ORACLE

Graph Analysis – Do We Have to Reinvent the Wheel?

Adam Welc
Raghavan Raman
Zhe Wu
Sungpack Hong
Hassan Chafi
Jay Banerjee

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle

Overview

- Graph analysis is becoming increasingly important
- Graph databases have been developed to facilitate graph analysis, but
  - They are not the only tool available
  - They are not always the fastest tool available
- We pitch the Neo4j graph database against SQL and a Domain Specific Language (DSL) targeting graph analysis

A bit of myth busting

- From "Graph Databases, NOSQL and Neo4j" by Peter Neubauer:

With shortest-path-calculations, Neo4j is even on small graphs of a couple of 1000 of nodes 1000 times faster than MySQL, the difference increasing as the size of the graph increases.

- We will show that SQL can be faster than Neo4j on shortest path calculations

Graph analysis solutions

- Neo4j - open source NOSQL graph database
  - Stores graph data directly as nodes and edges
  - Supports persistent data storage and full ACID txn-s

  SQL
  - Querying data stored in a relational DB (storage + ACID)
  - Graph data stored in table(s)
- Green-Marl – DSL for graph analysis
  - Optimizing source-to-source (Green-Marl to C++) compiler + shared memory runtime supporting parallel in-memory execution of graph algorithms (no persistence or txn-s)

Case study

- Focused on shortest path algorithm
- All graph analysis solutions support bi-directional Dijkstra's shortest path
- Used two real-life graphs:
  - Live Journal (~4.8M nodes, ~69M edges) – represents members of an on-line community
  - Twitter (~41.7M nodes, 1.47B edges) – represents user profiles and relations between users
- Computed shortest path for 50 randomly chosen src/dst pairs for each graph

Experimental setup

- Machine used for experiments: 2.2GHz Intel Xeon E5-2660 “Sandy Bridge” with 264GB of RAM
- No special configuration options for Green-Marl (binary generated using gcc 4.4.7)
- SQL code executed in Oracle Database 12c allocated 16GB of RAM
- Used 1.8.2 community edition of Neo4j and Java JDK 1.6.0_43 – spent large amount of time tuning the environment to get the best results

