CRAY

Quantifying CGE Performance: A Unified Scalable Pattern Mining and Search System

Rob Vesse, Software Engineer, Analytics R&D, Cray Inc

Overview

- **Purpose**
  - Demonstrate the ability to architect a scalable multi platform analytics application
  - Quantify the performance of Cray Graph Engine across multiple platforms and contrasts with competing projects e.g. Apache Spark
- **Results**
  - Strong scaling on Cray XC^{TM} systems
  - Equivalent performance between Cray XC^{TM} and Urika-GX^{TM} systems
  - Substantially better performance than Apache Spark on the same systems
- **Summary**
- **Q&A**

Terminology

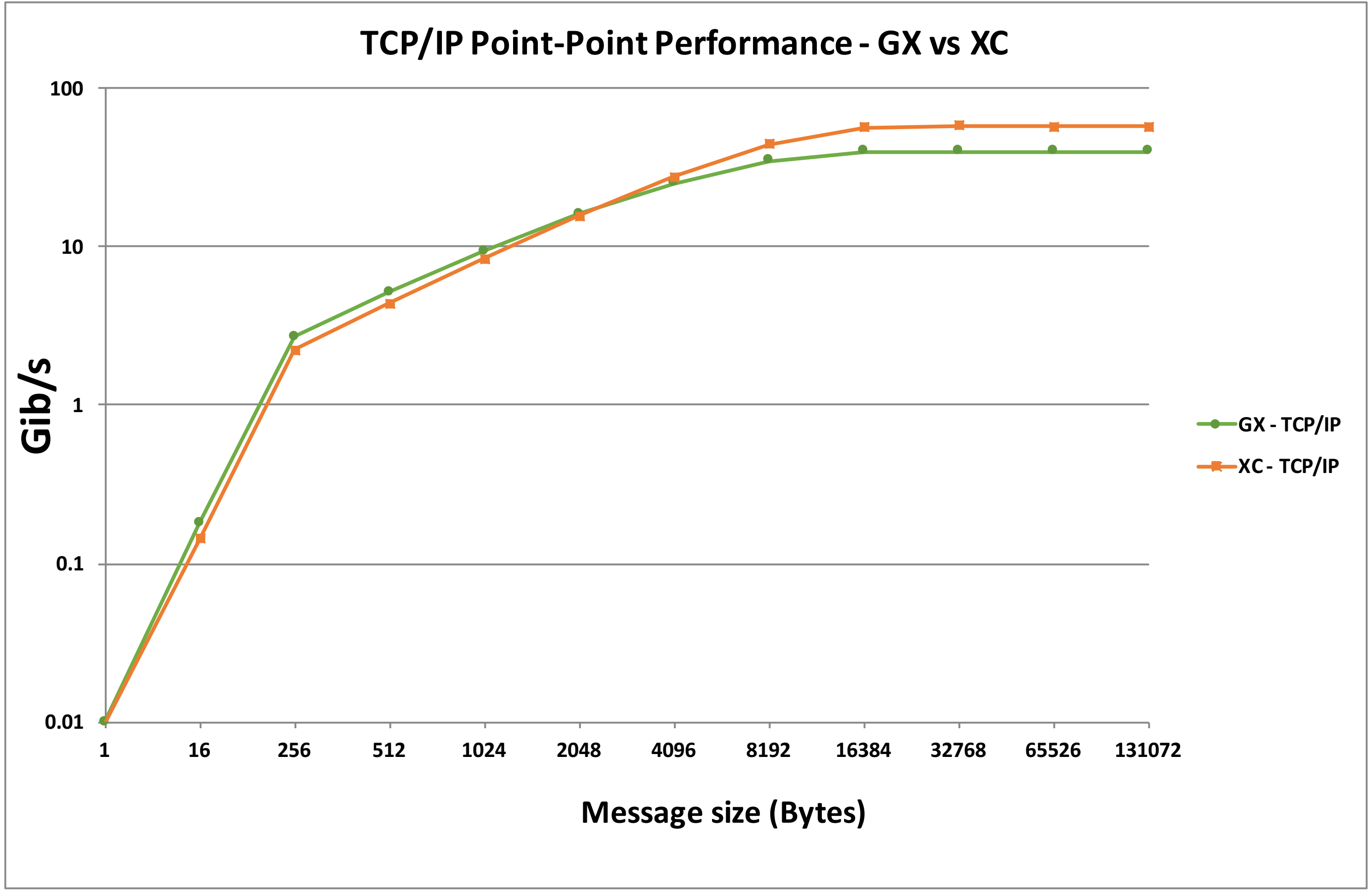
- **Cray Graph Engine (CGE)**
  - Scalable parallel graph analytics framework
  - W3C Standards Based
    - Uses RDF Data representation
    - Uses SPARQL as query language
- **Apache Spark**
  - In memory parallel analytics framework
  - Originated from UC Berkeley AMPLab
  - GraphX is their graph analytics component

