A Portable, High-Level Graph Analytics Framework Targeting Distributed, Heterogeneous Systems

Robert Searles*, Stephen Herbein*, and Sunita Chandrasekaran

November 14, 2016

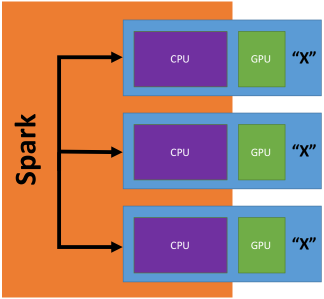
Motivation

- HPC and Big Data communities are converging
- Heterogeneous and distributed systems are becoming increasingly more common
- Distributing data and leveraging specialized hardware (e.g. accelerators) is critical
- Graph analytics are important to both communities

Goal

- Develop a portable, high-level framework for programming current and future HPC systems that:
  - Distributes data automatically
  - Utilize heterogeneous hardware
- Accelerate two real-world graph analytics applications
- Demonstrate portability by running on a variety of hardware, including multi-core Intel CPUs, NVIDIA GPUs, and AMD GPUs

Our Framework: Spark + X



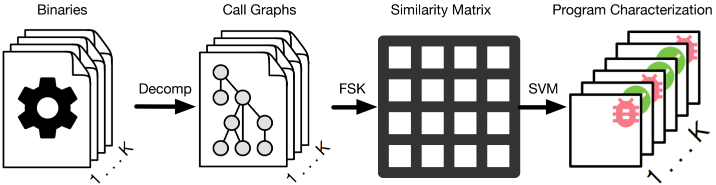
- Utilize the MapReduce framework, Spark, to handle data and task distribution
  - Automatic data/task distribution
  - Fault-tolerant
  - Minimal programmer overhead
- Leverage heterogeneous resources to compute the tasks local to each node
  - Accelerators and other emerging trends in HPC technology

Case Study Applications

- Fast Subtree Kernel (FSK)
  - Call graph similarity analysis
    - Program characterization
    - Malware analysis
- Triangle enumeration
  - Spam detection
  - Web link recommendation
  - Social network analysis

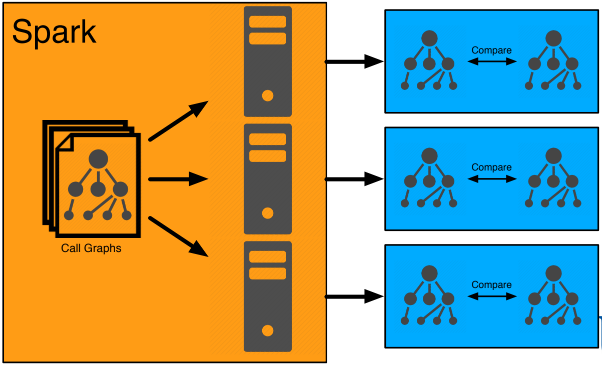
What is FSK?

- Compute-bound graph kernel
- Measures the similarity of graphs in a dataset
- A graph is represented by a list of feature vectors
  - Each feature vector represents a subtree



FSK in our framework

- Spark Component
  - Split up pairwise graph comparisons
- Local Component
  - For each pair of graphs
    - Compare all feature vectors



What is Triangle Enumeration?

- Data-bound graph operation
- Finds all cycles of size 3 (AKA triangles) within a graph

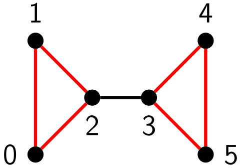
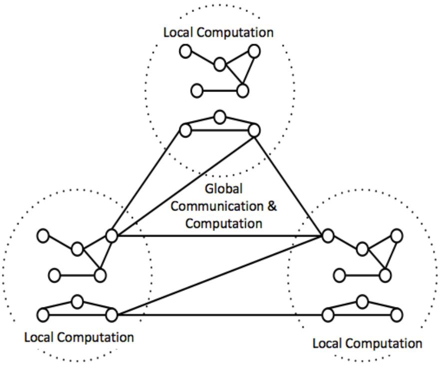


Figure: This graph contains 2 triangles (highlighted in red).

Triangle Enumeration in our framework

- Spark Component
  - Partition the graph
  - Distribute the vertices/edges across the cluster
- Local Component
  - Count triangles within each subgraph
  - Done using matrix-matrix multiplication (BLAS)
- Spark Component
  - Count triangles between subgraphs



Hardware/Software

Machine #	CPU	GPU
1	Intel Xeon E5520 (Dual socket, 4 cores each)	NVIDIA K20 (5GB GDDR5)
2	Intel Core i7-5930K (6 cores)	NVIDIA GTX 970 (4GB GDDR5)
3	Intel Core i7-950 (4 cores)	NVIDIA GTX 470 (1.2GB GDDR5)
4	Intel Core i7-4771 (4 cores)	AMD Fury X (4GB HBM)

