GARAPH
EFFICIENT GPU-ACCELERATED GRAPH PROCESSING ON A SINGLE MACHINE WITH BALANCED REPLICATION

By Lingxiao Ma, Zhi Yang, Han Chen, Jilong Xue and Yafei Dai
Presented by Brian Wheatman
Goals of Garaph

- Use all resources of machines
  - Large Memory
  - Fast Secondary Storage
  - CPU
  - GPU

- Prior issues
  - Skewed degree distribution
    - Write contention
    - Work imbalance
Graph Representation

- **Shards**
  - *Disjoint set of vertices along with the incoming edges*
    - Increasing order of destination
  - *Sized to fit in the GPUs shared memory (about 12k vertices)*
  - *Replicated*

- **Pages**
  - *A set of shard for efficient movement of data*

- **CSC for incoming edges**
  - *pull*

- **CSR for outgoing edges**
  - *Poor behavior on GPUs*
  - *Notify pull*
    - Only neighbors of the active set pull
System Architecture

- **Dispatcher**
  - Loading graph
  - Distributing computation
  - Distribute data onto multiple SSDs

- **GPU/CPU computation kernel**
  - All blocks processed in parallel
  - Only pull on GPU
  - Both pull and notify pull on CPU
  - Can run either synchronously or asynchronously
    - When asynchronous updates are immediately visible
- **Fault Tolerance**
  - Periodically writes state to secondary storage
Programming API’s

**Gather (Reduce)**
Accumulate information about neighborhood

User Defined:
- Gather
- \(\sum_1 \oplus \sum_2 \rightarrow \sum_3\)

**Apply**
Apply the accumulated value to center vertex

User Defined:
- Apply
- Activate

**Scatter**
Update adjacent edges and vertices.

User Defined:
- Sum
- Apply

*GAS figure from PowerGraph slides

```cpp
interface GASVertexProgram(u) {
  gather(D_u, D_{(u,v)}, D_v) \rightarrow Accum
  sum(Accum left, Accum right) \rightarrow Accum
  apply(D_u, Accum) \rightarrow D_u^{new}
  activate(D_u^{new}, D_u) \rightarrow A[u]
}

template<typename T>
__host__ __device__ void unifiedAdd(T *addr, T val) {
  // For GPU, atomic operation
  atomicAdd(addr, val);
  // For CPU, non-atomic operation
  *addr += val;
}
```
GPU-Based Graph Processing

- Graph Processing Engine
- Replication-Based Gather
Graph Processing Engine

- Vertices stored in GPU global memory
- Each SM has a local copy of the vertices of the shard
  - Gather by reading from global memory updating the local copy
  - Then written back to global memory on GPU
  - After round GPU global memory synchronized with host memory
Replication-Based Gather

- To avoid write contention
  - *Within a shard lots of edges going to the same node*
  - Made worse by natural graphs power law distribution
  - *Replicate the node data and sum up partial values then accumulate*

- Customized replication
  - $O(N) \rightarrow O(\log N), N \leq 32$
  - Modeling: balance profits and costs
  - $R_i = 2^{\min\left\{\left\lceil \log \frac{|V_i|}{|V|} \right\rceil - 0.5, 0.5\right\}}.$
CPU-Based Graph Processing

- Processing with Edge Partitions
  - Edges or split up equally into different partitions
  - Vertexes split are duplicated
    - Later aggregated to obtain value

- Dual-Mode Processing Engine
Evaluation

- Comparison with Other Systems
- Customized Replication
- Dual Modes of the CPU Kernel
- Hybrid CPU-GPU Scheduling

| Graph       | $|V|$  | $|E|$  | Max in-deg | Avg deg | Size edgelist |
|-------------|-------|-------|------------|---------|---------------|
| uk-2007@1M  | 1M    | 41M   | 0.4M       | 41      | 0.6GB         |
| uk-2014-host| 4.8M  | 51M   | 0.7M       | 11      | 0.8GB         |
| enwiki-2013 | 4.2M  | 0.1B  | 0.4M       | 24      | 1.7GB         |
| gsh-2015-tpd| 31M   | 0.6B  | 2.2M       | 20      | 10GB          |
| twitter-2010| 42M   | 1.5B  | 0.8M       | 35      | 27GB          |
| sk-2005     | 51M   | 1.9B  | 8.6M       | 39      | 35GB          |
| renren-2010 | 58M   | 2.8B  | 0.3M       | 48      | 44GB          |
| uk-union    | 134M  | 5.5B  | 6.4M       | 41      | 0.1TB         |
| gsh-2015    | 988M  | 34B   | 59M        | 34      | 0.7TB         |
Comparison with Other Systems

- 10 iterations Pagerank
Comparison with Other Systems

- Connected components
- Until convergence
- GPU can be much slower
Customized Replication

- Helps some pages dramatically
Dual Modes of the CPU Kernel

- On some iterations notify pull is much better
Hybrid CPU-GPU Scheduling

(a) SSSP: twitter-2010
(b) SSSP: renren-2010
(c) CC: twitter-2010
(d) CC: renren-2010