Gunrock: A High-Performance Graph Processing Library on the GPU

Yangzihao Wang, Andrew Davidson, Yuechao Pan, Yuduo Wu, Andy Riffel, John D. Owens
University of California, Davis

Objectives

For large-scale graph analytics on the GPU, the irregularity of data access and control flow and the complexity of programming GPUs have been two significant challenges for developing a programmable high-performance graph library. we describe *Gunrock*, our system for graph processing on the GPU.Our goal with Gunrock is to deliver the performance of GPU hardwired graph primitives with a high-level programming model that allows programmers to quickly develop new graph primitives.

Introduction

The superior performance, price-performance, and power-performance capabilities of the modern GPU over the traditional CPU make it a strong candidate for data-intensive applications like graph processing. Previous CPU-based large graph analytics work either uses a serial or coarse-grained-parallel programming model (single-node systems) or has substantial communication cost (distributed systems). GPU low-level implementations of specific graph primitives ("hardwired" primitives) require expert knowledge of GPU programming and optimization. Existing high-level GPU graph processing systems often recapitulate CPU programming models and do not compare favorably in performance with hardwired primitives.

With Gunrock, we design and implement a set of simple and flexible APIs that significantly reduce the code size and the development time and apply to a wide range of graph processing primitives. We also implement several GPU-specific optimization strategies for memory efficiency, load balancing, and workload saving that together achieve high performance.

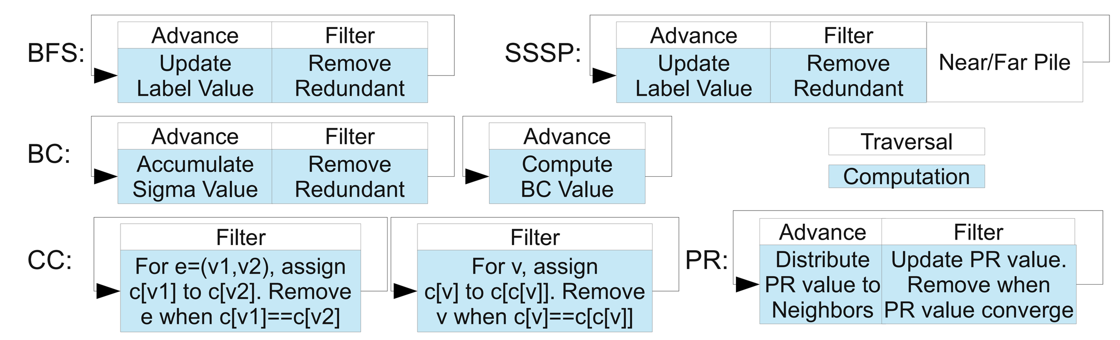
The Gunrock Abstraction

Gunrock targets graph operations that can be expressed as iterative convergent processes. each step operates on a *frontier* of active vertices or edges in the graph. Steps are *bulk-synchronous parallel* (BSP): different steps may have dependencies between them, but individual operations within a step can be processed in parallel.

- A **traverse** step generates a new frontier from the current frontier.
  - *advance* generates a new frontier by visiting the neighbors of the current frontier; According to the direction of the edges, advance can perform both push-style traversal (scatter) and pull-style traversal (gather).
  - *filter* chooses a subset of the current frontier based on programmer-specified criteria.
- A **computation** step defines an operation on all elements (vertices or edges) in the current frontier; Gunrock then performs that operation in parallel across all elements.

Applications

By reusing Gunrock's efficient operators and combining different functors, users can build new graph primitives with minimal extra work. Currently, Gunrock supports graph traversal-based algorithms (Breadth-First Search (BFS) and Single-Source Shortest Path (SSSP)), node ranking algorithms (Betweenness Centrality (BC), PageRank, HITS, SALSA, and Twitter's "Money" [which requires bipartite graph support]), and subgraph-based algorithms (Connected Component Labeling, Minimum Spanning Tree). We are moving forward to more complex graph primitives as well as extending our operators within the current traversal-computation programming model.