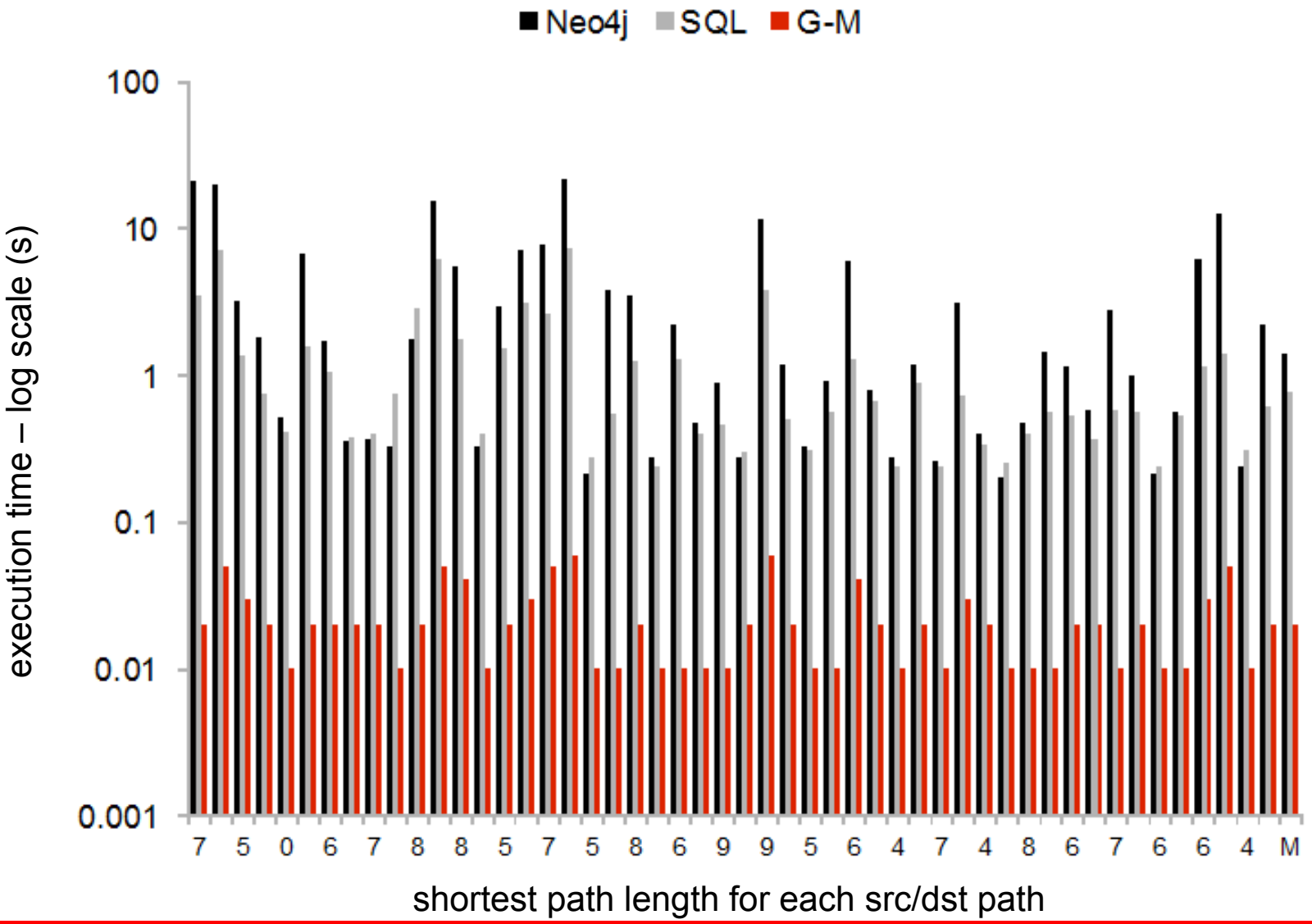
Neo4j configuration experiments

- Different cache types: none, weak, strong, soft
  - Settled on soft for Live Journal, none for Twitter
- Embedded Graph DB and Embedded Read-only Graph DB
  - Settled on Embedded Read-only Graph DB
- Different GC algorithms and GC parameters
  - CMS and parallel GC (also only for full collections)
  - Different heap sizes (-Xmx option) between 8GB and 128GB
  - Multiple values of -Xmn, -XX:SurvivorRatio options
  - Settled on parallel GC throughout (GC overhead < ~5%)
    - Twitter: -Xmx32g -Xmn16g -XX:SurvivorRatio=8
    - Livej: -Xmx16g -Xmn12g -XX:SurvivorRatio=6
- GC-resistant caches (enterprise edition 1.8.2)
  - Couldn't get it to work