Architecture

Architecture - Hardware Differences

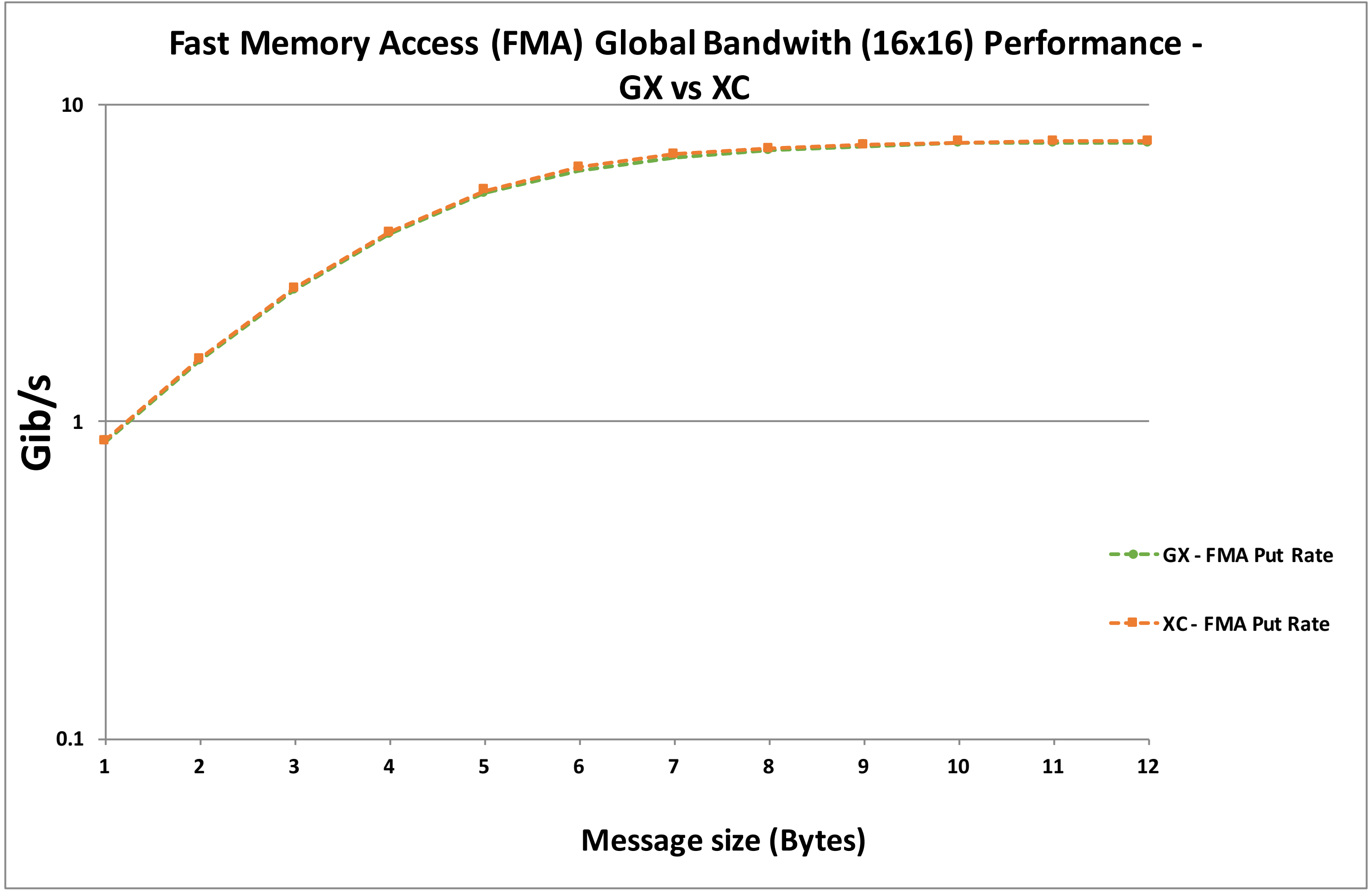
- Understand the physical hardware differences upfront
- Analyze how that will impact your application
- Cray XC™ versus Urika-GX™
  - Proprietary blades versus commodity blades
  - Typically larger local memory on GX nodes
  - Direct connection to Aries versus PCI-e connection to Aries expansion card
  - Thousands of nodes versus 48 nodes
  - No local storage versus multiple on-board disks (HDD and SSD)

Architecture - Network Throughput



- Iperf3 Point-Point benchmark
- TCP/IP substantial gap of 20 GiB/s
- Spark relies on TCP/IP

Architecture - Network Throughput



- Custom global bandwidth benchmark (16x16)
- Fast Memory Access (FMA) is equivalent
- CGE relies on FMA

Architecture - Software Considerations

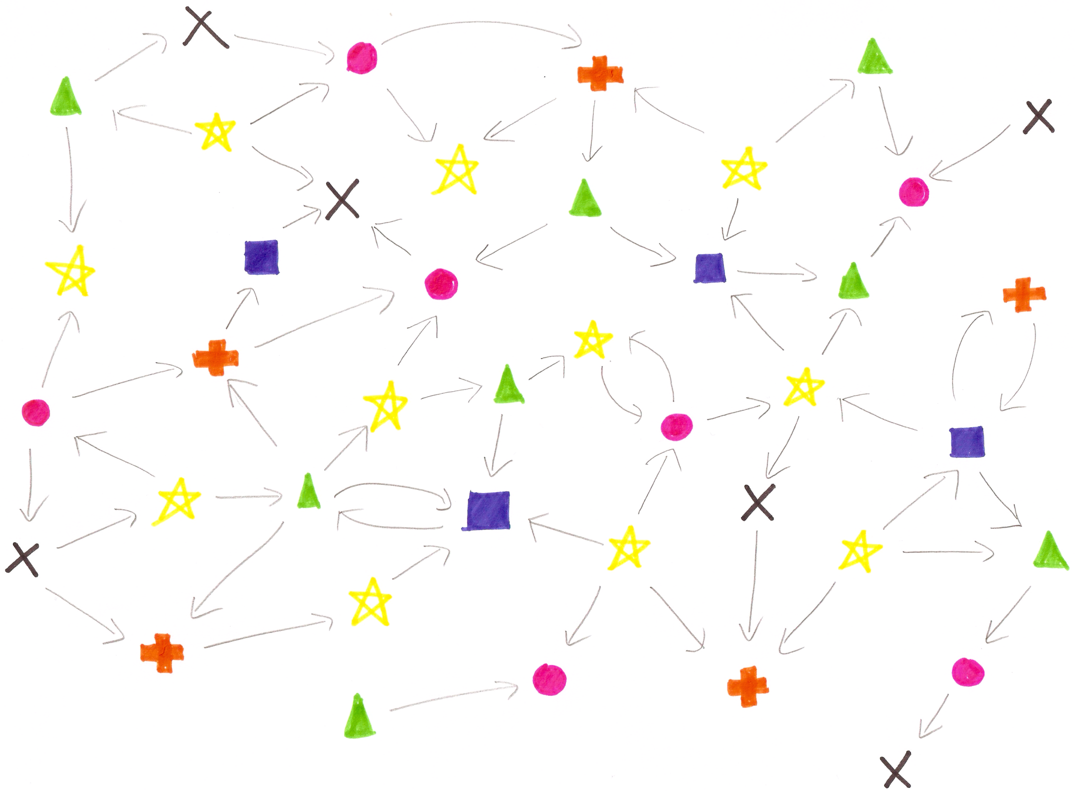
- **Target an appropriate portable runtime**
  - We use Coarray C++ which is backed by PGAS/DMAPP
- **Treat network as the bottleneck**
  - Maximize locality to minimize network usage and global synchronization
- **Abstract launch**
  - Provide a wrapper that allows your application to be agnostic to the underlying workload manager
- **Abstract other subsystems as appropriate**
  - We support IO from both POSIX compliant and HDFS file systems

