USENIX ATC'17, Santa Clara, CA, USA

Garaph: Efficient GPU-accelerated Graph Processing on a Single Machine with Balanced Replication

Lingxiao Ma †, Zhi Yang †, Han Chen †, Jilong Xue ‡, Yafei Dai *

† *Computer Science Department, Peking University*

‡ *Microsoft Research*

* *SPCCTA, Peking University*



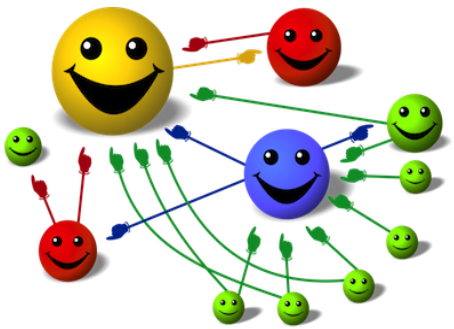
Microsoft Research

Large-Scale Graph Processing



10^{10} pages, 10^{12} tokens

PageRank



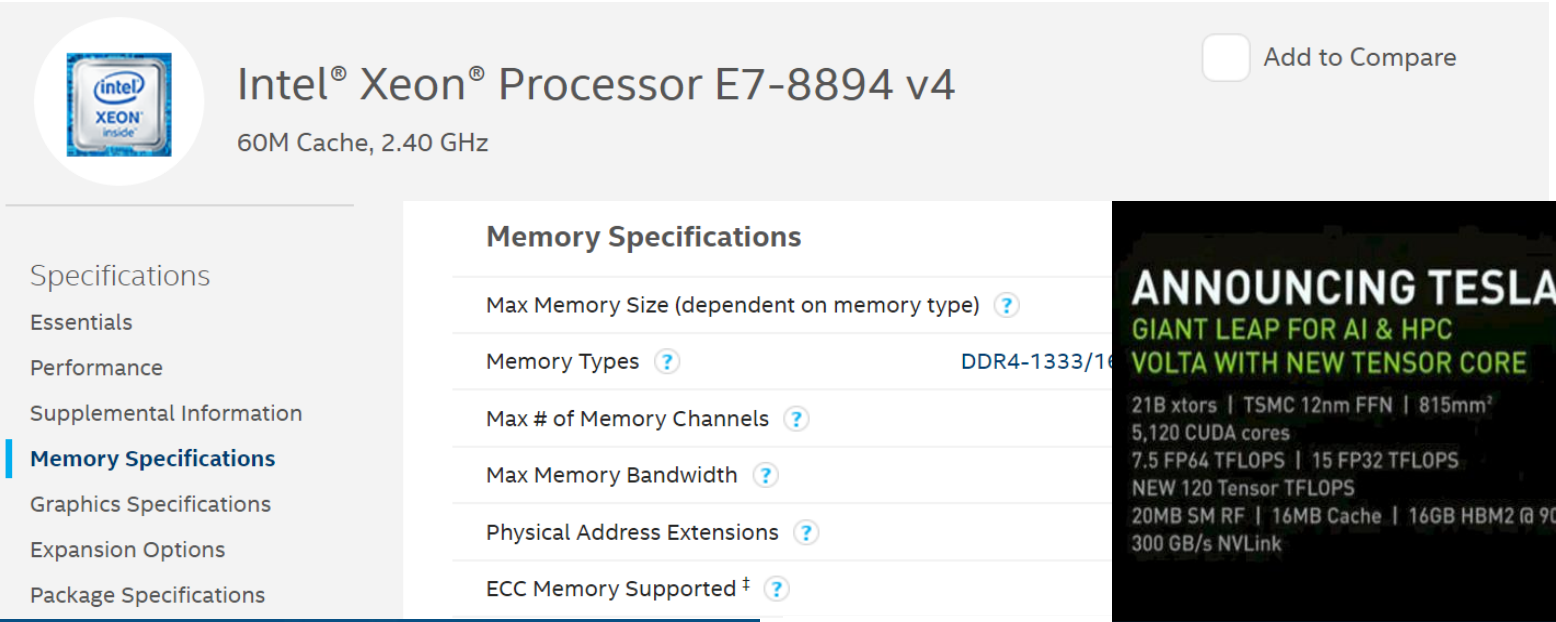


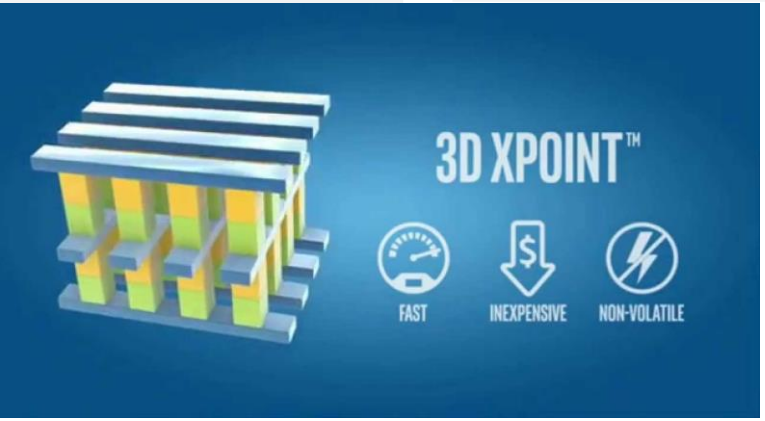
10^{9} nodes, 10^{12} edges

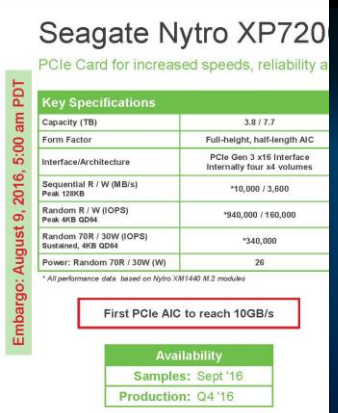
Social Network Analysis

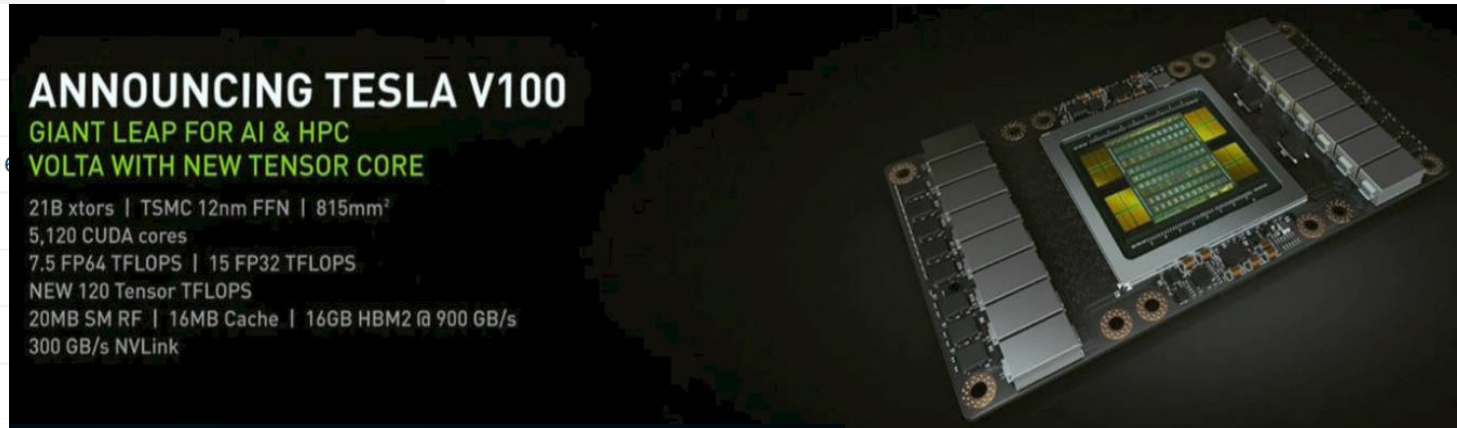


Powerful Storage & Computation Technologies









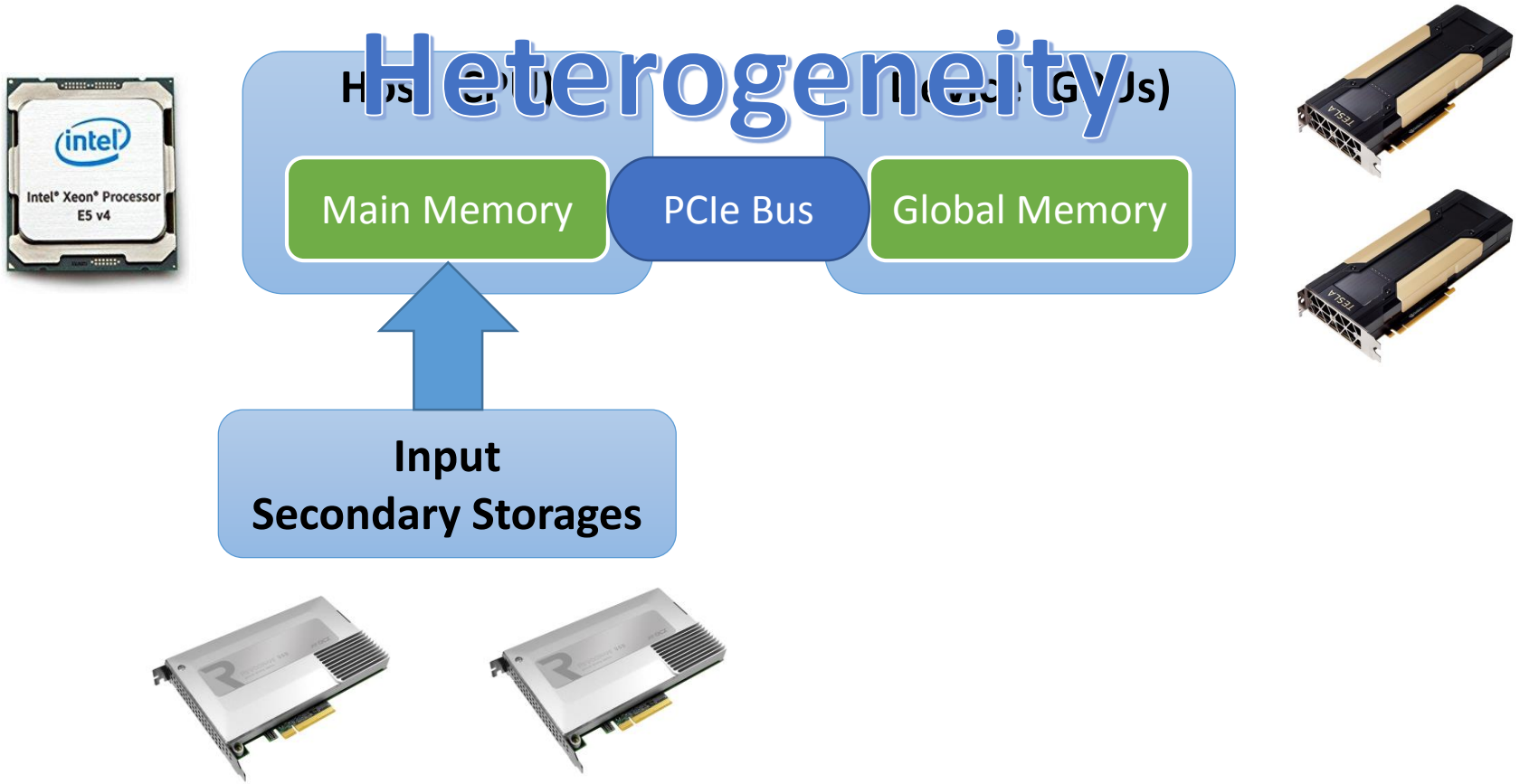




