HIGH-PERFORMANCE GRAPH PROCESSING PROGRAMMING MODEL ON THE GPU

Yangzihao Wang

January 29, 2015

University of California, Davis

Topics

Context	Related works on parallel graph processing
Current	Design and implementation of Gunrock
Future	Research problems and next steps

Parallel Large Graph Analytics

Single-node CPU | Distributed CPU | GPU Hardwired | GPU Library

The trade-off between programmability and performance

Challenges for Graph Processing on the GPU

- The irregularity of data access/control flow
- The complexity of programming GPUs

Goal of Gunrock

Deliver the performance of GPU hardwired graph primitives with a high-level programming model that allows programmers to quickly develop new graph primitives

Preliminaries

A graph is an ordered pair $G = (V, E, w_e, w_v)$ comprised of a set of vertices V together with a set of edges E, where $E \subseteq V \times V$.

Our primary working set is a *frontier*. A vertex frontier is a subset of vertices $U \in V$ and an edge frontier a subset of edges $I \in E$.

Methods: Build High-Level Abstraction

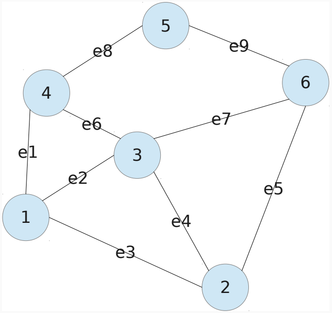
Most graph algorithms have two major operations:

Traverse	Updating a frontier by traversing in the graph or subsetting the current frontier.
Compute	Doing computation on edges or nodes.

Methods: Build High-Level Abstraction

Two ways to traverse:

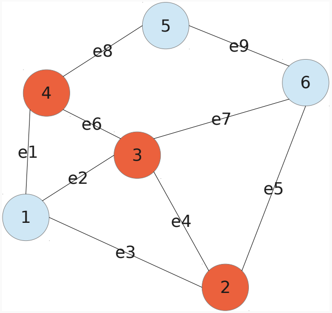
Advance	Generate a new frontier by visiting the neighbors of the current frontier.
Filter	Chooses a subset of the current frontier based on programmer-specified criteria.



Methods: Build High-Level Abstraction

Two ways to traverse:

Advance	Generate a new frontier by visiting the neighbors of the current frontier.
Filter	Chooses a subset of the current frontier based on programmer-specified criteria.

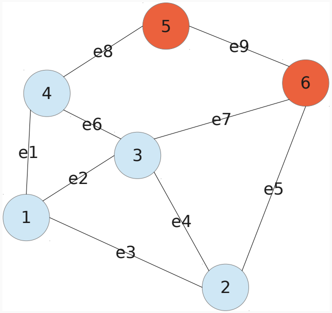


Methods: Build High-Level Abstraction

Two ways to traverse:

Advance Generate a new frontier by visiting the neighbors of the current frontier.

Filter Chooses a subset of the current frontier based on programmer-specified criteria.

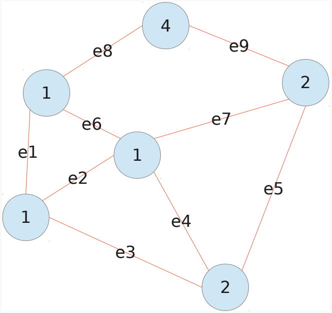


Methods: Build High-Level Abstraction

Two ways to traverse:

Advance Generate a new frontier by visiting the neighbors of the current frontier.

Filter Chooses a subset of the current frontier based on programmer-specified criteria.

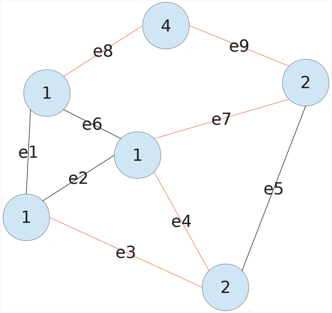


Methods: Build High-Level Abstraction

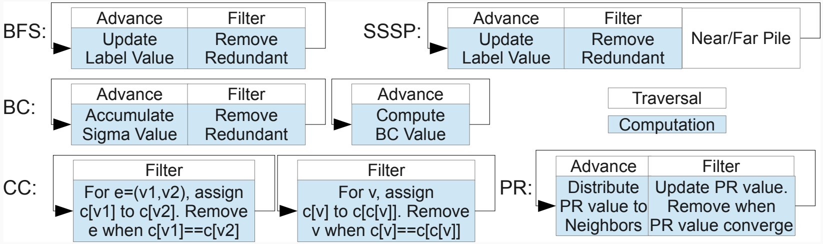
Two ways to traverse:

Advance Generate a new frontier by visiting the neighbors of the current frontier.

Filter Chooses a subset of the current frontier based on programmer-specified criteria.



