GraphReduce: Processing Large-Scale Graphs on Accelerated-Based Systems

Shuaiwen Leon Song
Pacific Northwest National Lab

Dipanjan Sengupta, Kapil Agarwal, Karsten Schwan
Georgia Institute of Technology

Graphs are Ubiquitous

Social Network · Web Graph · Advertising · Infrastructure · Science

- **High Volume:** Billions of edges and vertices carrying rich metadata
- **High Velocity:** 100s of billions photos, posts, tweets etc per month
- **Fast graph analytics on large graphs**

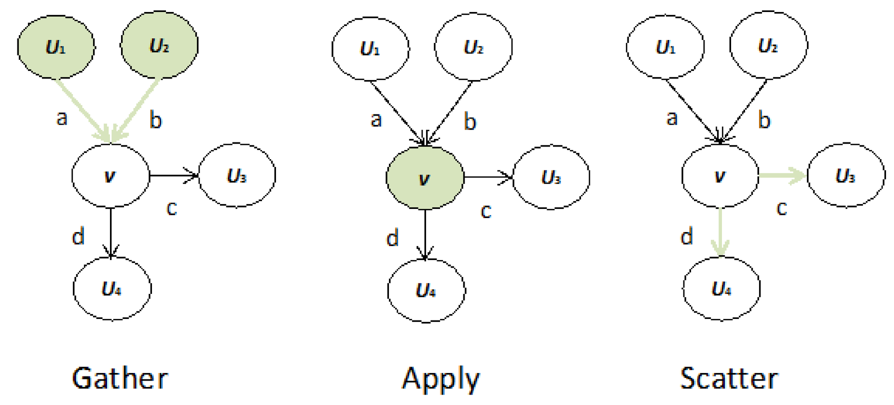
Motivation

- **Why use GPUs ?** – GPU-based frameworks are orders of magnitude faster
- Previous GPU-based graph processing doesn't handle datasets that **doesn't fit in GPU memory**
  - Yahoo-web graph with 1.4 billion vertices requires 6.6 GB memory just to store its vertex values.
- Several challenges in large-scale graph processing
  - How to partition the graph ?
  - How and when to move the partitions between host and GPU ?
  - How to best extract multi-level parallelism in GPUs ?

Graphs	X-Stream (ms)	CuSha(ms)	Speedup
ak2010	215.155	7.75	28x
belgium_osm	2695.88	791.299	3x
coAuthorsDBLP	1275	11.553	110x
delaunay_n13	80.89	5.184	16x
kron_g500-logn20	46550.7	119.824	389x
webbase-1M	3909.12	13.515	290x

GAS Programming Model

- Gather phase: each vertex aggregates values associated with its incoming edges and source vertices
- Apply phase: each vertex updates its state using the gather result
- Scatter phase: each vertex updates the state of every outgoing edge.



Hybrid Programming Model

```
vertex_scatter (vertex v) :
  send updates over outgoing edges of v
vertex_gather (vertex v) :
  apply updates from inbound edges of v

while not done :
  for all vertices v that need to scatter updates
    vertex_scatter (v)
  for all vertices v that have updates
    vertex_gather (v)
```

Vertex-centric GAS

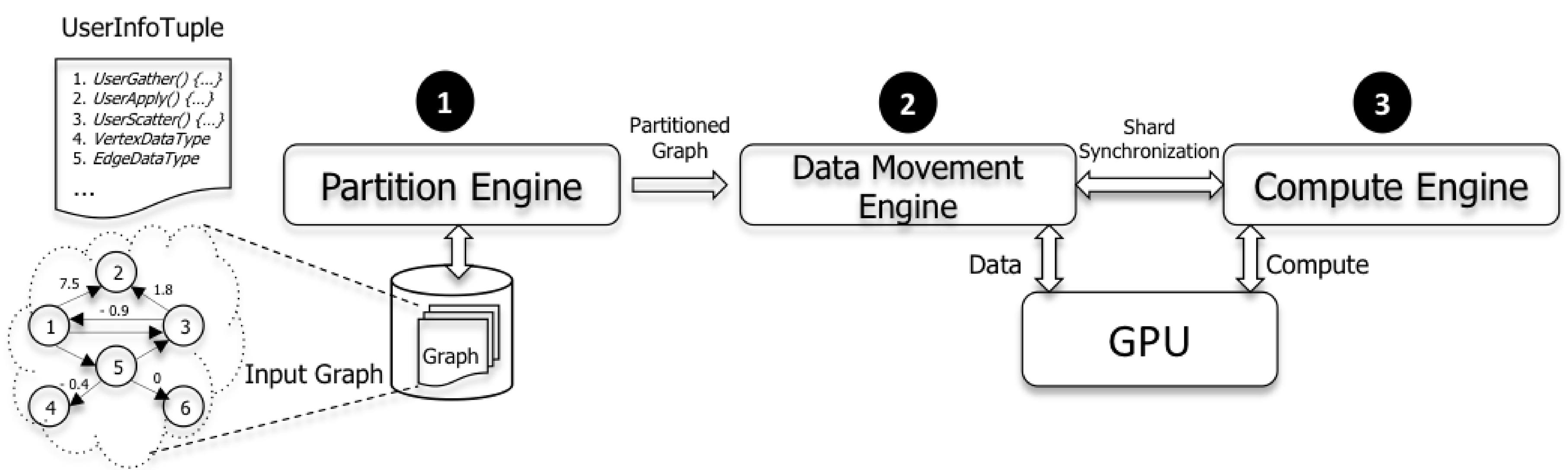
```
edge_scatter (edge e) :
  send update over e
update_gather (update u) :
  apply update u to u.destination

while not done :
  for all edges e
    edge_scatter (e)
  for all updates u
    update_gather (u)
```