Experiments

Experimental Setup

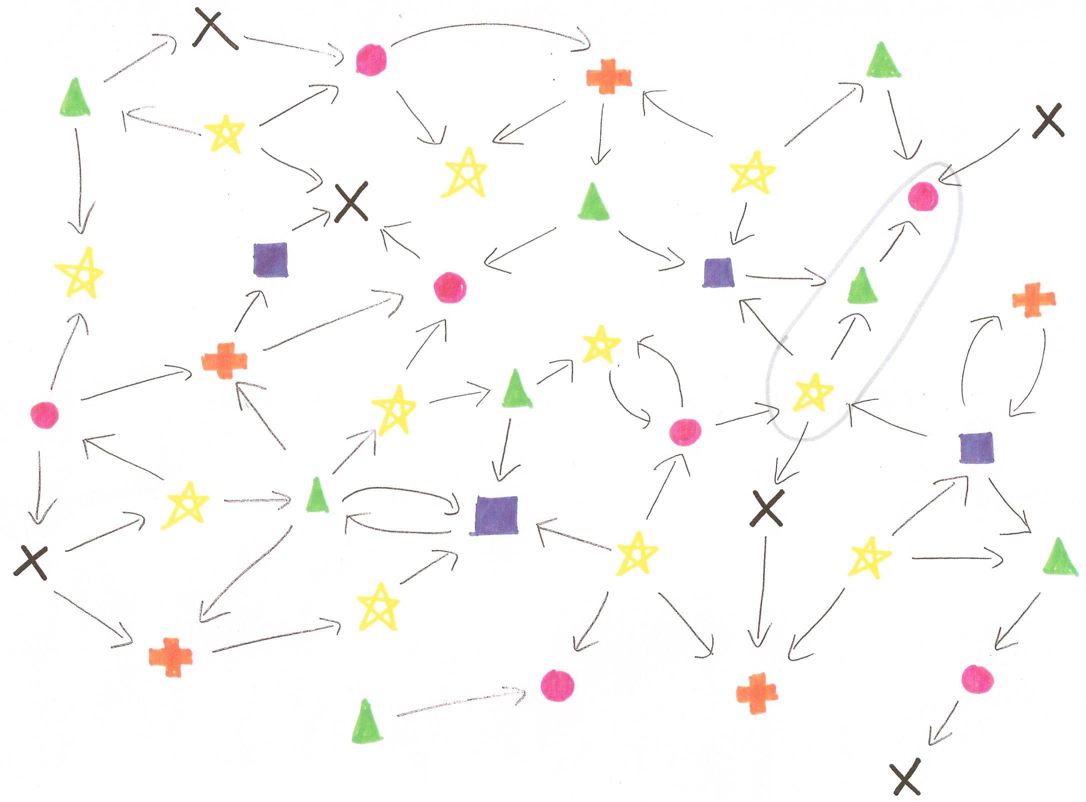
- **Hardware**
  - Six cabinet XC with mixed node types
    - 36 core Broadwell nodes with 128GB DDR4-2400 RAM
  - 48 Node Urika-GX
    - 32 core Broadwell nodes with 256GB DDR4-2400 RAM
- **Software**
  - CGE 3.0UP00 running 16 processes per node
  - Apache Spark 2.1.0 w/ Cray patches
- **Datasets**
  - Lehigh University Benchmark (LUBM), a synthetic graph representing academic institutions
    - 25k scale is ~3 billion triples, 200k scale is ~24 billion triples
  - Stanford Network Analysis Project (SNAP)
    - US Patent Citations and two online social networks

Graph analysis workloads

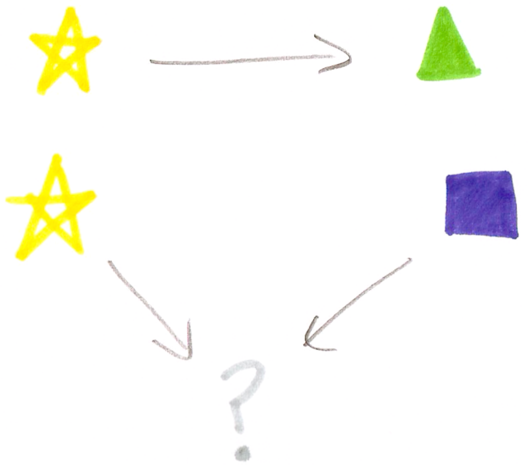


- Two main workloads
  - Pattern matching
  - Whole graph analysis
- Typical systems only good at one
- CGE excels at both