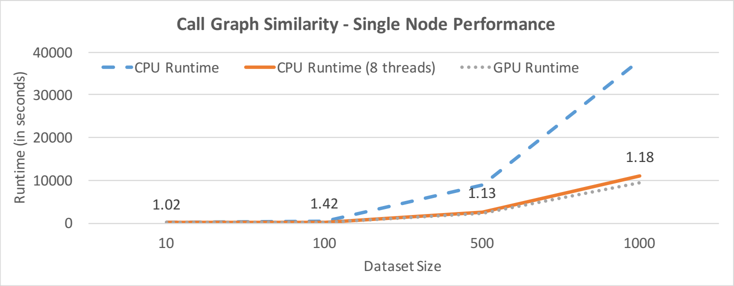
Fast Subtree Kernel

- Software
  - PySpark
  - PyOpenCL
- Hardware: AMD GPU
  - Fury X

Triangle Enumeration

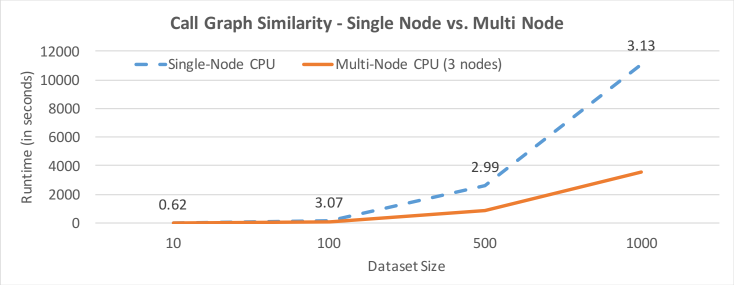
- Software
  - PySpark
  - ScikitCUDA
- Hardware: NVIDIA GPUs
  - GTX 470
  - GTX 970
  - Tesla K20c

FSK Results - Single-Node Parallelism



- Single node runtimes (Single thread, 8 thread, and GPU)

FSK Results - Multi-Node Scalability

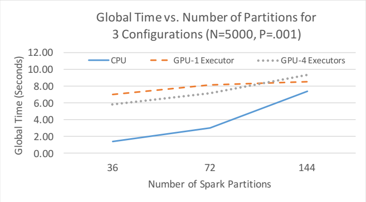


- Multiple node runtimes (CPU saturated on all nodes)

Triangle Enumeration - Optimizing Data Movement

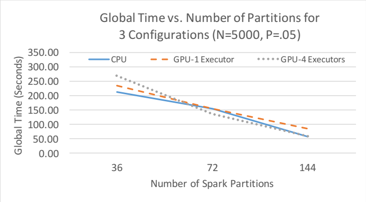
- Runtime of Spark component for Triangle Enumeration with a variable number of partitions for Erdos-Renyi random graphs with differing densities

Sparse graphs (P=.001)



- Fewer partitions allows for more triangles to be counted locally

Denser graphs (P=.05)

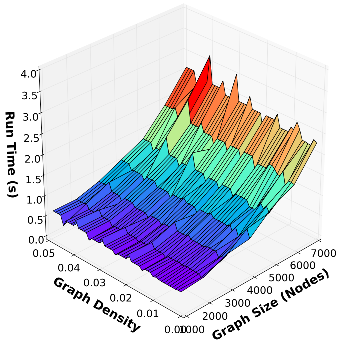


- More partitions means oversubscription of the GPU
- Overlaps communication with computation

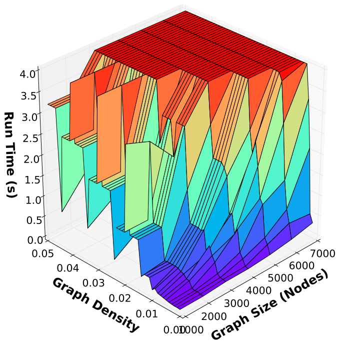
Triangle Enumeration - Optimizing Local Computation

- Performance of the local component of Triangle Enumeration on the CPU and GPU for graphs of varying size and density

GPU (ScikitCUDA)



CPU (Scipy)



- Running on the GPU is preferred unless the graph is sparse (density $< .01$), then running on the CPU is preferred

Conclusion

- FSK
  - Linear Scaling
  - GPU outperforms CPU
  - Free load balancing with Spark
- Triangle Enumeration
  - Optimize data movement by changing the number of Spark partitions
  - Improve local performance by choosing where to execute tasks
- Our high-level framework
  - Demonstrated portability using a variety of hardware

Future Work

- Additional case-study application
  - Spike neural network training
  - Detecting common subgraphs within neural networks
- Additional tests
  - Scalability test on a large-scale homogenous cluster
  - Add latest Nvidia GPUs (K40/80) to our heterogenous cluster

Reproducibility

- All data and code on GitHub
  - `https://github.com/rsearles35/WACCPD-2016`