Example: Comparing Abstractions on Single-Source Shortest Path

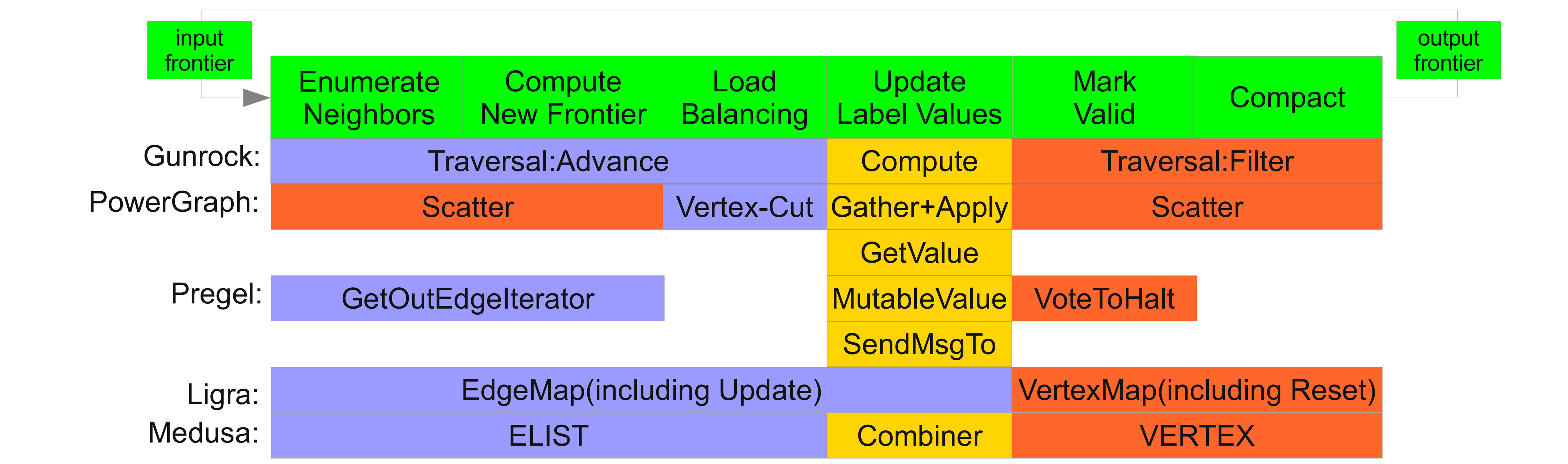


Figure 1: Operations that make up one iteration of SSSP and their mapping to the Gunrock, PowerGraph (GAS), Pregel, Ligra, and Medusa abstractions.

Results

Table 1: Gunrock's runtime comparison with other graph libraries and hardwired GPU implementations. Ligra's timings for PageRank and Gunrock's one-iteration PageRank are in **bold**. Hardwired GPU implementations for each primitive are b40c BFS (Merrill et al., PPoPP '12), deltaStep SSSP (Davidson et al., IPDPS '14), gpu_BC (Sariyüce et al., GPGPU-6 '13), and conn connected component labeling (Soman et al., IPDPSW '10).

		Runtime (ms) [lower is better]						
Alg.	Dataset	BGL	PG	Medusa	MapGraph	Hardwired GPU	Ligra	Gunrock
BFS	soc	816	—	75.82	84.08	37.87	57.4	24.37
	bitc	480	—	1557	142.4	69.22	94.9	67.79
	kron	388	—	46.21	44.29	18.67	13.3	17.28
	roadnet	72	—	223.9	53.44	8.18	51.5	17.16
SSSP	soc	5664	1900	—	225.7	236.7	172	361.6
	bitc	2440	1610	7311	250.9	183.6	133	178.8
	kron	1268	1000	—	124.8	125.1	16.4	105.2
	roadnet	408	5800	1143	76.48	163.7	62.2	140
BC	soc	2120	—	—	—	543.8	264	205.3
	bitc	4840	—	—	—	190.2	271	206.6
	kron	1456	—	—	—	156.1	52.6	246.9
	roadnet	732	—	—	—	256.3	129	100.1
PgRank	soc	49568	9500	—	5431	—	**265**	1927 · **175**
	bitc	20400	8600	48156	2471	—	**240**	651.4 · **79.6**
	kron	33432	2500	—	5702	—	**114**	2766 · **212**
	roadnet	2440	2600	532.8	122.7	—	**13.1**	63.25 · **4**
CC	soc	2176	12802	—	803.8	72	498	110
	bitc	1508	8464	—	612.5	28	6180	58.33
	kron	716	5375	—	260	48	1890	67.21
	roadnet	232	9995	—	1935	8	1320	21.33

Next Steps

- Scalability to multiple GPUs/nodes;
- Higher-level graph primitives;
- In-depth comparison to GAS and Graph BLAS;
- Support for mutable graph/time-series graph.

Contact Information

- Gunrock Website: http://gunrock.github.io
- Author's Email: yzhwang@ucdavis.edu

UCDAVIS COLLEGE OF ENGINEERING

Funding

DoD XDATA, STTR ST13B-004; NSF OCI-1032859, CCF-1017399.