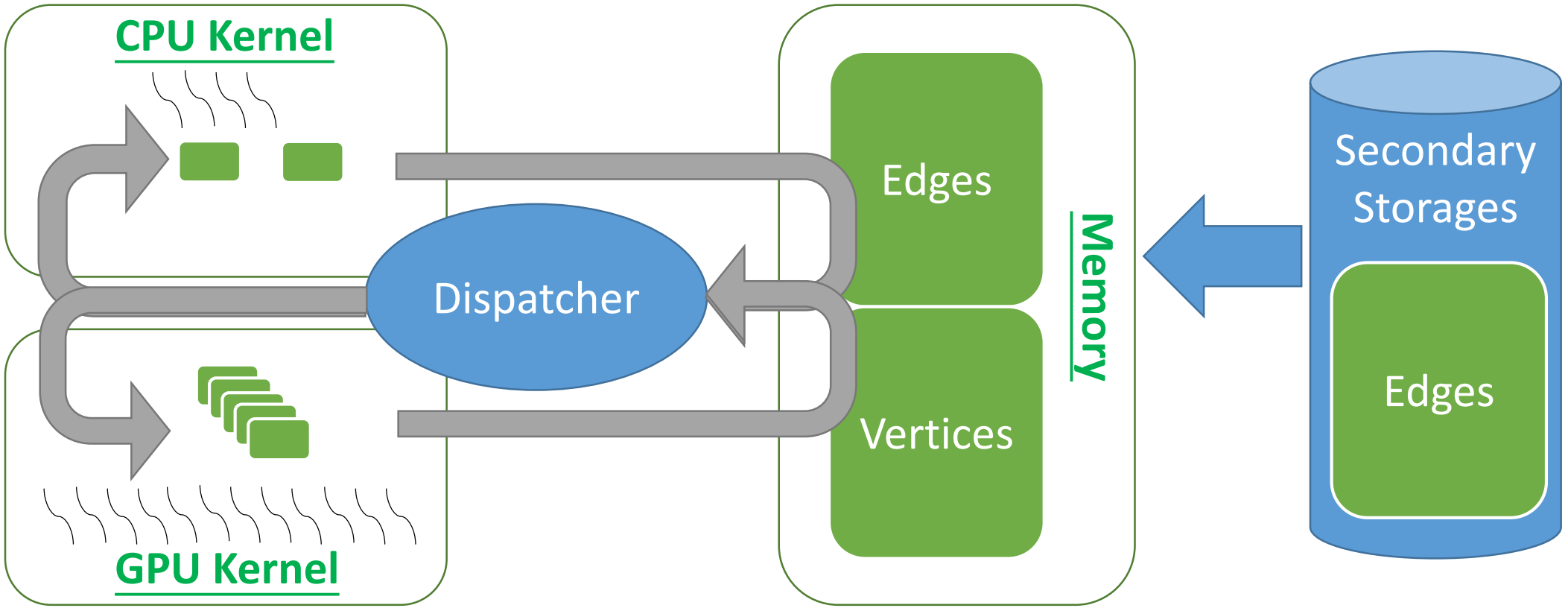
* Figure from Internet

Our Goal

- Large Memory + Fast Secondary Storages
- CPU+GPUs

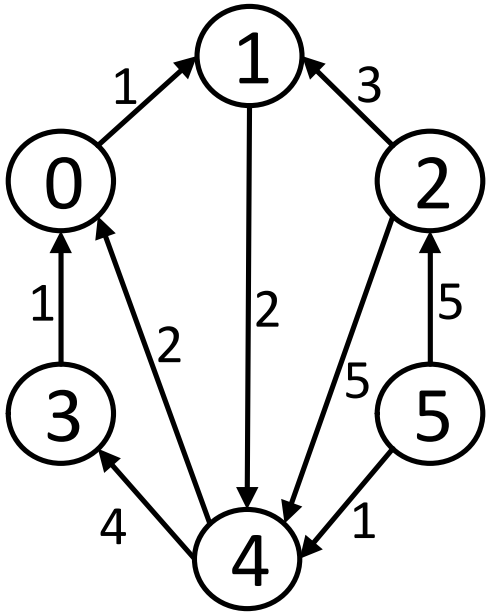


Architecture



Graph Representation for Hybrid CPU and GPU

- CSC & CSR representation
- Shard: vertex interval
- Page: batched shards







Idx
Nbr
Edge
IdxOff

CSC (incomming edges)

