

• Support distributed data-intensive applications
  – Strong data-compute interaction
  – Locality-awareness
Motivation

• **Nebula Architecture**
• MapReduce on Nebula
• Evaluation/Demo
• Scheduling Challenges
• Concluding Remarks
Nebula Architecture

Nebula Services
Dedicated Nodes

DataStore Master
Network Monitor
ComputePool Master
Nebula Central

Data Nodes

Volunteer Nodes

Compute Nodes
Nebula Services

- **DataStore: Storage layer**
  - Consists of Data nodes and DataStore Master
  - Supports locality-aware put and get operations

- **ComputePool: Computation layer**
  - Consists of Compute nodes and ComputePool Master
  - Nodes run Chrome native client (NaCl) for sandboxing
  - Supports locality-aware scheduling

- Masters are specific to the application framework
  - In this talk, MapReduce
Locality Awareness

- **Challenge:** Network may be bottleneck

- **DataStore:** Locality-aware `put/get`
  - Data nodes ordered by their locality (b/w, latency) to a client => uses IP geo-coordinates

- **ComputePool:** Locality-aware scheduler
  - Select compute node for a task, in part, based on estimate of data access cost to that node
Locality Aware Task Scheduler (LA)

- Schedule task $t$:

1. Estimate current network conditions for each link based on recent data transfers, $T_{\text{link}}(\text{dnode}, \ cnode)$

2. For each running task $t'$:
   - Estimate finishing time $T_{\text{fin}}(t', cnode)$

3. For each compute node $cnode$:
   - Estimate completion time:
     \[ T_{\text{cnode}} = T_{\text{exec}}(cnode, \ t) + \sum T_{\text{fin}}(t', cnode) \]

Select $\min T_{\text{cnode}}$ for $t$ subject to cap of $k$ tasks per node
Fault Tolerance

• **Challenge:** Node/link failures, node churn

• **DataStore (proactive):**
  – Replication of input and intermediate/output data
  – Programmer specifies # and system maintains it

• **ComputePool (reactive):**
  – Soft failures: Compute node retries task again
  – Hard failures: Re-execution of tasks on other nodes
MapReduce on Nebula

• DataStore and ComputePool Masters tuned to MapReduce
  – locality-awareness

• Consists of two job types:
  – Map and Reduce, each with multiple tasks
  – Reduce dependent on Map

• Input, intermediate, and output data managed by DataStore
Talk Outline

✓ Motivation
✓ Nebula Architecture
✓ MapReduce on Nebula

• Evaluation/Demo
• Scheduling challenges
• Concluding Remarks
Experimental Setup

• Deployed on PlanetLab
  – 52 machines in PL Europe
  – Bandwidth from 256Kbps - 32Mbps
  – UMN: Hosts dedicated services, e.g. masters

• Nebula applications
  – MapReduce: Wordcount, Inverted Index
  – Blog analysis
Alternate Distributed Models

Central Source Central Intermediate Data (CSCI)

Central Source Distributed Intermediate Data (CSDI)

BOINC

U Texas, Dahlin et al
Comparison to other Models

Nebula significantly outperforms other distributed computing models
Choice of Scheduler

Locality-awareness improves runtime by 16-34%
## Fault Tolerance

### Compute Node Failures

<table>
<thead>
<tr>
<th>% of Compute node failures</th>
<th>MAP</th>
<th>REDUCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Data Node Failures

<table>
<thead>
<tr>
<th>Number of Failures</th>
<th>MAP</th>
<th>REDUCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Nebula is robust to multiple failures
Demo

• Nebula demo
Nebula: Usage Scenarios

• In-situ data processing
  – Exploit locality of data and computation
• On-ramp to centralized cloud
  – Edge filtering, aggregation, sampling, ...
• Low cost cloud
  – If built using volunteer resources
Current Scheduling Research

• Ingesting data from outside nebula
  – How to co-schedule data + computation

“Run MapReduce using X mappers, Y reducers on K nodes on this file”

• Scheduling across the conventional cloud and nebula
Concluding Remarks

• Geo-distributed data increasingly common in many domains
  – Data movement to central cloud can be costly
• Nebula is a decentralized edge cloud
  – Designed to support data-intensive applications
  – Uses distributed compute and storage resources
  – Locality-aware and robust to failures
  – Many important scheduling challenges
Thanks!

http://dcsg.cs.umn.edu/
Nebula Implementation

- Has used several languages and technologies

<table>
<thead>
<tr>
<th>Component</th>
<th>Lines of Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Node</td>
<td>1272 (Java)</td>
</tr>
<tr>
<td>DataStore Master</td>
<td>736 (JavaScript)</td>
</tr>
<tr>
<td>ComputePool Master</td>
<td>750 (Python), 124 (JavaScript)</td>
</tr>
<tr>
<td>Scheduler</td>
<td>1068 (Python)</td>
</tr>
<tr>
<td>MapReduce NaCl Framework</td>
<td>1167 (C++)</td>
</tr>
<tr>
<td>WordCount Application</td>
<td>193 (C++), 149 (JavaScript)</td>
</tr>
</tbody>
</table>
Nebula Hybrids

- Hybrid paradigms (two directions)
  - not cloud bursting
  - early discard
  - fan-in