Gunrock: GPU Graph Analytics

Jonathan Wapman, Yangzihao Wang, Yuechao Pan, Andrew Davidson, Yuduo Wu, Carl Yang, Leyuan Wang,
Muhammad Osama, Chenshan Yuan, Weitang Liu, Andy T. Riffel, and John D. Owens

University of California, Davis

Overview

Gunrock is the state-of-the-art CUDA-based library specifically designed for GPU graph analytics. It works on single node single GPU and multi-GPU platforms. Gunrock offers:

- A high-level, bulk-synchronous, data-centric model.
- A balance between performance and expressiveness.

Available Primitives

Gunrock provides a wide range of graph primitives and a high-level programming model for programmers to develop their own primitives on the GPU. The following are some of the available apps:

- Betweenness Centrality
- Breadth-First Search
- Community Detection (Louvain)
- Connected Components
- Geolocation
- Graph Coloring
- Graph Projection
- Hyperlink-Induced Topic Search
- Single Source Shortest Path
- Subgraph Matching
- PageRank
- Random Walks
- Triangle Counting
- Vertex Nomination
- etc.

Higher-level Primitive Call

```
double elapsed_time = gunrock_<primitive>
  (
  parameters,  // Execution parameters
  graph,       // Input graph
  ...          // Add problem specific data
  );
```

Gunrock's Programming Model

Data-Centric Abstraction

- **Frontier:** A compact queue of nodes or edges.
- Manipulation of frontiers is an **operation**.

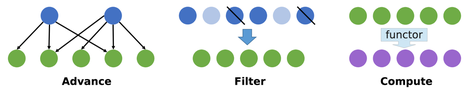


Figure 1: Gunrock's Operators

- **Advance:** Generates new frontier by visiting the neighbors.
- **Filter:** Chooses a subset of current frontier as the new frontier.
- **Compute:** Applies a compute operation on all elements in its input front.
- **Neighbor Reduce:** Uses the advance operator to visit the neighbor list of each item in the input frontier and performs a segmented reduction over the neighborhood generated via the advance.

Bulk-Synchronous Programming

- Series of parallel operations separated by **global barriers**.

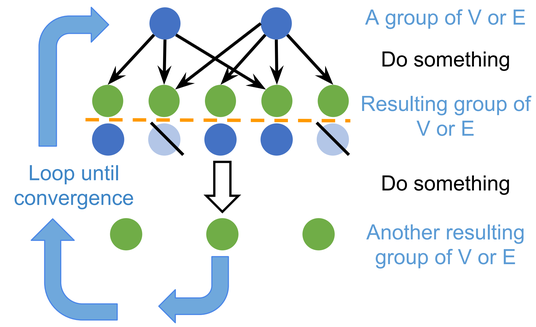


Figure 2: Gunrock's BSP Model

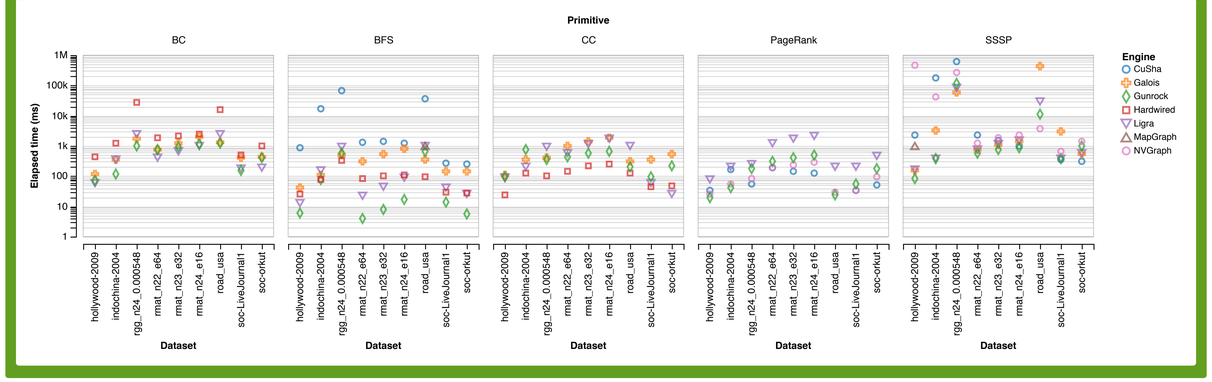
Advance Operator API

```
auto advance_op =
  [ /* Problem specific data */ ]
  __host__ __device__ (
  const VertexT &src,
  VertexT &dest,
  const SizeT &edge_id,
  const VertexT &input_item,
  const SizeT &input_pos,
  SizeT &output_pos) -> bool
{ /* Per-edge advance operation */ };
```

Filter Operator API

```
auto filter_op =
  [ /* Problem specific data */ ]
  __host__ __device__ (
  const VertexT &src,
  VertexT &dest,
  const SizeT &edge_id,
  const VertexT &input_item,
  const SizeT &input_pos,
  SizeT &output_pos) -> bool
{ /* Per-vertex filter operation */ };
```

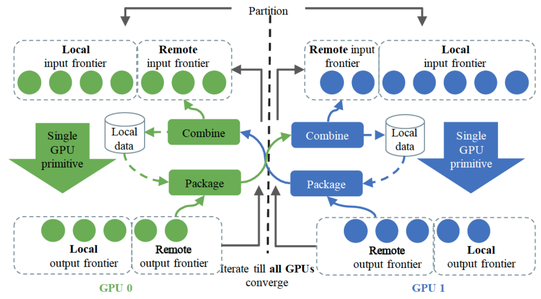
Gunrock's Performance



Optimization Strategies

- Static/dynamic Workload mapping and load-balancing strategy
- Enable idempotent operations
- Pull v.s. push traversal
- Priority Queue, reorganizing frontier items into "near" and "far" slice
- Kernel fusion
- Enable idempotent operations
- Efficient coalesced-memory access

Multi-GPU Gunrock



Future Work

- Expand core operators and new primitives
- Support dynamic/streaming graphs
- Asynchronous graph primitives
- Add meta-linguistic abstraction to support other programing languages
- Software-Defined Hardware

Contact Information

- Web: https://gunrock.github.io
- Email: jdwapman, jowens@ucdavis.edu

GUNROCK
GPU GRAPH ANALYTICS FRAMEWORK AT UC DAVIS

UCDAVIS
UNIVERSITY OF CALIFORNIA