Socrates: A System for Scalable Graph Analytics
C. Savkli, R. Carr, M. Chapman, B. Chee, D. Minch

September 10, 2014

Cetin Savkli
Cetin.Savkli@jhuapl.edu
240 228 0115
Challenges of Big Data & Analytics

Scalability Challenges

- **Storage**: Ability to handle ever growing data sizes
- **Processing**: Need a parallel processing capability that does not move data

Complexity Challenges

- **Data Complexity**: Often overlooked; handling changing data structures is one of the key bottlenecks
- **Software Complexity**: Ability to integrate new software and services is another challenge that determines how well the system can evolve
- **Analytics**: Making Sense and learning from complex data using mathematically robust methods.
The Role of Graphs in Big Data & Analytics

- **Graph as a Data Structure:**
  - A flexible structure with power in expressing relationships
  - Facilitates integration of new data without having to deal with complex schema
  - Facilitates enriching data with analytics during and after data ingest - *Question Focused Data (QFD)*

- **Graph as an API:**
  - De-couple Apps from underlying structure of data
  - Provides a standard interface that unifies access to non-relational as well as relational data bases
  - Allows higher level analysis
Challenges and Motivation for Graph Processing

Challenges:

- Indexing
- Locality Control
- Attribute Based Queries
- Structure Based Queries

Need a solution that provides:

- Indexing & query capability of RDBMS
- Parallel processing & scalability of NOSQL
- Facilitates rapid prototyping of scalable analytics in diverse problem domains

Additional Desirable Features:

- Platform independent and simple to deploy
- Simple paradigm for development of parallelized processing
Socrates (The Philosopher)

Socrates: 470 BC – 399 BC, Athens

- Development of scientific method: Series of questions & answers to arrive at the truth.
- Executed by poisoning for asking too many questions and failing to acknowledge the gods.

"I know that I know nothing“, Socrates.
Socrates (Software) Overview

- Socrates general features
  - Stores graphs on distributed MySQL with the TokuDB engine
  - Indexing & query capability of RDBMS
  - Parallel processing & distributed storage
  - Locality control for partitioning data
  - Platform independent (runs in JVM)
  - No centralized management (all machines are equal)

- Socrates is under development, features not essential to most prototypes are not yet implemented
  - Transaction guarantees, backups, failover
  - If prototype needs to move into production, can reuse most/all code by switching to other Blueprints API implementation
    - Accumulo (APL developed)
    - Titan (Hbase, Cassandra, BerkleyDB)
Socrates Overview

Data Sources (Inputs): Oracle DB, CSV File, PCAP data

Entity & Link Extraction

Ingest

Oracle DB

CSV File

PCAP data

Distributed Graph Database

Entity 1 | Link          | Entity 2
---------|---------------|---------
Person A | Attends       | Flight School B
CreditCard C | ReferencedIn  | TravelRecord X
Person B  | ReferencedIn  | TravelRecord X

Display

User Interfaces & Alerting

Ingest

Distributed Graph Database

Enrichment

Analytic Outputs: Activity Patterns, Anomaly Detection, …

Scientist, Engineer Analytics Development

Analyst OWF Web Based Interface
Graph Structure

Socrates uses the following conventions:

- **Each graph vertex exists only on 1 machine and each graph edge can exist at most on 2 machines.**
- **Edges know the IDs of vertices they connect as well as the machine those nodes reside on.**
- **There is no central management of location information.**

![Graph Diagram]

### Table 1: Vertices and Edges (Machine A)

<table>
<thead>
<tr>
<th>ID</th>
<th>F</th>
<th>FID</th>
<th>EID</th>
<th>TID</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>A</td>
<td>V2</td>
<td>E_2_1</td>
<td>V1</td>
<td>A</td>
</tr>
<tr>
<td>V2</td>
<td>A</td>
<td>V3</td>
<td>E_3_2</td>
<td>V2</td>
<td>A</td>
</tr>
<tr>
<td>V3</td>
<td>A</td>
<td>V3</td>
<td>E_3_4</td>
<td>V4</td>
<td>A</td>
</tr>
<tr>
<td>V4</td>
<td>A</td>
<td>V4</td>
<td>E_4_1</td>
<td>V1</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>V1</td>
<td>E_1_7</td>
<td>V7</td>
<td>B</td>
</tr>
</tbody>
</table>

### Table 2: Vertices and Edges (Machine B)

<table>
<thead>
<tr>
<th>ID</th>
<th>F</th>
<th>FID</th>
<th>EID</th>
<th>TID</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>V5</td>
<td>B</td>
<td>V7</td>
<td>E_7_5</td>
<td>V5</td>
<td>B</td>
</tr>
<tr>
<td>V6</td>
<td>B</td>
<td>V5</td>
<td>E_5_6</td>
<td>V6</td>
<td>B</td>
</tr>
<tr>
<td>V7</td>
<td>B</td>
<td>V6</td>
<td>E_7_6</td>
<td>V7</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>V1</td>
<td>E_1_7</td>
<td>V7</td>
<td>B</td>
</tr>
</tbody>
</table>
Socrates stores vertex and edge attributes following the following principles:

- Every attribute type is stored in its own two column table (primary key & value)
- All attributes of a vertex resides on the machine where the vertex is. Attributes of an edge exist only on the machine where an edge originates.
- Attributes are typed and the type information is automatically determined.
- Attributes are indexed automatically by default with the exception of text and blob attributes.
Interacting with Socrates

Data Access:

- Interaction with the cluster using graph API and graph query methods
- Communication provided by JDBC
- Each machine runs a server that processes the query in parallel

Processing:

- Submitting jobs that are executed in parallel on local graphs
- Communication provided by JMS
- Jobs are executed in parallel on each machine and results are merged at the client
Interacting with Socrates, Cont’d

- Lowest level of parallelism: Fireworks
- Runs a given function on every vertex of the graph and its immediate neighborhood
  - Results written back into the graph
- Socrates takes care of multithreading, optimizing database queries, inserts, etc. in the background
  - Uses JGraph parallelism to run on every cluster node in parallel
  - Easily parallelize algorithms that can be performed with only local operations (PageRank, etc.)
Socrates Cluster

- APL test cluster: 16 nodes, Intel Xeon E5-2609 2.5GHz 4-core processors, 64 GB RAM, two 4.0TB HDDs in RAID 0
  - MySQL 5.5 with TokuDB storage engine as Socrates data store
    - InnoDB ingest hit severe bottleneck at about ~100m graph elements, TokuDB has scaled to over 1b elements

Ingest speed (1.1 x 10^8 elements)  
Fireworks connected component processing speed (1.1 x 10^8 elements)

- Primarily used to test scaling of algorithms
General Algorithms Support Rapid Development

- Connected components
- Subgraph isomorphism
- Frequent subgraph mining
- Time series aggregation
- Activity pattern analysis
- Mutual information
- Clustering
- Anomaly detection
- Colocation analysis
Examples of Analytics Developed in Socrates

• Detection of communities and trade patterns in international cargo shipments.

• Analysis of cyber networks to detect anomalous activities, including identification of collaborative behavior.

• Discovered previously unknown associates of person of interest hidden in travel activity data

• Detected anomalous flight behaviors such as formation flying, loitering.

• Analyze genetic data to determine gene measurements correlated with various diseases