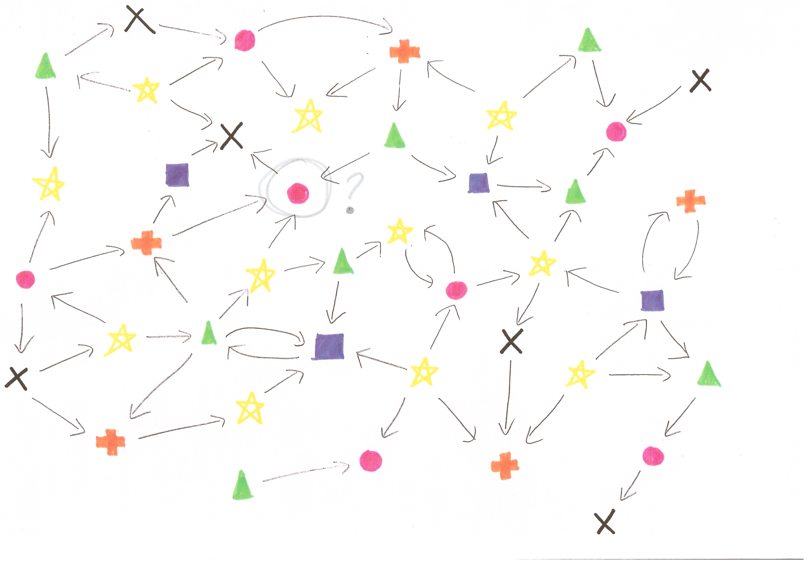
Pattern matching workload



- Given a pattern of interest find all instances thereof

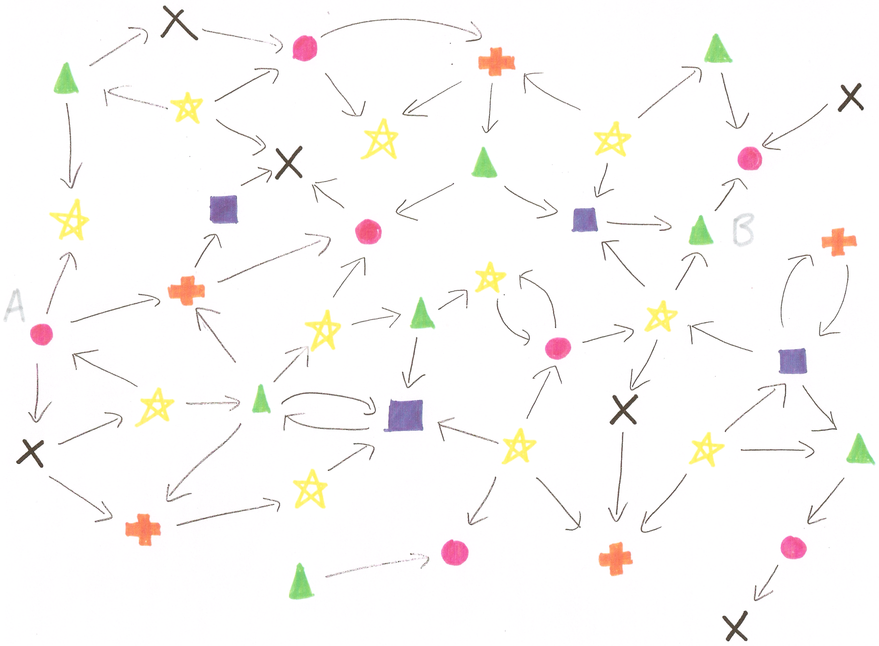


Whole graph analysis workload



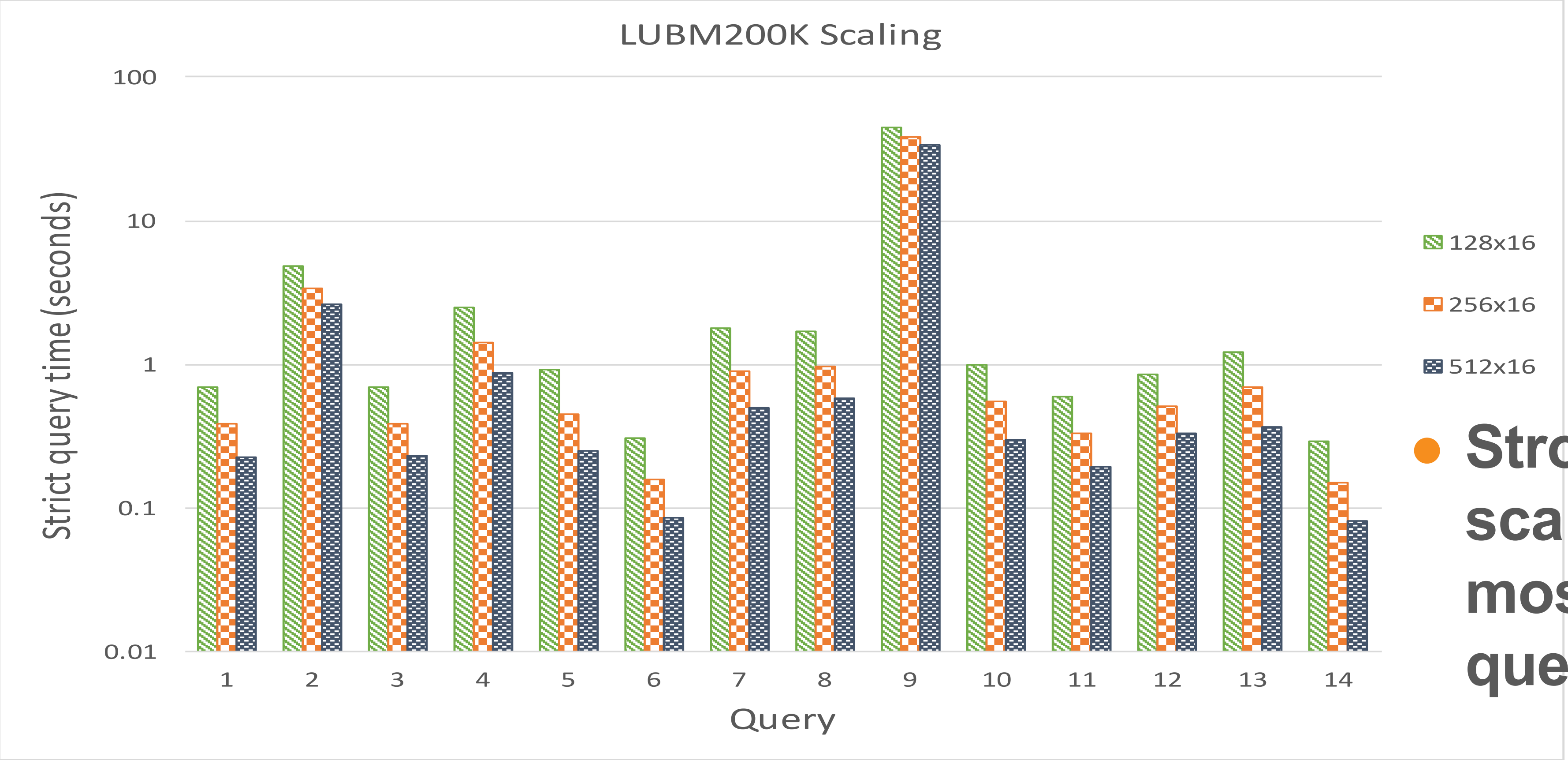
- What is the ranking of the targeted vertex?