Edge-centric GAS

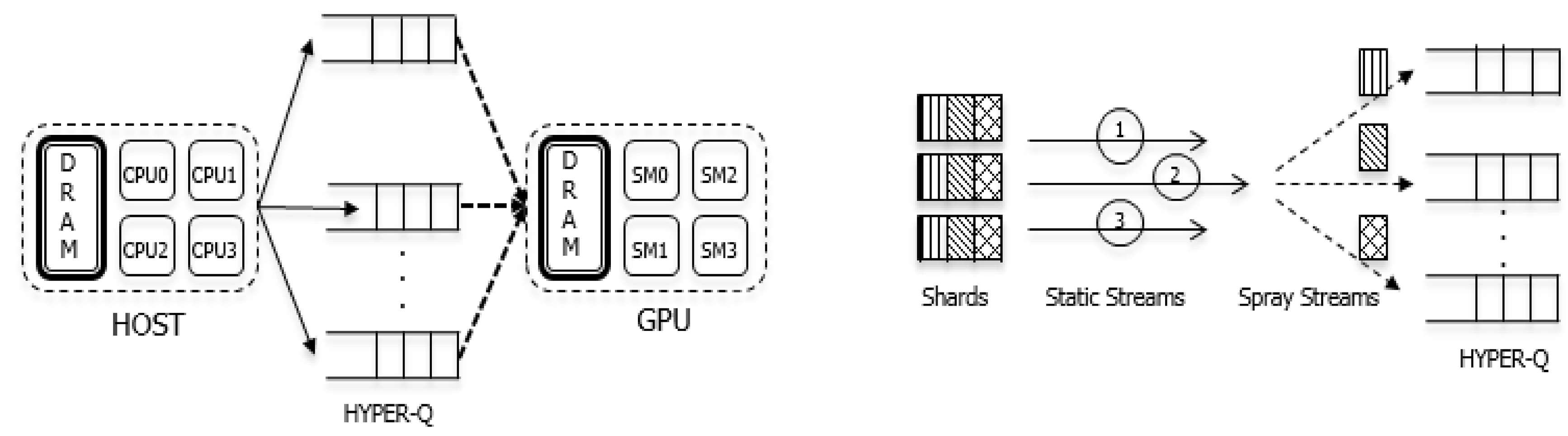
- Existing systems choose either vertex- or edge-centric GAS programming model for graph execution
- Different processing phases have different types of parallelism and memory access characteristics
- GraphReduce adopts a hybrid model with a **combination of both vertex- and edge-centric model**

GraphReduce Architecture



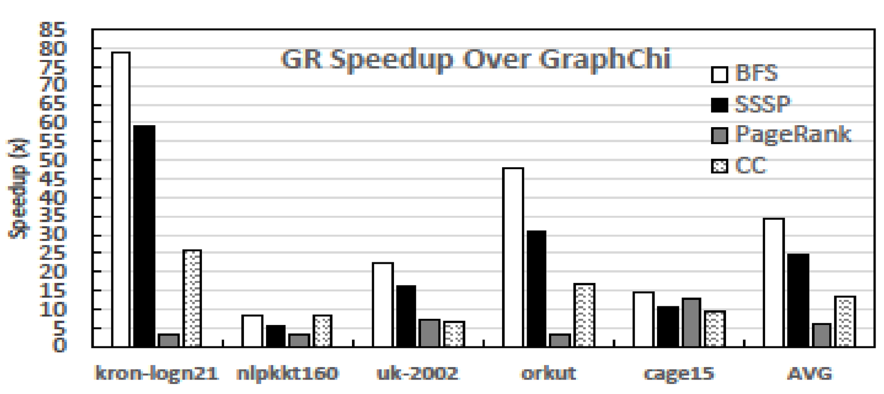
- User defined functions: *gatherMap(), gatherReduce(), apply() and scatter()*
- User defined graph data types : *VertexDataType* and *EdgeDataType*