Methods: Build High-Level Abstraction



Methods: Develop Low-Level Optimization

- Workload Mapping for Advance
- Improving Work Efficiency
- Primitive-Specific Optimization

Methods: Develop Low-Level Optimization

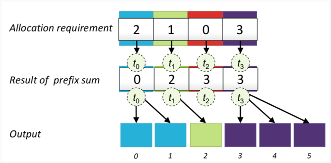
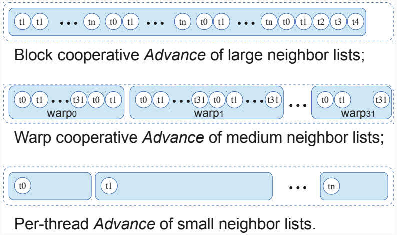


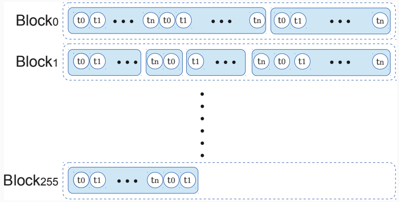
Figure: Merrill et al. PPoPP'12



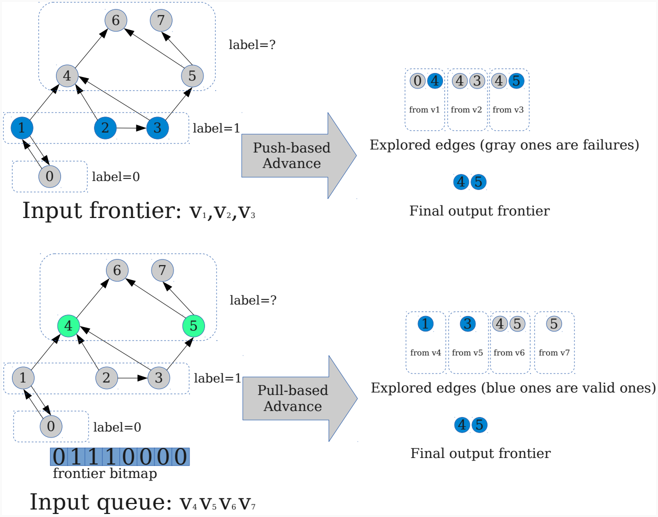
Block cooperative *Advance* of large neighbor lists;

Warp cooperative *Advance* of medium neighbor lists;

Per-thread *Advance* of small neighbor lists.



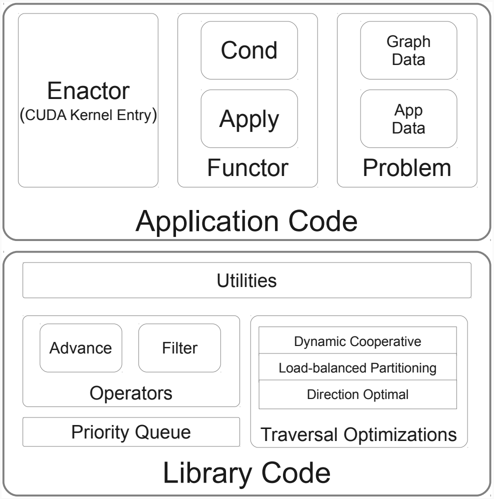
Methods: Develop Low-Level Optimization



Methods: Develop Low-Level Optimization

Other optimizations: Priority Queues, Idempotent Operation, Optimizing Filter, and Output Frontier Storage, etc..

Software Framework



Performance Evaluation

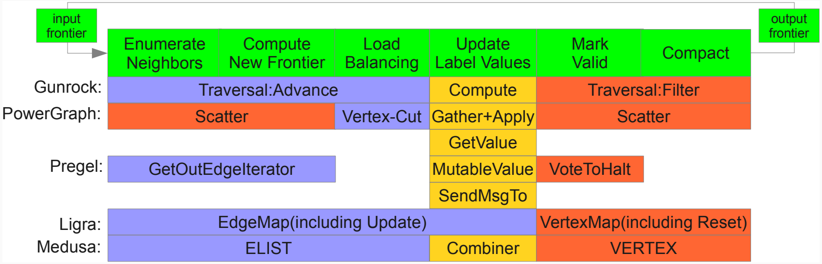
- 10x better than BGL and PowerGraph;
- Always better than other programmable GPU;
- On par with Ligra and hardwired GPU.

Open Questions:

- What is the right programming model? (Expressivity, Mutable Graph, and Performance Model)
- How to expand the current framework? (To Graph BLAS, Multi-Node GPUs, and Out-of-Core)

Open Questions:

What is the right programming model for graph processing on the GPU?