- What's the shortest route from A to B?



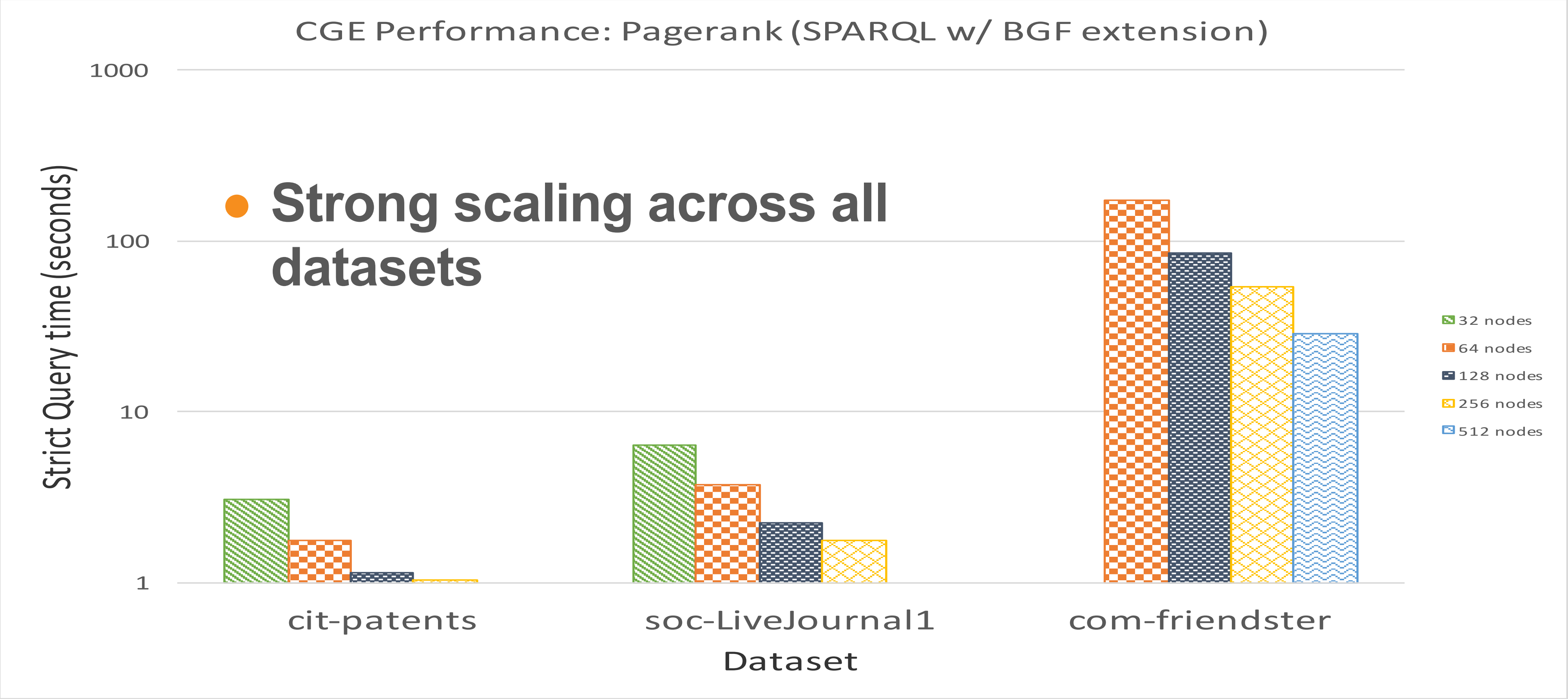
Results

Pattern matching scaling



- Strong scaling on most queries

Whole graph analysis scaling

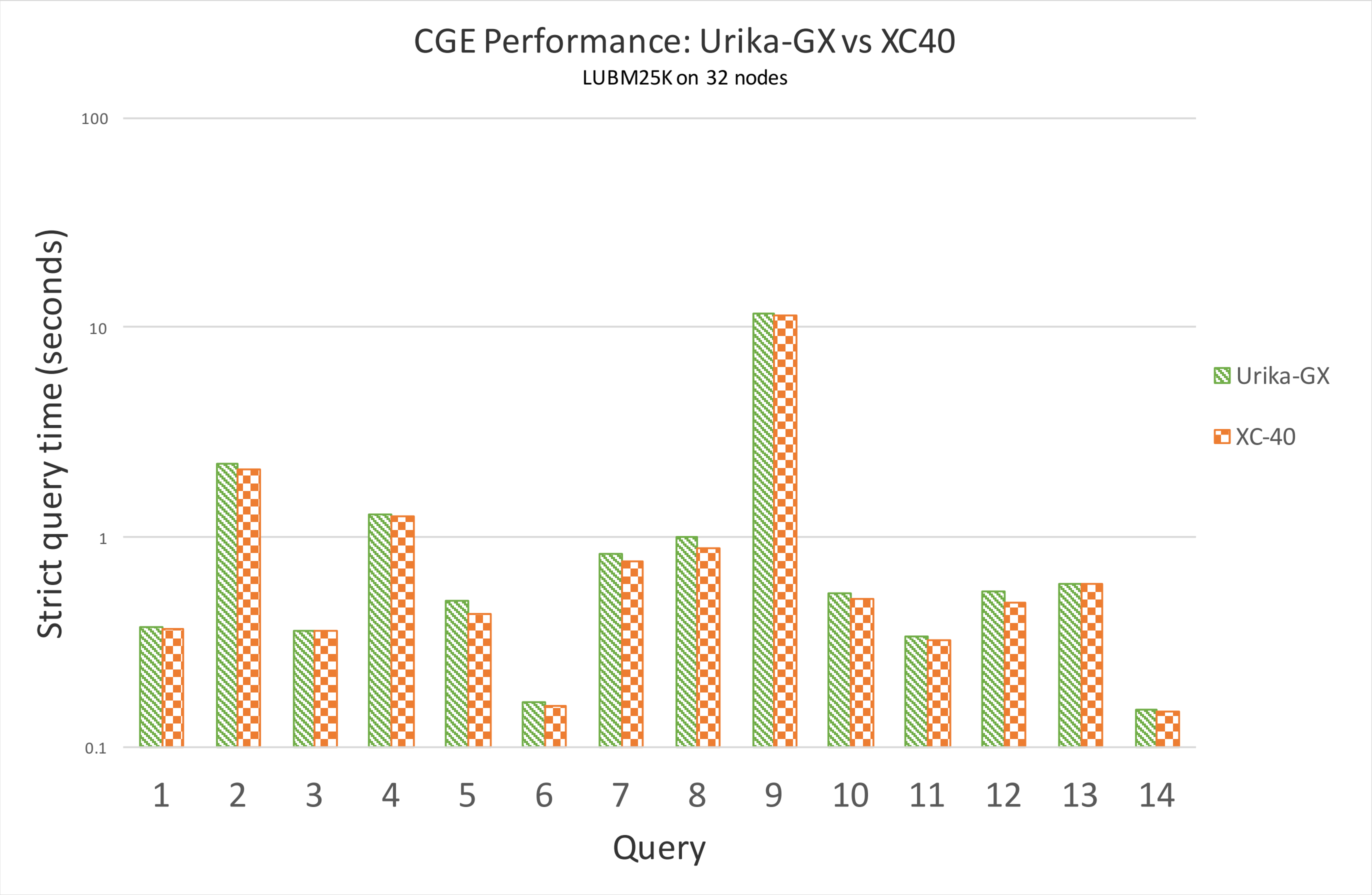


- Strong scaling across all datasets

Challenges for multi-platform performance

- **Core Affinity & NUMA Binding**
  - Different workload managers provide very different defaults for core affinity
  - Non-HPC workload managers e.g. Apache Mesos^{TM} don't provide this at all
- **Network performance**
  - Need to minimize network usage so that the additional latency doesn't matter too much

