High-Performance Graph Primitives on the GPU: Design and Implementation of Gunrock

Yangzihao Wang

University of California, Davis

yzhwang@ucdavis.edu

March 24, 2014

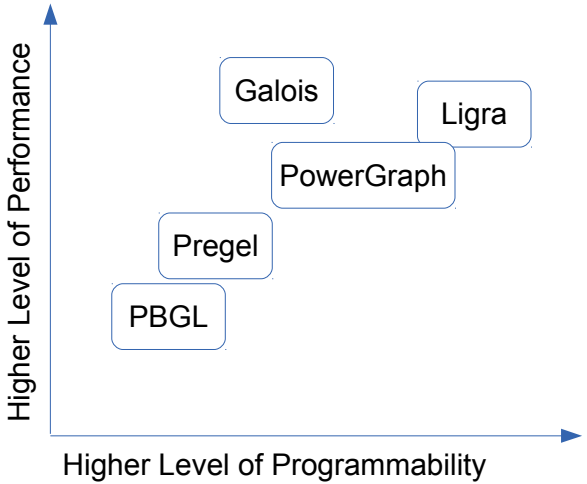
UCDAVIS COLLEGE OF ENGINEERING

Problem Overview

There is a significant demand for efficient parallel framework for processing and analytics on large-scale graphs.

Applications in fields such as: Social Media, Science and Simulation, Advertising, and Web.

Graph Processing on the CPU



Graph Processing on the GPU

Initial results show that GPU computing is promising for future processing of large-scale graphs. Most current research on GPU graph processing focuses on the challenge of irregular memory accesses and work distribution. However, two gaps in graph computation on the GPU are overlooked.

Graph Processing on the GPU

Initial results show that GPU computing is promising for future processing of large-scale graphs. Most current research on GPU graph processing focuses on the challenge of irregular memory accesses and work distribution. However, two gaps in graph computation on the GPU are overlooked.

- **Algorithmic and Implementation Gap:** few algorithms have efficient single-node GPU implementations when compared to the state of the art on CPUs, and multi-node GPU implementations are even rarer.

UCDAVIS COLLEGE OF ENGINEERING

Graph Processing on the GPU

Initial results show that GPU computing is promising for future processing of large-scale graphs. Most current research on GPU graph processing focuses on the challenge of irregular memory accesses and work distribution. However, two gaps in graph computation on the GPU are overlooked.

- **Algorithmic and Implementation Gap:** few algorithms have efficient single-node GPU implementations when compared to the state of the art on CPUs, and multi-node GPU implementations are even rarer.
- **Programmability Gap:** the implementations of these algorithms are complex and require expert programmers. The resulting code couples graph computation to parallel graph traversal, and is difficult to maintain and extend.

UCDAVIS COLLEGE OF ENGINEERING

Graph Processing on the GPU

The main goal of Gunrock's programming model is to enable end-users to express, develop, and refine iterative graph algorithms with a high-level, programmable, and high-performance abstraction.

Graph Processing on the GPU

The main goal of Gunrock's programming model is to enable end-users to express, develop, and refine iterative graph algorithms with a high-level, programmable, and high-performance abstraction.

- **Programmability:** To be expressive enough to represent a wide variety of graph computations.

UCDAVIS COLLEGE OF ENGINEERING

Graph Processing on the GPU

The main goal of Gunrock's programming model is to enable end-users to express, develop, and refine iterative graph algorithms with a high-level, programmable, and high-performance abstraction.

- **Programmability:** To be expressive enough to represent a wide variety of graph computations.
- **Performance:** To leverage the highest-performing GPU computing primitives.

UCDAVIS COLLEGE OF ENGINEERING

Common Characteristics of Graph Problems

Common Characteristics of Graph Problems

- **Large-Scale:** Facebook has over 1 billion nodes, 144 billion friendships and Twitter has 500 million nodes and over 15 billion follower edges.

UCDAVIS COLLEGE OF ENGINEERING

Common Characteristics of Graph Problems

- **Large-Scale:** Facebook has over 1 billion nodes, 144 billion friendships and Twitter has 500 million nodes and over 15 billion follower edges.
- **Heterogenous Node Degree Distribution:** Several real-world graphs have such scale-free structure whose degree distribution follows a power law, at least asymptotically, which brings challenge on load-balancing.