0	2	4	5	6	9	9		
3	4	0	2	5	4	1	2	5
1	2	1	3	5	4	2	5	1
0	0	1	1	2	3	4	4	4

Shard 0 | Shard 1

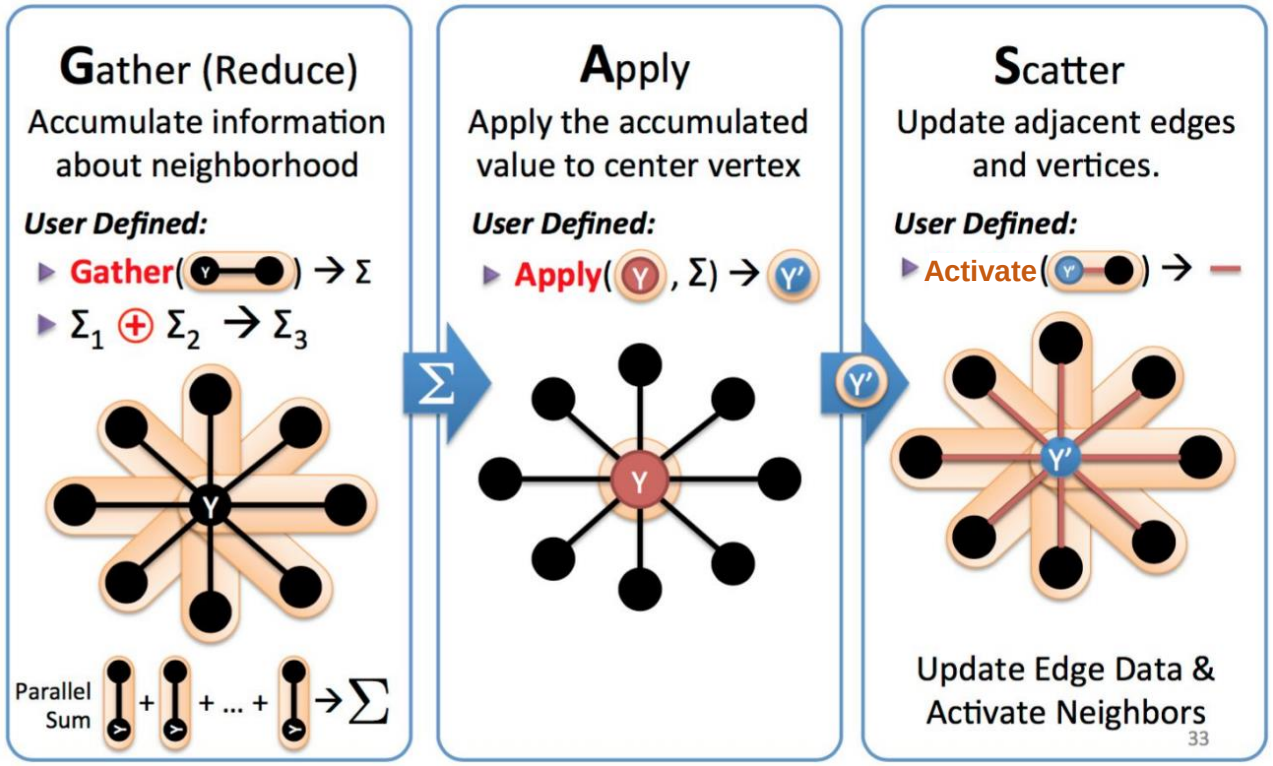
Idx
Nbr

CSR (outgoing edges)

0	1	2	4	5	7	9		
1	4	1	4	0	0	3	2	4

Programming APIs

- GAS Decomposition
- One program for both CPU and GPU

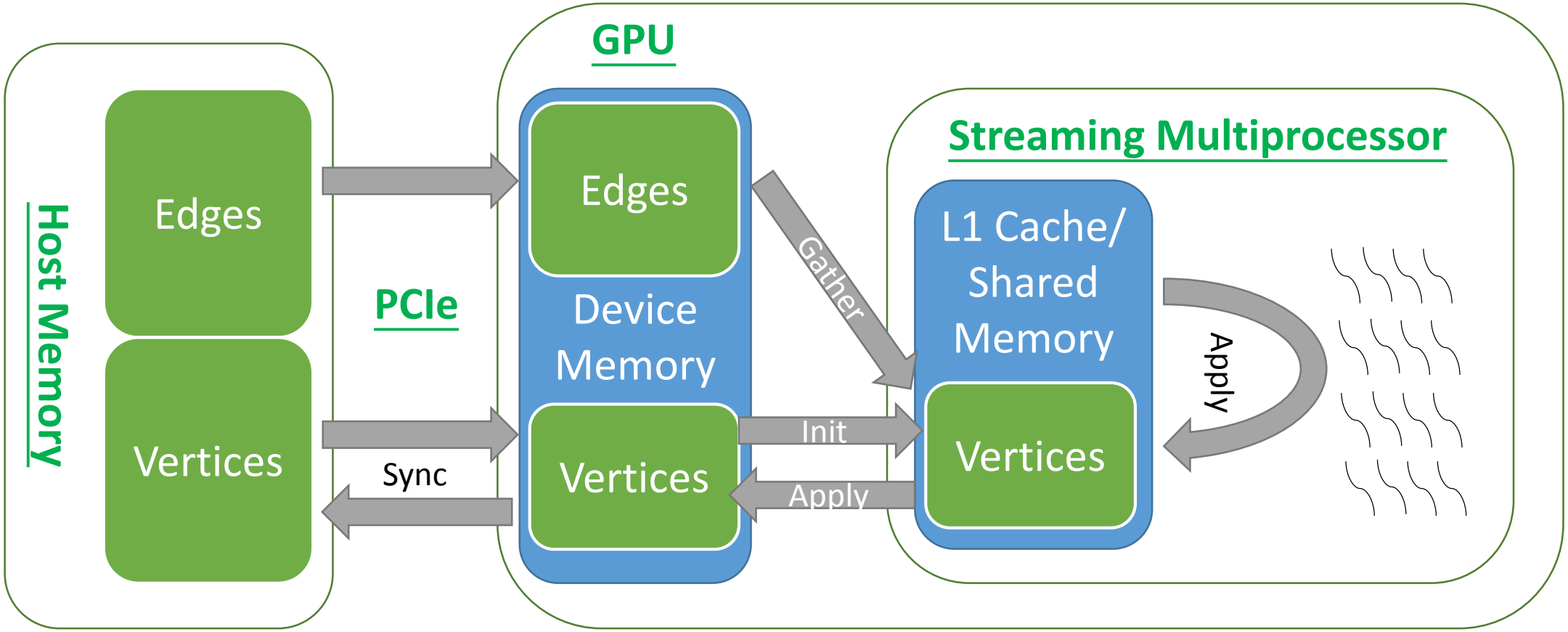



```
interface GASVertexProgram(u)
  gather(D_u, D_(u,v), D_v) → Accum
  sum(Accum left, Accum right) → Accum
  apply(D_u, Accum) → D_u^new
  activate(D_u^new, D_u) → A[u]
```

```
template<typename T>
__host__ __device__ void unifiedAdd(T *addr, T val) {
#ifdef __CUDA_ARCH__ // For GPU, atomic operation
    atomicAdd(addr, val);
#else // For CPU, non-atomic operation
    *addr += val;
#endif
}
```

