



Gunrock High-Performance Graph Analytics for the GPU

Muhammad Osama — University of California, Davis

Why use GPUs for graph processing?

GPUs and Graphs

Graphs

- Found **everywhere**
 - Road & social networks, web, etc.
- Require **fast processing**
 - Memory bandwidth, computing power and GOOD software
- Becoming **very large**
 - Billions of edges
- **Irregular data access pattern** and control flow
 - Limits performance and scalability

GPUs

- Found **everywhere**
 - Data center, desktops, mobiles, etc.
- Very **powerful**
 - High memory bandwidth (900 GBps) and computing power (15.7 TFlops)
- **Limited memory** size
 - 32 GB per NVIDIA V100
- **Difficult to program**
 - Harder to optimize

What is Gunrock?

A **CUDA**-based
graph processing
library, aims for

Performance

State-of-the-art graph
processing library

A **CUDA**-based
graph processing
library, aims for

Generality

Covers a **broad range** of
graph algorithms

A **CUDA**-based
graph processing
library, aims for

Programmability

Makes it easy to
implement and extend
graph algorithms from
1-GPU to multi-GPUs

A **CUDA**-based
graph processing
library, aims for

Scalability

Fits in (very) limited GPU
memory space

performance scales

when using many GPUs

Where can you find Gunrock?

High-Performance Graph Primitives on GPUs <https://gunrock.github.io/>

Edit

gunrock cuda graph-processing graph-analytics gpu graph-primitives graph-engine Manage topics

2,542 commits 3 branches 0 packages 9 releases 1 environment 32 contributors Apache-2.0

Branch: master New pull request Create new file Upload files Find file Clone or download

 crozhon	Modify device intrinsics to compile on all arches	Latest commit a6c626b 13 days ago
 .github	Create SECURITY.md	8 months ago
 cmake	Use ccbbin option for gencode auto-detection	17 days ago
 dataset	adding test toy dataset	8 months ago
 docker	add docker for ubuntu1604-cuda10.1	3 months ago
 docs @ 5be4ef2	Updating docs to a more latest release.	8 months ago
 doxygen	Updating project version for doxygen.	11 months ago
 examples	Get rid of most warnings.	2 months ago

Project's Workflow

Release ([master](#)) branch

Development ([dev](#)) branch

Git Forking Workflow

Contribute GitHub [Issues](#) &
[Pull Requests](#)

Apache 2.0 License

Code Coverage [codecov.io](#)

Central Integration [jenkins.io](#)

Documentation [slate](#) & [doxygen](#)

Project's Workflow (cont.)

The screenshot displays a web-based roadmap for the Gunrock project, organized into several columns. At the top, it shows 'Roadmap' with a progress bar and 'Updated yesterday'. A search bar labeled 'Filter cards' is present. The roadmap is divided into several categories, each with a list of tasks:

- Possible Research:** Includes links to GitHub docs, SIMD-X programming, Realtime Top-k Personalized PageRank, Low-latency graph streaming, Tigr: Transforming Irregular Graphs, and SHOVE ASIDE, PUSH: THE CASE FOR PULL-BASED GRAPH PROCESSING.
- Algorithms/Operators:** Lists 'Shared Nearest Neighbor (Implementation, correction and optimizations)', 'Union Find Operator', 'All-Pairs Shortest Path (APSP)', 'Planned new primitives using current operator set', 'Stripmining Operator', and 'Making pull-based traversal in dofs a general operator'.
- HIVE (DARPA):** Lists 'Dynamic (mutable) graphs', 'MGPU Application Classification', 'MGPU Graph Projections', 'MGPU SeededGraphMatching', 'MGPU SparseGraphLasso', 'MGPU GraphSAGE', 'MGPU Louvain', and 'MGPU VertexNomination'.
- Research:** Lists 'Adaptive Load Balancer', 'Refactor: Multi-GPU Gunrock Framework', and 'Asynchronous algorithm support'.
- Features:** Lists 'Add file IO support for comma-separated values file', 'Extending Subgraph matching with conditions', 'Filter needs to handle null frontier (complete graph as a frontier input)', 'ENH: Stream capture added to Gunrock', 'Framework requirement for GraphBlas <-> Gunrock and NetworkX -> Gunrock', and 'Improve pointer jumping in CC'.