UCDAVIS COLLEGE OF ENGINEERING

Common Characteristics of Graph Problems

- **Large-Scale:** Facebook has over 1 billion nodes, 144 billion friendships and Twitter has 500 million nodes and over 15 billion follower edges.
- **Heterogenous Node Degree Distribution:** Several real-world graphs have such scale-free structure whose degree distribution follows a power law, at least asymptotically, which brings challenge on load-balancing.
- **Iterative Convergent Process:** Start with a working queue contains a subset of the graph, iteratively do the computation, will finally converge.

UCDAVIS COLLEGE OF ENGINEERING

Design Choices

Design Choices

- **Bulk Synchronous Parallel (BSP) Model:** Large amount of independent computations can be parallelized when traverse the graph. Programming simplicity and scalability advantages.

Design Choices

- **Bulk Synchronous Parallel (BSP) Model:** Large amount of independent computations can be parallelized when traverse the graph. Programming simplicity and scalability advantages.
- **Queue-based Iterative Method:** Workload efficient. Small cost of keeping a compact working queue with GPU primitives: scan + stream compact. Could easily expand to support priority scheduling.

UCDAVIS COLLEGE OF ENGINEERING

Design Choices

- **Bulk Synchronous Parallel (BSP) Model:** Large amount of independent computations can be parallelized when traverse the graph. Programming simplicity and scalability advantages.
- **Queue-based Iterative Method:** Workload efficient. Small cost of keeping a compact working queue with GPU primitives: scan + stream compact. Could easily expand to support priority scheduling.
- **Hybrid Push-vs-Pull style of graph traversal:** Start graph traversing step from either the current frontier (push) or the unvisited set (pull) to achieve the highest performance.

UCDAVIS COLLEGE OF ENGINEERING

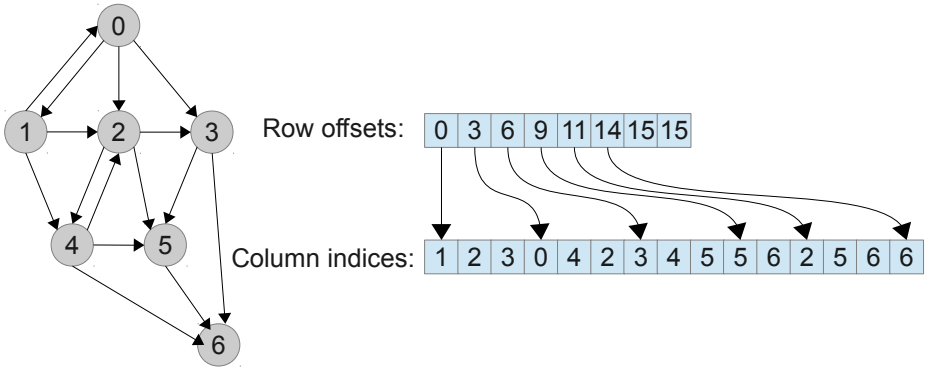
Design Choices

- **Bulk Synchronous Parallel (BSP) Model:** Large amount of independent computations can be parallelized when traverse the graph. Programming simplicity and scalability advantages.
- **Queue-based Iterative Method:** Workload efficient. Small cost of keeping a compact working queue with GPU primitives: scan + stream compact. Could easily expand to support priority scheduling.
- **Hybrid Push-vs-Pull style of graph traversal:** Start graph traversing step from either the current frontier (push) or the unvisited set (pull) to achieve the highest performance.
- **Idempotent Operation:** Whether to permit a vertex to appear multiple times in frontier(s) from one or more iterations.

UCDAVIS COLLEGE OF ENGINEERING

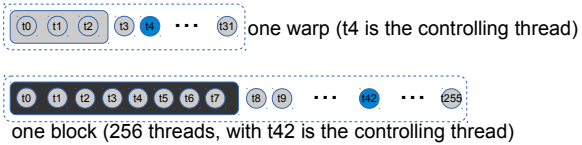
Graph Data Representation

Gunrock uses compressed sparse row (CSR) format for vertex-centric operations and edge list for edge-centric operations.