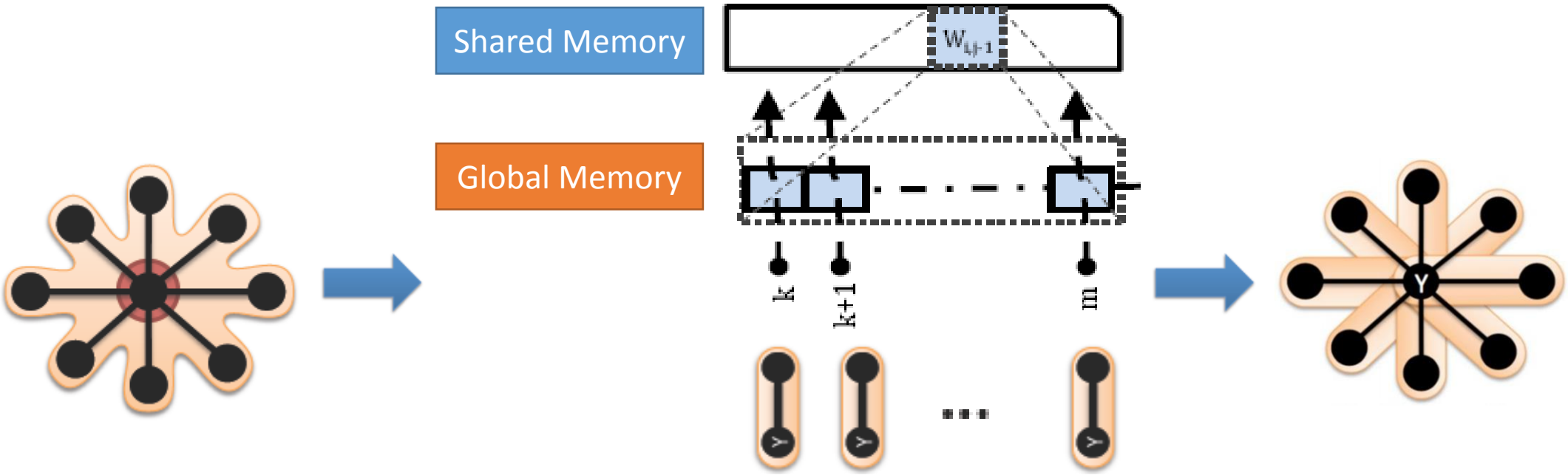
*GAS figure from PowerGraph slides

GPU Computation Kernel

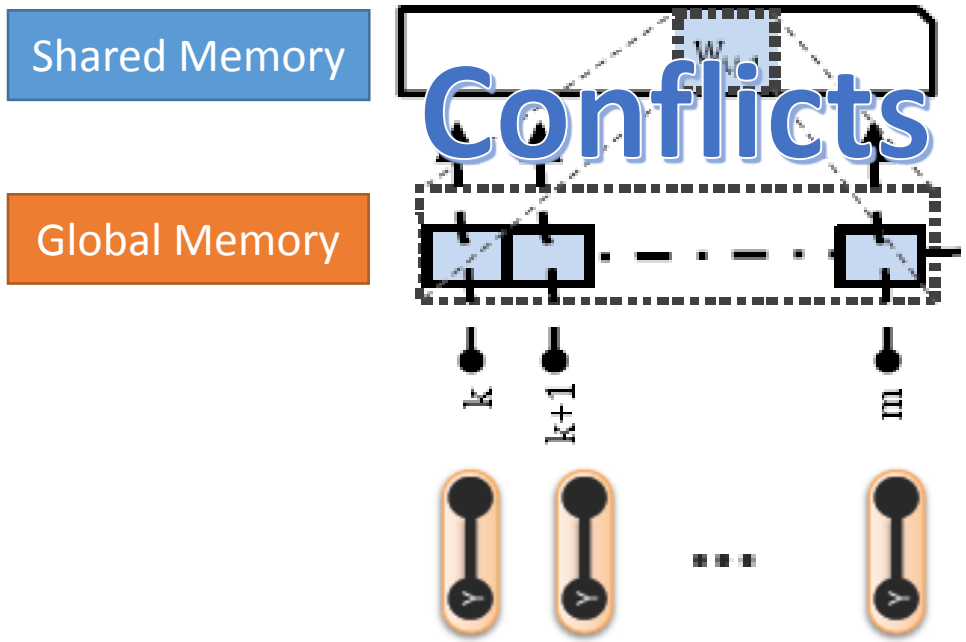


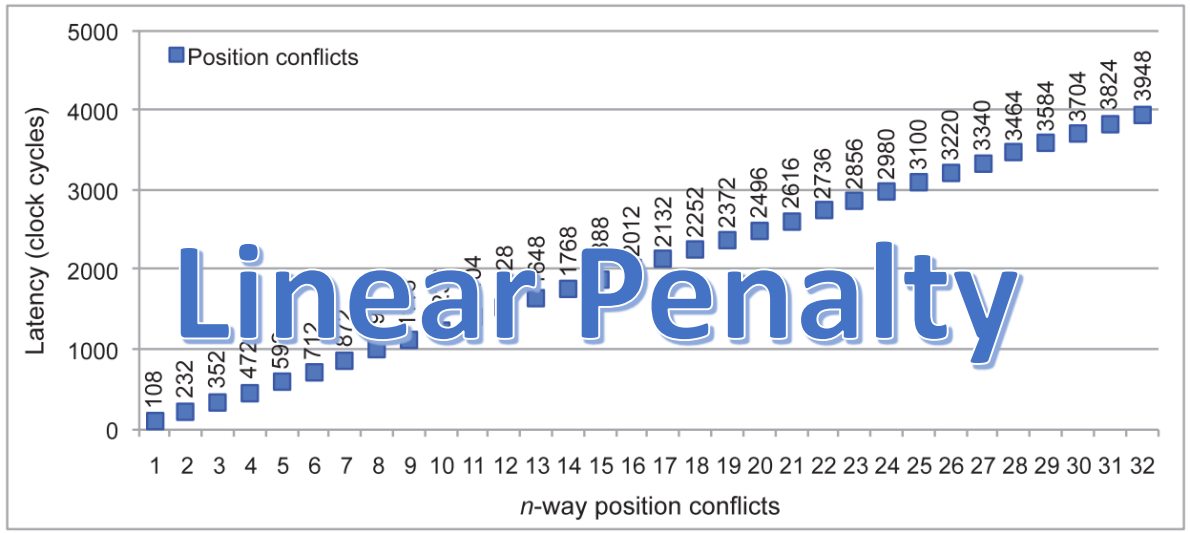
Gather in GPU Computation Kernel

- Gather() → Σ