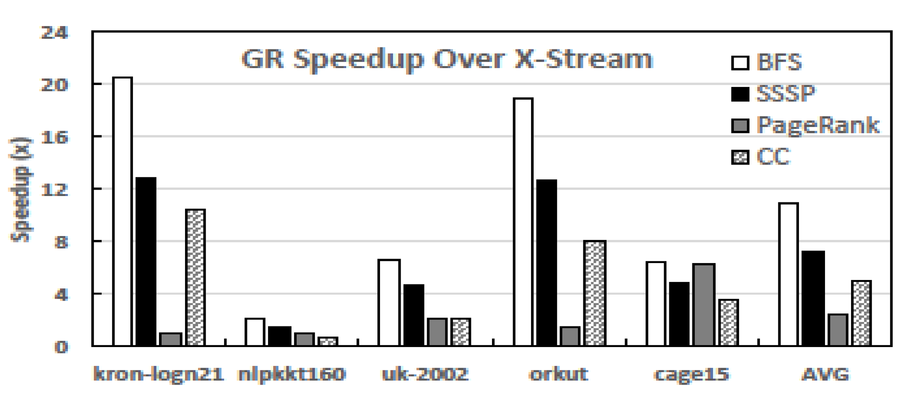
Optimizations



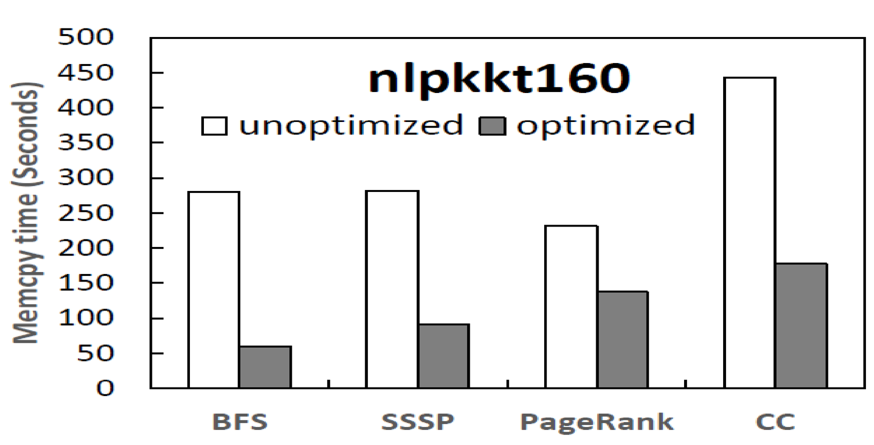
- Asynchronous execution and Spray (deep-copy) operation
- Dynamic frontier management
- Dynamic phase fusion and elimination

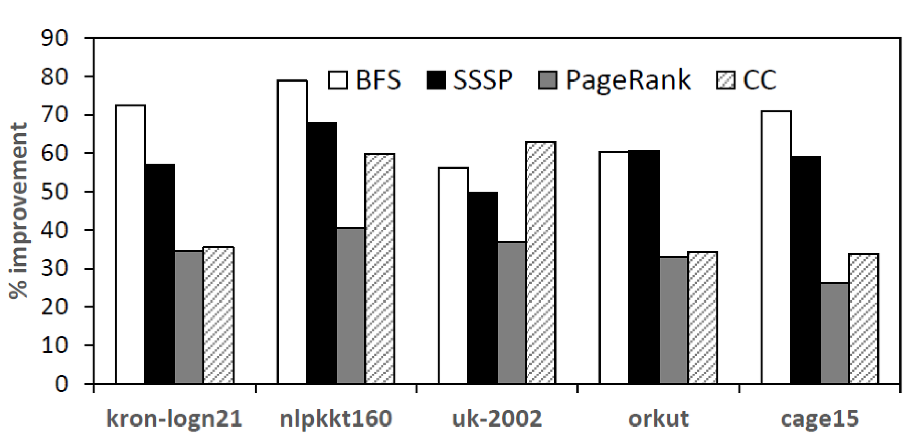
Results





GraphReduce's speedup over Graphchi and X-Stream for out-of-memory graph inputs





Benefits of GraphReduce optimizations over memcpy time

Conclusions

- **GraphReduce** develops a high performance graph processing framework for input datasets that may or may not fit in GPU memory
- Adopts a hybrid model of a combination of both edge- and vertex-centric implementation of GAS programming model
- Leverages CUDA streams and hardware supports like hyper-Qs to stream data in and out of GPU for high performance
- Optimizations like dynamic phase fusion/elimination and frontier management further reduces data transfer time
- Outperforms CPU-based out-of-core graph processing frameworks across a variety of real data sets achieving up to **79x** speedup

Intel Science & Technology Center for Cloud Computing
Carnegie Mellon University · Georgia Tech · intel · Princeton University · UC Berkeley · University of Washington