Blazegraph GPU-Accelerated Graph Query for Cyber Applications

- Blazegraph DASL

Jim Carbonaro, Sr Software Engineer, Blazegraph

April 2016
We enable Fortune 500 companies, startups, governments and research organizations achieve business objectives with graph data.

http://blazegraph.com/
Blazegraph™ Opensource Platform: Proven and in Production

200+ weekly downloads
Thousands of active deployments

Financial Services
Fraud Detection

Information Management Retrieval

Life Sciences
Precision Medicine

Powers Business with Billion Edge Datasets

Cyber Defense Intelligence

Industrial IoT

http://blazegraph.com/
Exploding Data Volumes Requires New Approach for Relationship Insight

Graph databases designed to analyze diverse entities and relationships. Today’s datasets have billions of edges and nodes.

A (very small) Knowledge Graph

http://blazegraph.com/
Scalability Challenge is the Blazegraph Opportunity

The Blazegraph Platform

- Embedded and Single Server Deployments
- Enterprise
- High Availability
- Enterprise
- Scale-out Architecture

Speed & Scale

- Data Scale (Edges)
  - Trillions
  - Billions
  - Millions

- Scale Out (1T+)
- Embedded Single Server (50B)
- High Availability (50B)

- Fast
- Speed
- Fastest

- Multi-GPU Clusters (100+B)
- Single GPU (500+M)
Blazegraph DASL for GPU Acceleration

Automation Enabling Analytics Coders to Optimize for GPU

Delivering ease of use of Spark and Scala for graph algorithms and predictive analytics… your analytic code can work on the Tesla GPUs and Blazegraph.

DASL Graph Algos

DASL Translator

DASL Executor Multi-GPU Extension

...with the speed of CUDA...

...Graph and Machine Learning Algorithms...

Blazegraph DASL – A Domain specific language for graphs with Accelerated Scala using Linear algebra

...you can see why we call it “DASL”
DASL your analytics

- DASL is functional, domain-specific language (DSL) for graph and machine learning algorithms
- Provides of linear algebra primitives supporting the DSL designed for efficient, multi-GPU execution
- Reduce the barrier of graphs on GPUs, enabling simpler and smarter algorithms
- Co-opt existing data ecosystems, such as Apache Spark and Hadoop, for ease of adoption and mission impact
Case Study - NetFlow Data Analysis

• What is NetFlow data?
  • NetFlow is a network protocol developed by Cisco for collecting IP traffic information and monitoring network traffic. [http://www.solarwinds.com/what-is-netflow.aspx](http://www.solarwinds.com/what-is-netflow.aspx)

• Sample NetFlow record
  ```json
  {
    "start_time": "2007-08-01 14:31:02.946",
    "duration": 0.000,
    "src_addr": "122.166.71.110",
    "dst_addr": "9.155.118.136",
    "src_port": 10822,
    "dst_port": 13567,
    "protocol": "UDP",
    "tcp_flags": "......",
    "input_packets": 1,
    "input_bytes": 46
  }
  ```

• How is NetFlow collected?
  • Generated by routers and forwarded to collection point
  • Processed out of pcap records using tools such as nfcapd/nfdump
DASL GPU-Accelerated Algorithms Applicable to NetFlow

• Path algorithms (BFS, SSSP, APSP, CC) and their extensions (closeness, betweenness, etc)

• Ranking algorithms (cardinality, PageRank, BadRank, SecureRank)

• Clustering algorithms (canopy, k-means, Jaccard, community detection, etc)

• Many others!
Challenges of Network Flow Analysis

• The volume of NetFlow packets could not previously be efficiently analyzed or visualized to identify threats in real time
  – One hour of collection on a modest network can easily generate more than 100M NetFlow records
  – Visualizing this data to identify points of interest is nearly impossible due to the “hairball effect”, even after time windowing the data
  – These issues largely led to attempts at batch processing the data on large clusters of machines using Hadoop and/or Spark
  – Pushing the problem into the batch space puts organizations in the unenviable position of hoping to identify why and by whom their network was attacked last month rather than being able to identify an attack as it is occurring.
How does Blazegraph DASL overcome these challenges?