- $\Sigma_1 \oplus \Sigma_2 \rightarrow \Sigma_3$



Problems in Gather

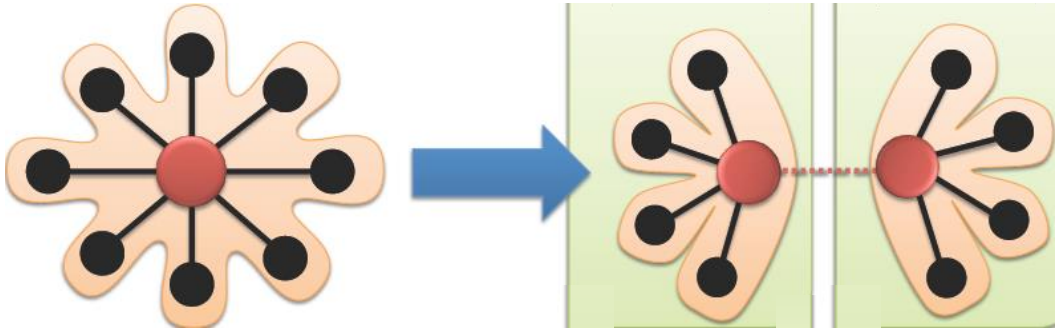




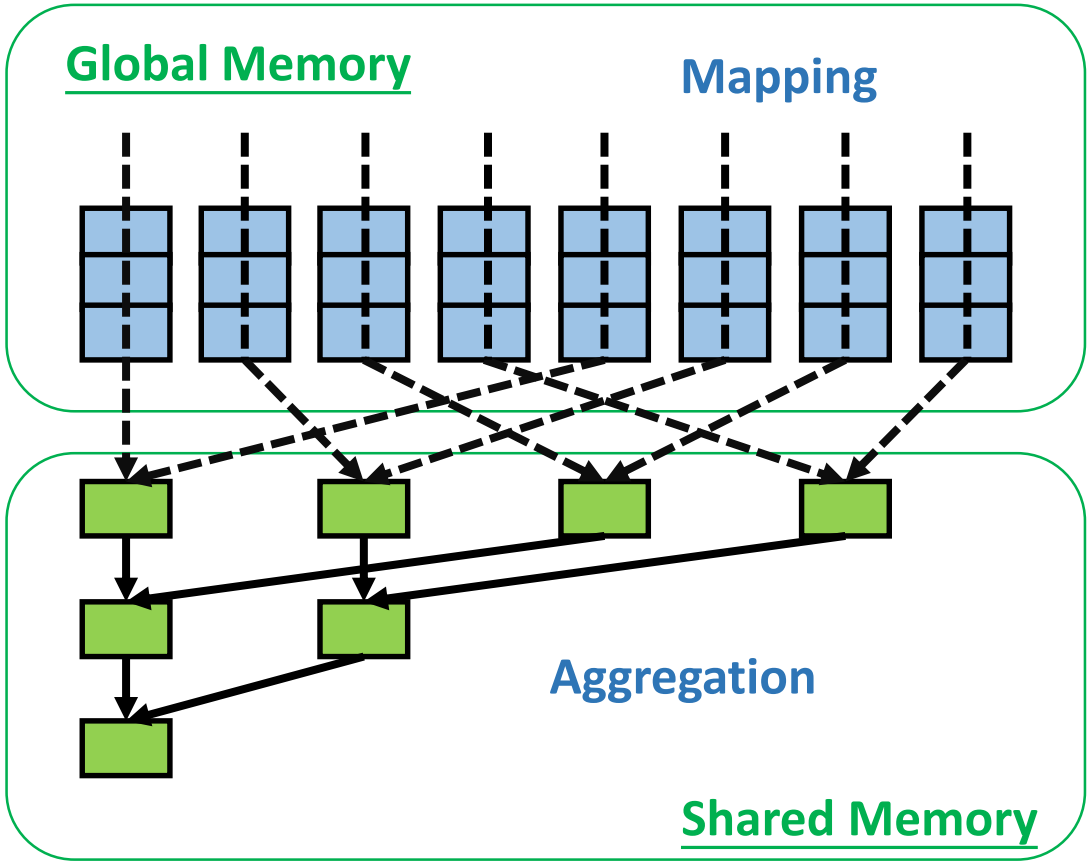
*Gomez-Luna, Juan, et al. "Performance modeling of atomic additions on GPU scratchpad memory." *TPDS* 24.11 (2013): 2273-2282.