Platform Comparison - GX vs XC

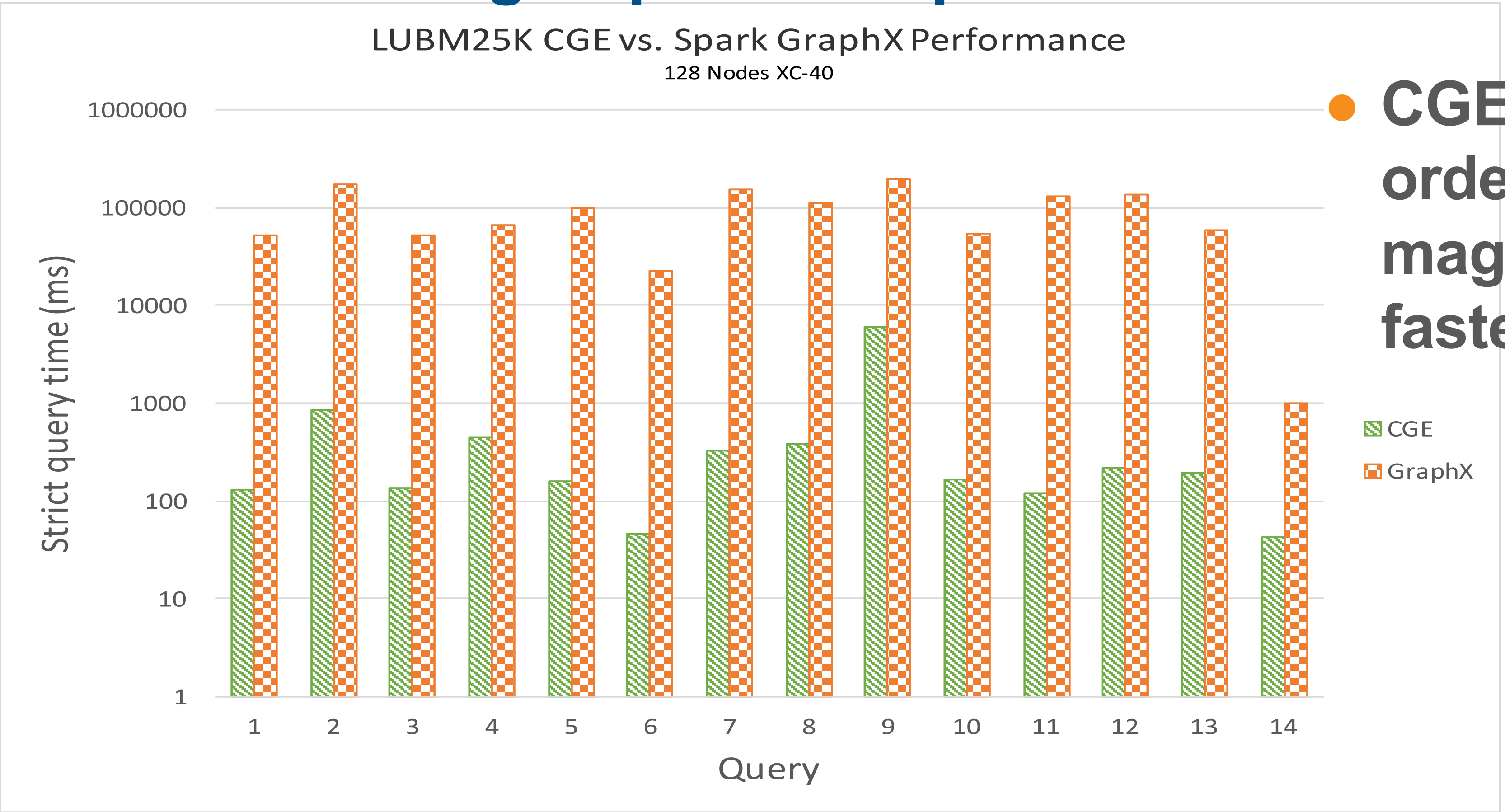


- Performance gap is minimal
- Lot of work to reach this point

Spark Comparison

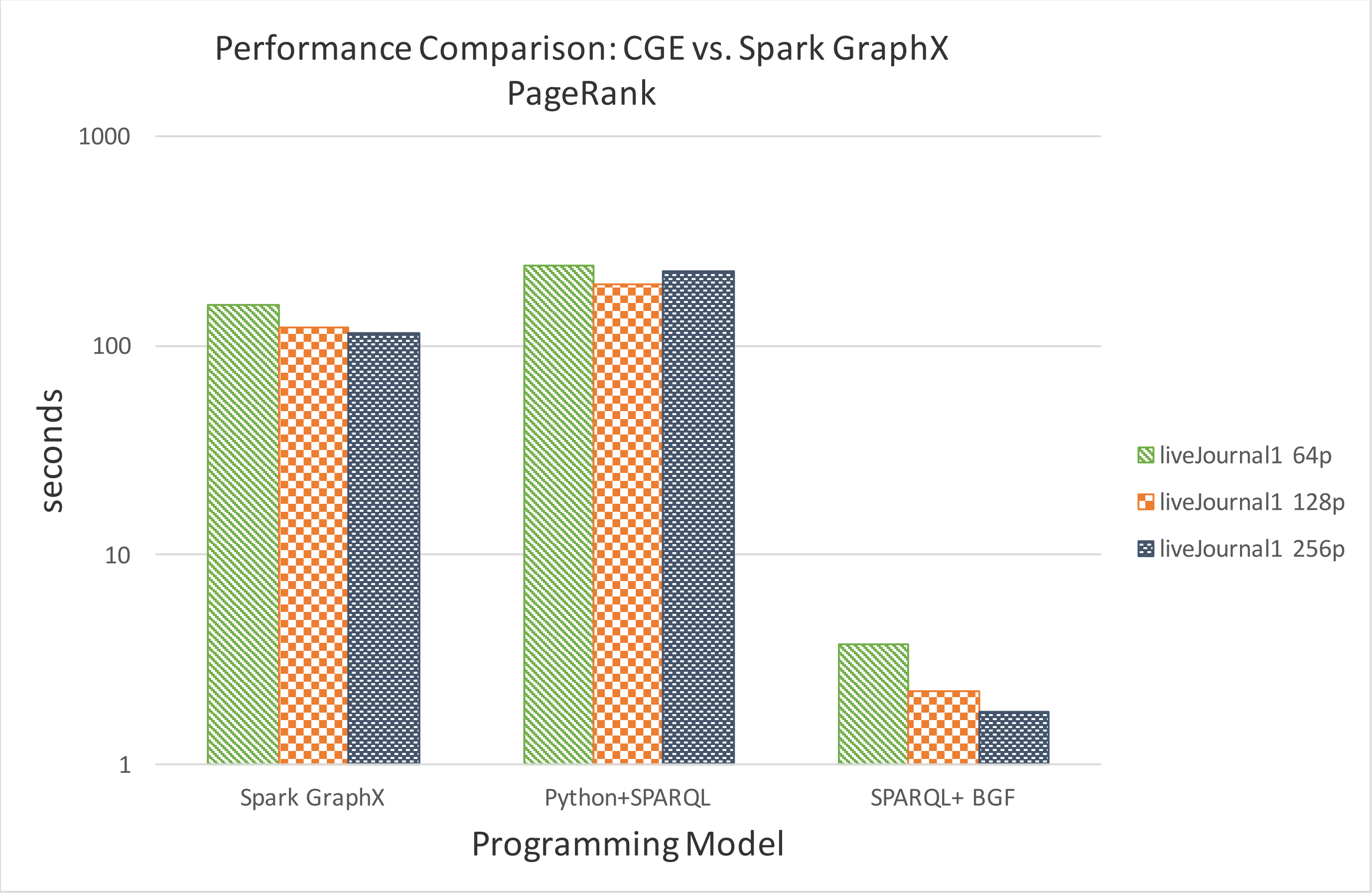
- We created Spark versions of all the experiments described in the paper
- For pattern matching our codes were taken from past work carried out by Oak Ridge National Lab (ORNL)
- For whole graph analysis we compare 3 approaches:
  1. Spark code using GraphX package
  2. Iterative approach using standard SPARQL
  3. Native Coarray C++ using Cray SPARQL extensions

