Comparing Spark GraphX and Cray Graph Engine using large-scale client data

Eric Dull and Brian Sacash
Deloitte Advisory, May 11, 2017
Topics

- Introductions
- Our collection and analysis environment
- Problem statement:
  - Experimental design
  - GraphX
  - CGE
- Weaknesses and “gotchas”
- Next steps
Introductions
Deloitte Advisory’s Cyber Reconnaissance team

Deloitte Advisory’s Cyber Reconnaissance team uses a combination of big data tools, data science, graph analytics, and supercomputing to uncover potential threat vectors or ongoing attacks

Eric Dull
Specialist Leader
Deloitte & Touche LLP
• Experience includes network analysis, applied graph analysis, behavior-based anomaly detection
• Prior CUG papers:
  • Cyberthreat analytics using graph analysis, CUG 2015

Brian Sacash
Specialist Senior
Deloitte & Touche LLP
• Data scientist who focuses on software development for analytic-based decision making
• Experience employing natural language processing, statistical analysis, and machine learning using big data technologies
Deloitte’s collection and analysis environment

Collectors (BRO)

SPARK
PARQUET
CGE

Build a graph
Execute Algorithm

Analyst

Results

Validated Findings
Data size

February Data:
- ~83,110,000,000 connection records
- ~1,800,000 unique clients
- ~55,000,000 unique external IPs

Source: Deloitte February data pull
Cray Urika-GX

What compute did we use for the experiments

Specifications
• 32 Blades
• 1000 cores
• 8 Terabytes of Ram
• 120 TB of Lustre
• 25 Blades available for Apache Spark

Additional details:
• Hosted in Deloitte’s Federal Technology Center in Suwanee, GA
• Used for multiple Spark work streams supporting multiple clients
Motivation
Connecting Cyber Kill Chain to Graph Algorithms

Cyber Kill Chain:
• External Reconnaissance
• Infection
• Lurking
• Activity

Applicable Graph algorithms:
• Community of Interest Identification
• Betweenness Centrality

Example Attack Model
Model

Target Graph Topologies

Betweenness Centrality: Which graph node has the most paths go through it?

Or, “All roads lead to Rome”
Experiment description
How did we execution on this vision?

Build a graph:
• Use network connection logs
• Focus on known behaviors

Run Betweenness Centrality:
• GraphX implementation
• Cray Graph Engine implementation

Validate Algorithm Results:
• Look for known scanners
• Analyst feedback
Graph Building
How did we build the graph?

Approach:
• Focus on TCP/UDP ports targeted by attackers
• Focus on successful connections
• Bring in multiple days

Observations:
• Successful connections reduces connection volumes by ~ 47%
• Targeted TCP ports (20, 21, 22, 23, 123, 445, 3389)
• Days remained in flux
GraphX results

How did GraphX perform?

Algorithm:
• No out-of-the-box implementation
• Spent time getting available 3rd party implementation running

Observations:
• GraphX did not perform above small graphs
• Observed variation in execution times likely related to network latency

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Edges</th>
<th>1 Node</th>
<th>4 Nodes</th>
<th>8 Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,419</td>
<td>11,726</td>
<td>54 seconds</td>
<td>50.1 seconds</td>
<td>71.8 seconds</td>
</tr>
<tr>
<td>42,687</td>
<td>125,564</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>
CGE results
How did the CGE implementation perform?

Algorithm:
• Cray Graph Engine provides betweenness centrality callable through Sparql
• CGE implementation uses directed edges, and traditional betweenness centrality is undirected

Observations:
• CGE ran, took longer than expected
• Performance did not scale well when given additional nodes

<table>
<thead>
<tr>
<th>Vertices</th>
<th>Edges</th>
<th>1 Node</th>
<th>8 Nodes</th>
<th>16 Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>5,419</td>
<td>11,726</td>
<td>2.6 seconds</td>
<td>29.4 seconds</td>
<td>29.3 seconds</td>
</tr>
<tr>
<td>15,359</td>
<td>52,042</td>
<td>77.7 seconds</td>
<td>653 seconds</td>
<td>718 seconds</td>
</tr>
<tr>
<td>42,687</td>
<td>125,564</td>
<td>354 seconds</td>
<td>1643 seconds</td>
<td>1891 seconds</td>
</tr>
<tr>
<td>66,955</td>
<td>369,720</td>
<td>938 seconds</td>
<td>4730 seconds</td>
<td>6600 seconds</td>
</tr>
<tr>
<td>115,276</td>
<td>1,281,918</td>
<td>3205 seconds</td>
<td>15068 seconds</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Observations, Weaknesses, and “gotchas”

• Analyst validation in progress

• Building meaningful graphs at scale is difficult

• CGE is easy to use, and some algorithms are in progress

• GraphX is hard to use
Next steps

• Further analyst validation

• Additional algorithms / use cases

• Hybrid architectures and workflows
This document contains general information only and Deloitte Advisory is not, by means of this document, rendering accounting, business, financial, investment, legal, tax, or other professional advice or services. This document is not a substitute for such professional advice or services, nor should it be used as a basis for any decision or action that may affect your business. Before making any decision or taking any action that may affect your business, you should consult a qualified professional advisor.

Deloitte Advisory shall not be responsible for any loss sustained by any person who relies on this document.