- Conflicts
  - Linear penalty
  - Intra-warp >> Inter-warp

Replication-Based Gather

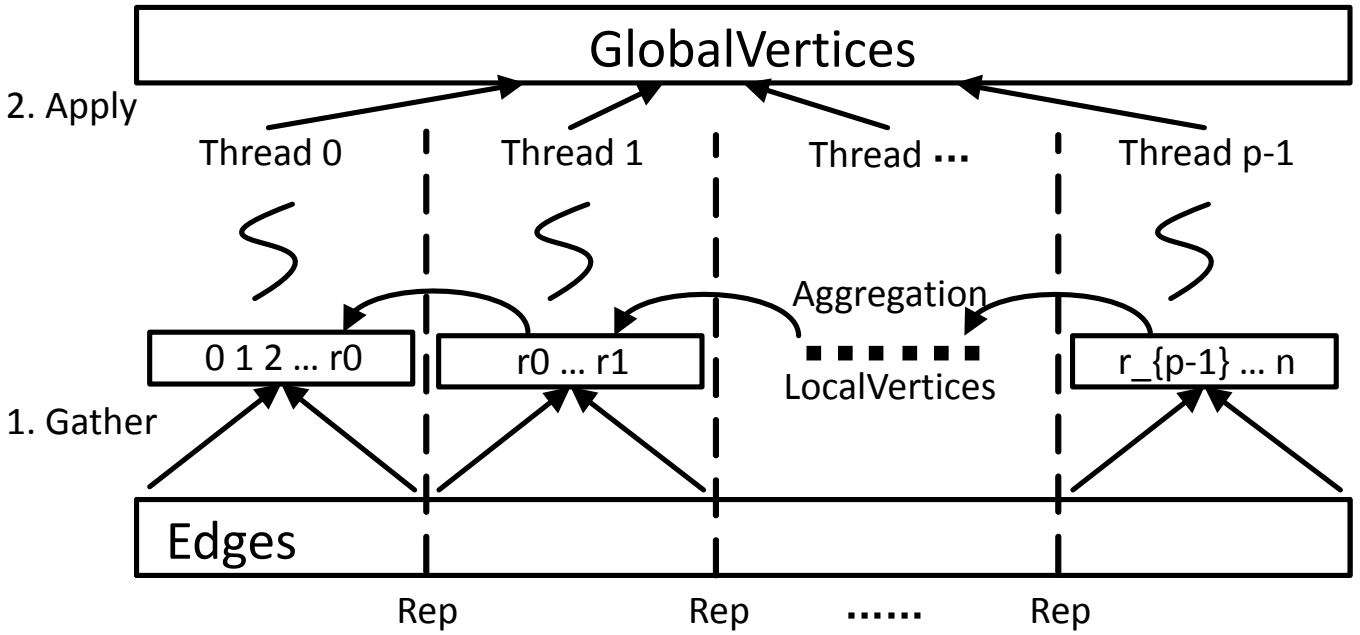


- Customized replication
  - O(N) -> O(logN), N≤32
  - Modeling: balance profits and costs
  - $R_i = 2^{\min\left\{\left\lceil \log \frac{|E_i|}{|V_i|} - 0.5 \right\rceil, 5\right\}}.$