Pattern matching - Spark Comparison



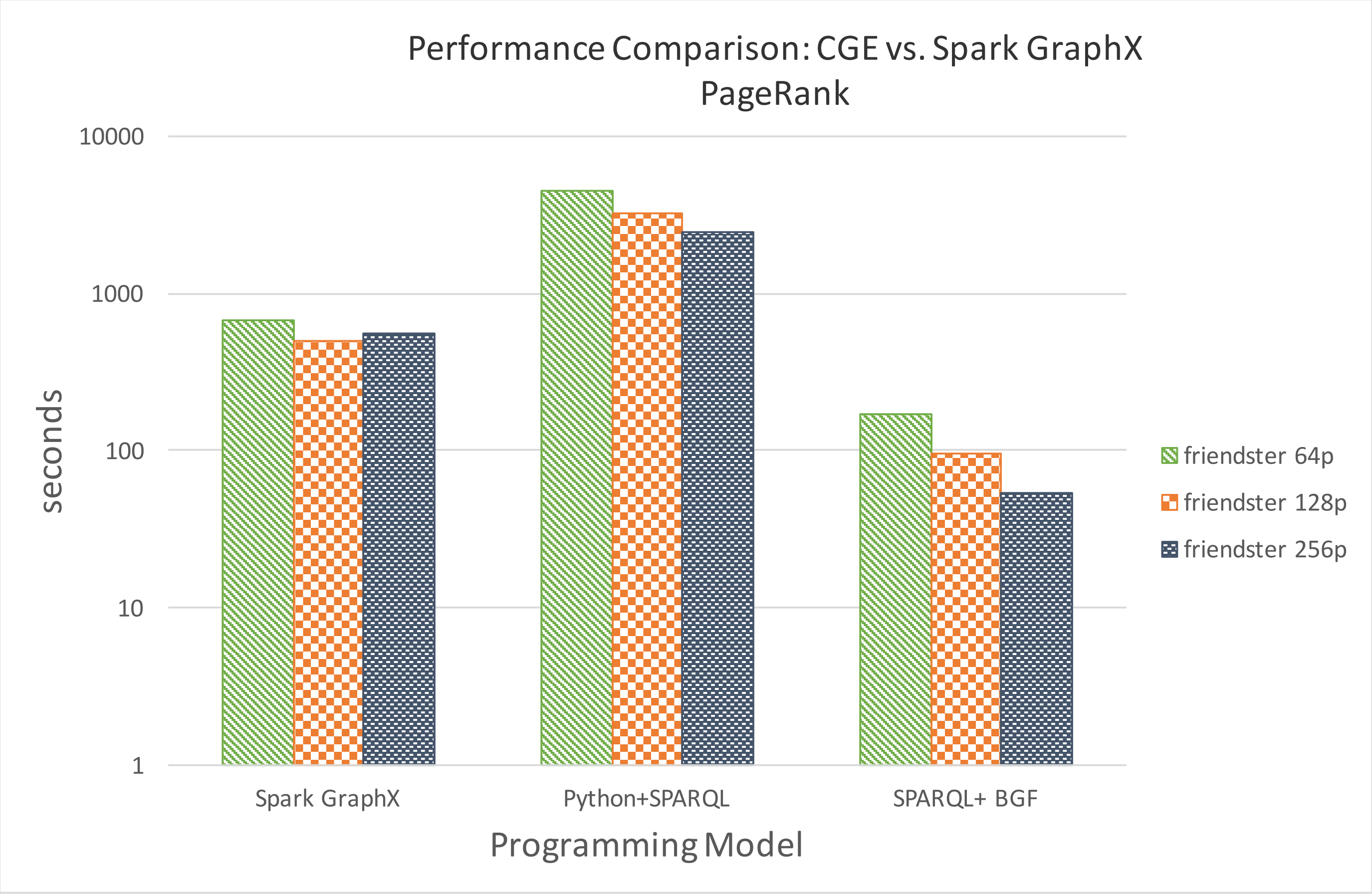
- CGE 1-2 orders of magnitude faster

Whole graph analysis - Spark Comparison



- CGE order of magnitude faster
- Iterative SPARQL approach equivalent to Spark

Whole graph analysis - Spark Comparison



- CGE order of magnitude better than Spark
- Dataset characteristics affect performance

Summary

- **Demonstrated the ability to architect a scalable multi platform analytics application**
- **Quantified the performance of Cray Graph Engine across multiple platforms on a variety of workloads**
  - Strong scaling on Cray XC^{TM} systems
  - Substantially better performance than Apache Spark for both pattern matching and whole graph analysis workloads

Legal Disclaimer

Information in this document is provided in connection with Cray Inc. products. No license, express or implied, to any intellectual property rights is granted by this document.

Cray Inc. may make changes to specifications and product descriptions at any time, without notice.

All products, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Cray hardware and software products may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Cray uses codenames internally to identify products that are in development and not yet publically announced for release. Customers and other third parties are not authorized by Cray Inc. to use codenames in advertising, promotion or marketing and any use of Cray Inc. internal codenames is at the sole risk of the user.

Performance tests and ratings are measured using specific systems and/or components and reflect the approximate performance of Cray Inc. products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance.

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, and URIKA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, CRAYPAT, CRAYPORT, ECOPHLEX, LIBSCI, NODEKARE, REVEAL, THREADSTORM. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

Q&A

Rob Vesse	rvesse@cray.com
Kristyn Maschoff	kristyn@cray.com
Jim Maltby	jmaltby@cray.com
Rangan Sukumar	ssukumar@cray.com

CUG.2017.CAFFEINATED COMPUTING

Redmond, Washington May 7-11, 2017