	Enumerate Neighbors	Compute New Frontier	Load Balancing	Update Label Values	Mark Valid	Compact
Gunrock:	Traversal:Advance			Compute	Traversal:Filter	
PowerGraph:	Scatter		Vertex-Cut	Gather+Apply	Scatter	
Pregel:	GetOutEdgeIterator			GetValue MutableValue SendMsgTo	VoteToHalt	
Ligra:	EdgeMap(including Update)				VertexMap(including Reset)	
Medusa:	ELIST			Combiner	VERTEX	

Open Questions:

Expressivity Does current model cover all operations? How difficult to express one operation using current model?

Primitive	Library	Description	Gunrock Implementation
Filter	Help	Remove elements from filter	Filtering
FS	Help	Form Supervertex	Filtering+Sort +Reduce+Scan+Advance +Filtering+Sort
ANV	Help	Aggregating Neighbor Values	Advance
LUV	Help	Local Update of Vertices	Filtering
UVUOV	Help	Update Vertices Using One Other Vertex	Filtering
AGV	Help	Aggregate Global Value	Filtering
Gather	PowerGraph	Aggregating Neighbor Values	Advance
Apply	PowerGraph	Local Update of Vertices	Filtering
Scatter	PowerGraph	Update Neighbor Values	Advance

Open Questions:

Expressivity How to support mutable graph? Two sources of mutability:

- **Algorithm** MST, Clustering, Mesh Refinement, etc. need new operators such as mergeEdge, formSuperVertices, reshapeSubgraph, ...
- **Data** Incrementally computation of node ranking or betweenness centrality in time-series graphs. New users and new links in Twitter's social graph appear constantly. How do we support that?

Open Questions:

Performance Model How to build a performance model to help us improve the current programming model?

- Runtime as a function of # iterations;
- Runtime as a function of {edges, vertices, etc.} touched/traversed;
- Runtime as a function of graph parameters.

Open Questions:

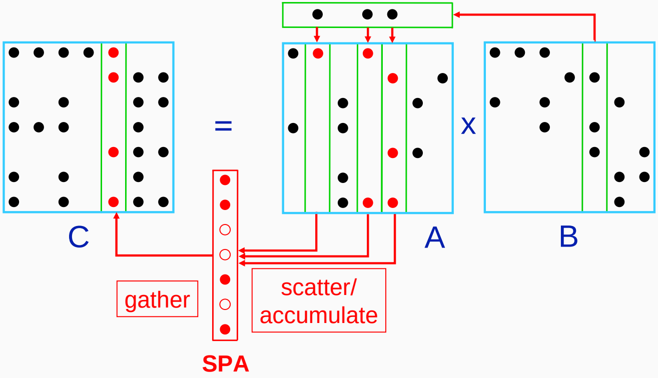
Performance Model How to build a performance model to help us improve the current programming model?

- How efficient the programming model is at exploiting parallelism?
- How performance changes as parallelism increases?
- How well we do at GPU-low-level performance metrics? (like degree of memory coalescing, branch coherence, etc.)
- How much computational work/memory bandwidth do we incur?
- How to shift workload from memory access to computation?

Open Questions:

Graph BLAS style: How to fit in?

Sparse matrix sparse vector multiplication (SpMSpV): Linear combination of columns specified by nonzero elements of the sparse vector.



Open Questions:

Beyond A Single GPU: How to utilize more computing power and larger memory space?

- Single-Node Multi-GPUs
- Out-of-Core
- Multi-Node GPUs

Open Questions:

Single-Node Multi-GPUs

Approach:

- Partition into subgraph, duplicate remote nodes
- Multithreaded host program with multistream support
- Reuse single-node code, partitioner as a plug-in

Issues:

- Best partitioner yet to be found
- Runtime bounded by diameter and # of iterations
- Scaling factor drops?

Open Questions:

Single-Node Multi-GPUs

Results:

- BFS scales across multiple GPUs when #iteration is small(<10), #edge is high (>1̃00M) and average degree is high (>80). For the best case, 3.7x speedup using 6 GPUs (kron_n21)
- SSSP 1.3x speedup using 2 GPUs, BC 2.7x speedup using 4 GPUs.

Open Questions:

Out-of-Core

Approach:

- Borrow the Partitioner + Single-Node Gunrock pattern
- Overlap data movement with computation

Issue:

- Communication cost between each data chunk would still take up the majority of the time.

Open Questions:

Multi-Node Multi-GPUs

- Target: A scalable multi-node layer using MPI
- Approach: Add network communication operation into the single-node multi-GPU version
- Issue: Global partitioner and auxiliary arrays, make good use of NVLink.

Acknowledgment

Gunrock team	Yuechao Pan, Yuduo Wu, Carl Yang, Andrew Davidson, Andy Riffel and John D. Owens
Royal Caliber team	Erich Elsen and Vishal Vaidyanathan
NVIDIA	Duane Merrill, Sean Baxter, the amazing GPU cluster and all other general technical supports
DARPA	XDATA Program, Eric Whyne of Data Tactics Corporation, and Dr. Christopher White.

QUESTIONS?