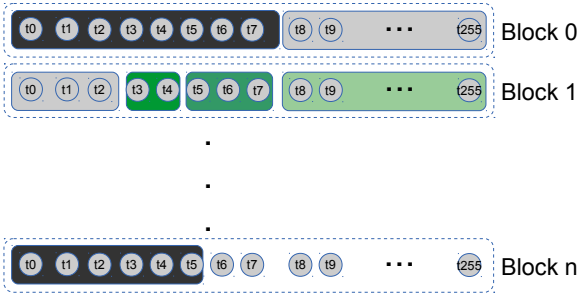
Workload Mapping Strategies (What makes Gunrock traversal fast?)

- **The Problem:** Write various-length neighbor lists of vertices into an output queue.
- **Per-thread+Per-warp and Per-CTA:** Develop specialized workload mapping strategies according to neighbor list's size.

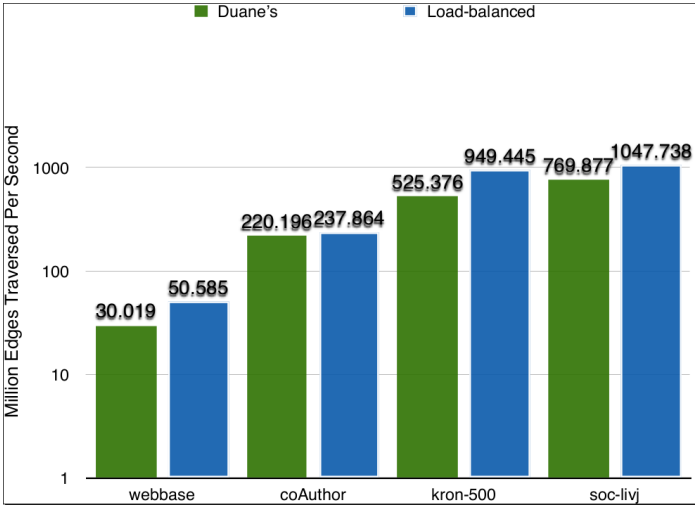


Workload Mapping Strategies (What makes Gunrock traversal fast?)

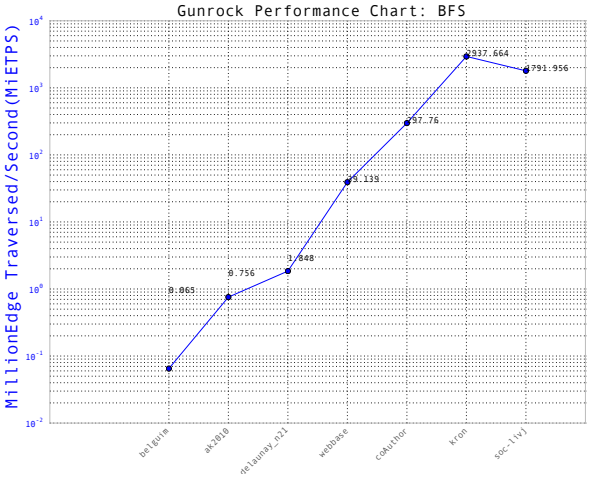
- **Partitioned Load-balancing:** Organize groups of edges into equal-length chunks and assigning each chunk to a block.



Workload Mapping Strategies (What makes Gunrock traversal fast?)

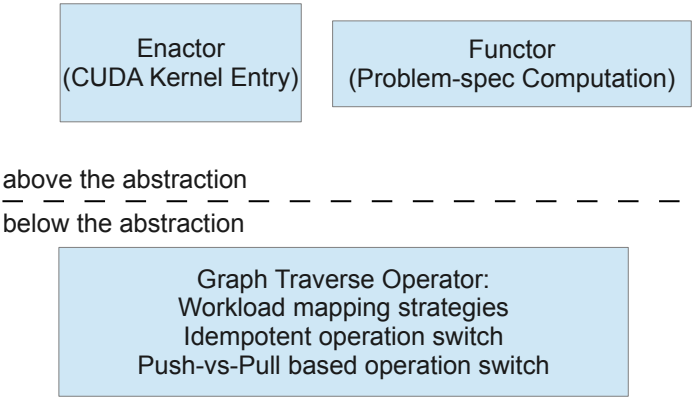


Workload Mapping Strategies (What makes Gunrock traversal fast?)