• Graph algorithms are inherently iterative. CPU architectures lack the memory bandwidth to efficiently run iterative algorithms over large data sets.

• DASL overcomes this by pushing the computation to the GPU where bandwidth can be 10x that available on CPU architectures.

• DASL makes the algorithm development process easily accessible to non-CUDA specialists
Anatomy of a DASL algorithm

```scala
t// An interface for an SSSP algorithm implementation.
@throws(wmat: Matrix[Float], srcId: Int) extends DASLAlgorithm[Vector[Float]]
  def assumesDiagonalIsZero() = Boolean

/** Initialize the current-distance vector such that it uses the semiring
  * (Float, min, plus) as default semiring for the following computation
  * Populate the current-distance vector as follows: shortest path to the
  * source vertex is 0, all other elements will be set to the additive
  * identity of of the semiring, that is, Float.PositiveInfinity.
  */
  def init(daslContext: DASLContext) : Vector[Float] = 
    // An interface for a DASL algorithm implementation.
    dacslContext
    size = wmat.numOfRows,
    dfltSemiring = wmat.dfltSemiring, // MinPlusSemiring
    dfltValue = wmat.dfltSemiring, // MinPlusSemiring, addIty,
    data = ( srcId, GF )

/** Run the SSSP algorithm, returning the shortest paths. */
  def run( daslContext : DASLContext ) : Vector[Float]
```

http://blazegraph.com/
DASL vs Spark GraphX (PageRank)

Spark CPU Configuration

<table>
<thead>
<tr>
<th>Name</th>
<th>Qty</th>
<th>Total RAM (G)</th>
<th>Total CPU Cores</th>
<th>Instance Type</th>
<th>CPU Cores</th>
<th>Node RAM (G)</th>
<th>Storage (G)</th>
<th>Physical Processor</th>
<th>Clock (GHZ)</th>
<th>Chip Memory Bandwidth (G/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16-r3.4xlarge</td>
<td>1</td>
<td>122</td>
<td>16</td>
<td>EC2 r3.4xlarge</td>
<td>16</td>
<td>122</td>
<td>1 x 320 SSD</td>
<td>Intel Xeon E5-2670 v2 (Ivy Bridge)</td>
<td>2.5</td>
<td>59.7</td>
</tr>
</tbody>
</table>

http://blazegraph.com/
# DASL vs Spark GraphX (PageRank)

## DASL GPU Configuration

<table>
<thead>
<tr>
<th>Ubuntu Workstation (k40)</th>
<th>Total RAM (G)</th>
<th>Total CUDA Cores</th>
<th>Model</th>
<th>CUDA Cores</th>
<th>GPU RAM (G)</th>
<th>GPU Chip</th>
<th>Memory Bandwidth (G/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>12</td>
<td>2880</td>
<td>K40</td>
<td>2880</td>
<td>12</td>
<td>Tesla K40c</td>
</tr>
</tbody>
</table>
# DASL vs Spark GraphX (PageRank)

PageRank over Netflow Data (140051582 edges)

<table>
<thead>
<tr>
<th></th>
<th>Time (ms)</th>
<th>Speedup (“DASL PR is faster than”)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DASL PR</td>
<td>12841.00</td>
<td>0.000%</td>
</tr>
<tr>
<td>Spark PR</td>
<td>102836.93</td>
<td>800.84%</td>
</tr>
</tbody>
</table>

https://traces.simpleweb.org/traces/netflow/netflow1/
Visualizing NetFlow Data

• Graphistry uses GPU acceleration to allow graphs containing 1M+ edges to be visualized and manipulated by an analyst

• Combining the capabilities of Graphistry and Blazegraph into an integrated suite can provide the capability to more quickly identify anomalies over wider spans of time while maintaining contextual awareness
Initial Graph Filtered by PageRank using DASL
Interactive Visual Query Session
Visual Analysis - Identification of Exfiltrated Traffic
Stay in Touch

Jim Carbonaro,
Sr Software Engineer
jcarbonaro@blazegraph.com
http://blazegraph.com
@blazegraph