Gunrock's Roadmap

Some Stats and Stuff! (as of 01/30/2020)

- 32 Contributors (over 2500 commits)
- ~600 stars 148 forks
- NVIDIA CUDA-X: GPU Accelerated Library
- RAPIDS

How does Gunrock work?

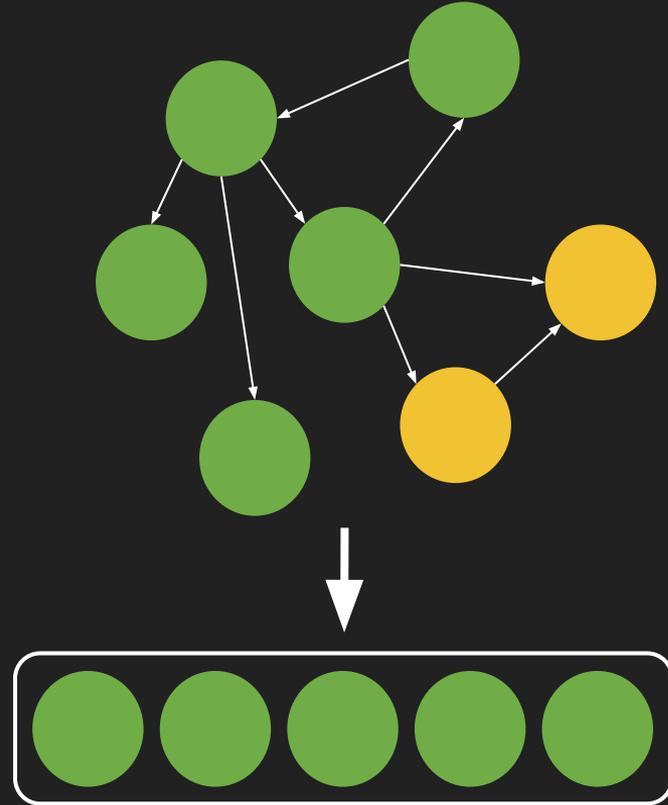
Programming Model

- **Data-centric** abstraction
- **Bulk-synchronous** programming

Frontier

A **frontier**; group of vertices or edges

An example **graph {G}**



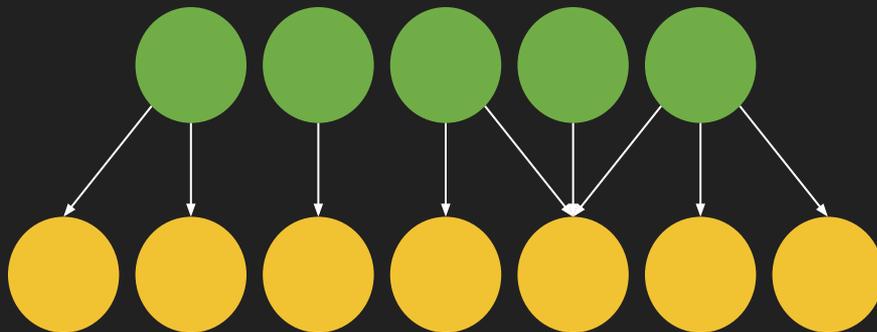
Frontier of **vertices** of **graph {G}**

Parallel Operators

Manipulation of frontiers
is an **operation**

- Advance
- Filter
- For
- Intersection
- Neighbor-Reduce
- ... and more.