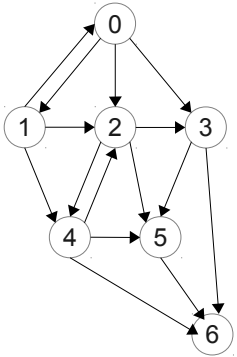
A High-Level View of Programming Model

Bulk iterative processes which iterate between graph traversal and computation using operators and functors.



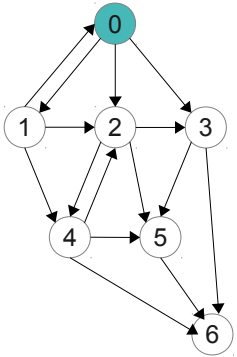
Graph Primitive Example: BFS

A breadth-first search (BFS) algorithm takes a graph and a source vertex s, and computes a breadth-first search tree rooted at s containing all vertices reachable from s. Our implementation will compute the predecessor and the distance from source vertex for each vertex.



Graph Primitive Example: BFS

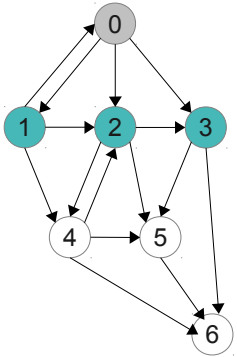
A breadth-first search (BFS) algorithm takes a graph and a source vertex s, and computes a breadth-first search tree rooted at s containing all vertices reachable from s. Our implementation will compute the predecessor and the distance from source vertex for each vertex.



UCDAVIS COLLEGE OF ENGINEERING

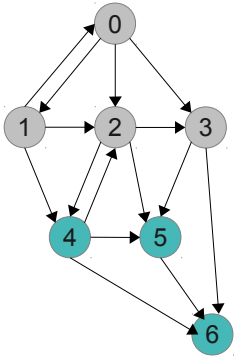
Graph Primitive Example: BFS

A breadth-first search (BFS) algorithm takes a graph and a source vertex s, and computes a breadth-first search tree rooted at s containing all vertices reachable from s. Our implementation will compute the predecessor and the distance from source vertex for each vertex.



Graph Primitive Example: BFS

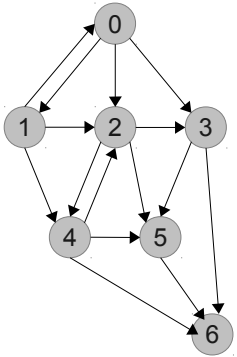
A breadth-first search (BFS) algorithm takes a graph and a source vertex s, and computes a breadth-first search tree rooted at s containing all vertices reachable from s. Our implementation will compute the predecessor and the distance from source vertex for each vertex.



UCDAVIS COLLEGE OF ENGINEERING

Graph Primitive Example: BFS

A breadth-first search (BFS) algorithm takes a graph and a source vertex s, and computes a breadth-first search tree rooted at s containing all vertices reachable from s. Our implementation will compute the predecessor and the distance from source vertex for each vertex.



Performance: Speedup

Primitive	webbase1M	coAuthor	socLiveJournal	Kron_g500
BFS	NA(GPU:0.001s)	4	27	22
CC	3	10	13	NA(GPU:0.253s)
BC	2	5	10	10
SSSP	2	15	10	16
PR	13	NA(GPU:0.1s)	NA(GPU:3.4s)	NA(GPU:29.6s)

UCDAVIS COLLEGE OF ENGINEERING

What's Next?

- More flexible operator set;
- More graph types and data structures;
- External memory and multi-node GPUs support;
- Non-regular Graph Operations/Advanced graph primitives.

UCDAVIS COLLEGE OF ENGINEERING

Acknowledgement

- This work is done with Yuechao Pan and my advisor Prof. John Owens.
- Thanks to Andrew Davidson for his help with SSSP implementation.
- Thanks to Erich Elsen and Vishal Vaidyanathan from Royal Caliber.
- Thanks to Duane Merrill and Sean Baxter from NVIDIA Research.
- Thanks to Dr. Chris White and DARPA XDATA funding.

Thank you!

UCDAVIS COLLEGE OF ENGINEERING