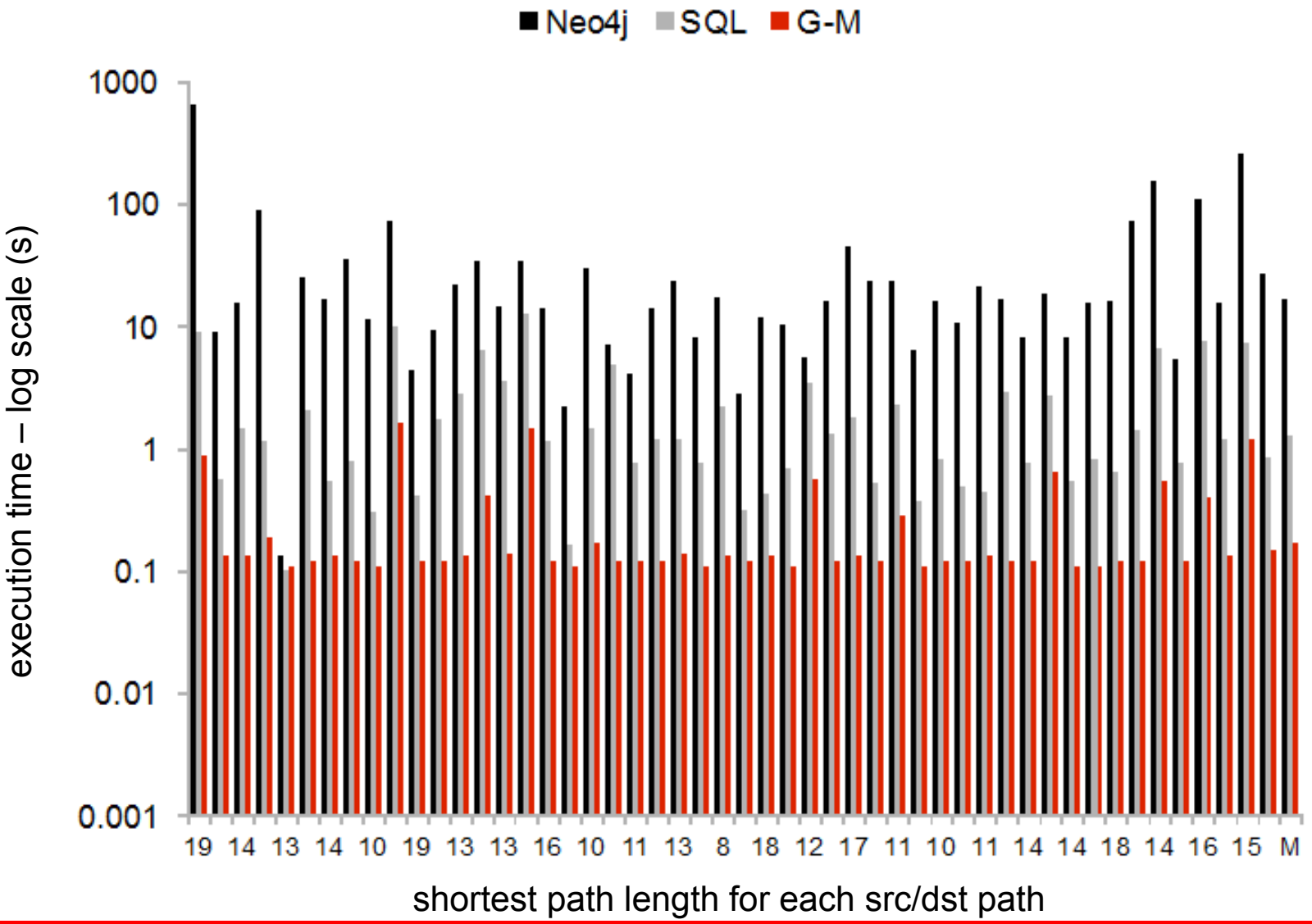
Additional Neo4j configuration options

- Basic parameters
  - node_auto_indexing = true
  - relationship_auto_indexing = true
  - use_memory_mapped_buffers = true
  - log_mapped_memory_stats = true
  - dump_configuration = true
- File Buffer Cache
  - nodestore = number of nodes * 9 bytes
  - relationshipstore = number of relationships * 33 bytes
  - propertystore (for primitive types) = number of properties (for nodes and edges) * 41 bytes
  - propertystore (for strings) = 100 M
    - We do not have any string properties on nodes or edges
  - propertystore (for arrays) = 100 M
    - We do not have any array properties on nodes or edges
  - propertystore (for index) = 2 G
  - propertystore (for index keys) = 2 G
- Java configuration
  - 64-bit mode
  - Server mode

Live Journal graph results



Twitter graph results



Conclusions

- Green-Marl is faster than both SQL and Neo4j but it performs analysis in-memory and does not support durability or txn-s
- SQL can be faster than dedicated graph analysis solutions such as Neo4j
  - Though we DO NOT claim that it will always be faster
- We postulate that a hybrid (in-memory + SQL) solution can
  - Have competitive or better performance than graph databases
  - Leverage the enterprise features provided by a modern RDBMS

http://tinyurl.com/olabs-grades2013-1