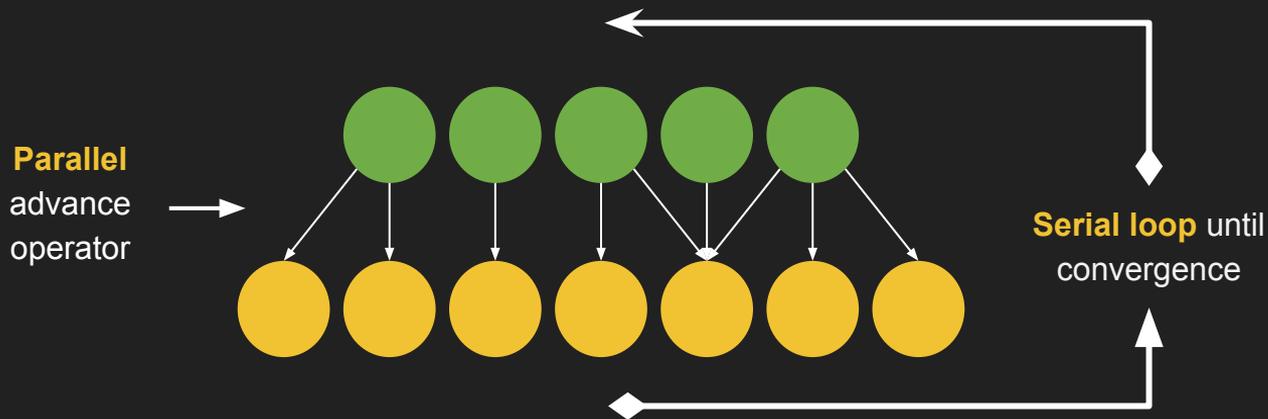
Illustration of **Advance** Operator



Generates new frontier by visiting the neighbors.

Bulk-Synchronous Programming

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



{Gunrock Algorithms}

A* Search

Betweenness Centrality

Breadth-First Search

Connected Components

Graph Coloring

Geolocation

RMAT Graph Generator

Graph Trend Filtering

Graph Projections

Random Walk

Hyperlink-Induced Topic Search

K-Nearest Neighbors

Louvain Modularity

Label Propagation

MaxFlow

Minimum Spanning Tree

PageRank

Local Graph Clustering

GraphSAGE

Stochastic Approach for Link-Structure Analysis

Subgraph Matching

Shared Nearest Neighbors

Scan Statistics

Single Source Shortest Path

Triangle Counting

Top K

Vertex Nomination

Who To Follow

Example application in Gunrock.

Single-Source Shortest Path

Implement the
advance and filter
C++ lambdas for
SSSP

{complete code}

```
auto advance_op =
    [distances, weights] __host__ __device__ (...) -> bool {

    auto distance = distances[vertex_id] + weights[edge_id];
    auto old_distance = atomicMin(distances + neighbor_id, distance);

    if (distance < old_distance) return true;
    return false;
};

auto filter_op =
    [labels, iteration] __host__ __device__ (...) -> bool {
    if (!util::isValid(neighbor_id)) return false;
    return true;
};
```

Single-Source Shortest Path

Launch the
lambdas within
the operator call

{complete code}

```
while (!frontier.isEmpty()) {  
    oprtr::Advance<oprtr::OprtrType_V2V>(  
        graph.csr(), frontier, oprtr_parameters,  
        advance_op, filter_op);  
}
```

NVIDIA AI Laboratory. UC Davis Center for GPU Graph Analytics.

Department of Defense Advanced Research Projects Agency (DARPA). SYMPHONY: Orchestrating Sparse and Dense Data for Efficient Computation. Award HR0011-18-3-0007.

Department of Defense Advanced Research Projects Agency (DARPA). A Commodity Performance Baseline for HIVE Graph Applications. Award FA8650-18-2-7835.

Adobe Data Science Research Award. Scalability and Mutability for Large Streaming Graph Problems on the GPU.

National Science Foundation (Award OAC-1740333) S²-SSE: Gunrock: High-Performance GPU Graph Analytics.

National Science Foundation (Award CCF-1637442) Theory and implementation of dynamic data structures for the GPU. Program: AitF---Algorithms in the Field.

National Science Foundation (Award CCF-1629657) PARAGRAPH: Parallel, Scalable Graph Analytics. XPS---Exploiting Parallelism & Scalability.

Department of Defense Advanced Research Projects Agency (DARPA) SBIR SB152-004. Many-Core Acceleration of Common Graph Programming Frameworks. Phase II: award W911NF-16-C-0020.

Department of Defense Advanced Research Projects Agency (DARPA) STTR ST13B-004 (“Data-Parallel Analytics on Graphics Processing Units (GPUs)”). A High-Level Operator Abstraction for GPU Graph Analytics. Awards D14PC00023 and D15PC00010.

Department of Defense (XDATA program). An XDATA Architecture for Federated Graph Models and Multi-Tier Asymmetric Computing. Oct. 2012--Sept. 2017. Prime contractor: Sotera Defense Solutions, Inc., US Army award W911QX-12-C-0059.

Acknowledgements & Thanks!