CPU Computation Kernel

- Sequential memory access & lock-free & load balance

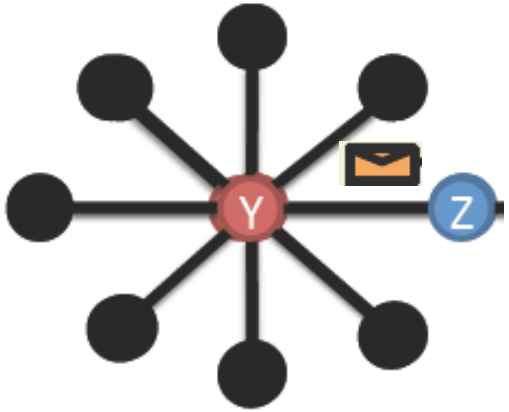


Dual Modes Processing Engine

- Pull & Notify-pull





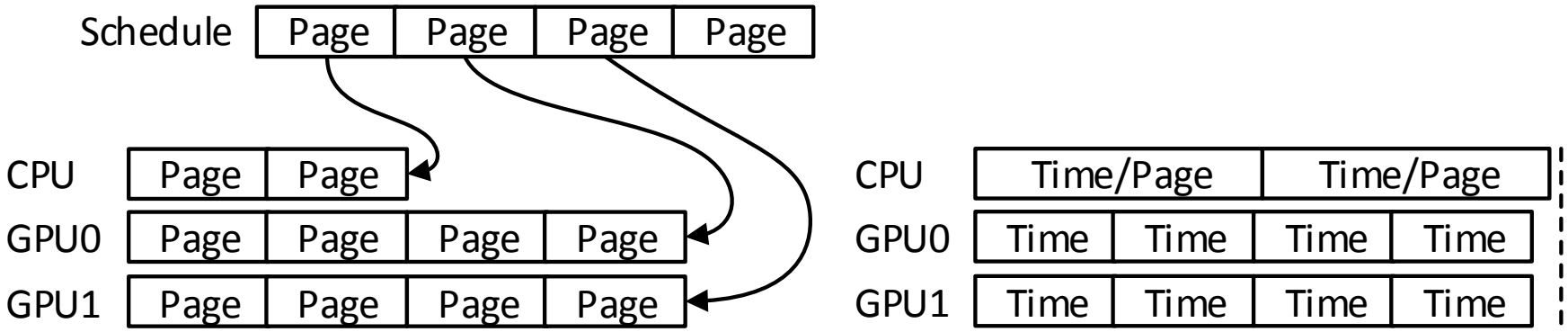








Hybrid CPU-GPU Scheduling



- CPU
  - Pros: thread sequential processing
  - Suit: pull/notify-pull dual-mode processing
- GPU
  - Pros: SIMD parallel processing
  - Suit: replication-based gather processing (only pull)

Experiment Setup

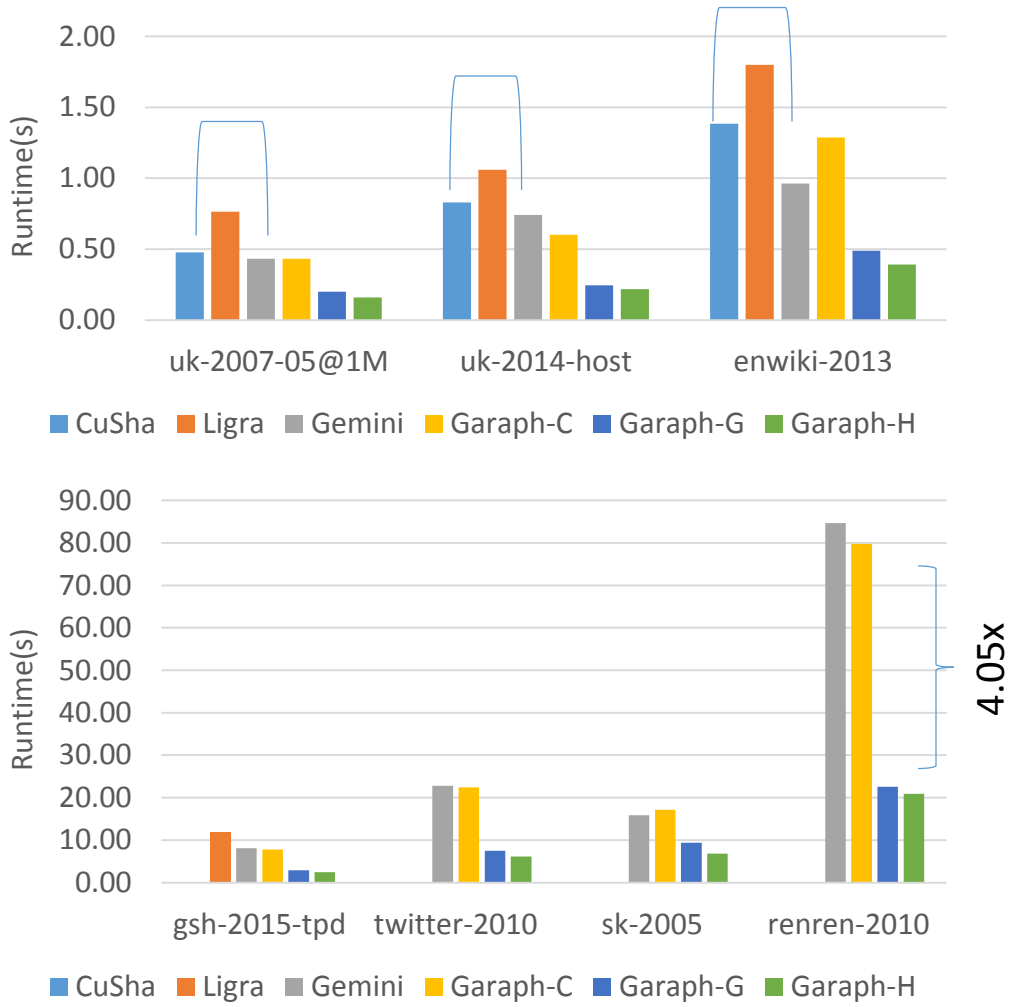
- Machine information
  - CPU: Intel Xeon E5-2650 v3
    - 10 cores, 20 threads, 2.3-3.0GHz
  - Memory: 64GB dual-channel DDR4 2133MHz
  - GPU: NVidia GeForce GTX 1070
    - 1920 cores, 15 SMs, 8GB memory
- Typical graph applications
  - PR, CC, SSSP, NN, HS, CS
- Datasets
  - 7 real world datasets
- Compare
  - CuSha(HPDC'14), Ligra(PPoPP'13), Gemini(OSDI'16)

Graph	|V|	|E|	Max in-deg	Avg deg	Size
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

Evaluation: Overall Performance

```
PageRank:
  gather(D_u, D_(u,v), D_v):
    return D_v.rank / D_v.outNbrs
  sum(Accum left, Accum right):
    return left + right
  apply(D_u, Accum):
    rnew = 0.15 + 0.85 * Accum
    D_u^new.rank = rnew
```

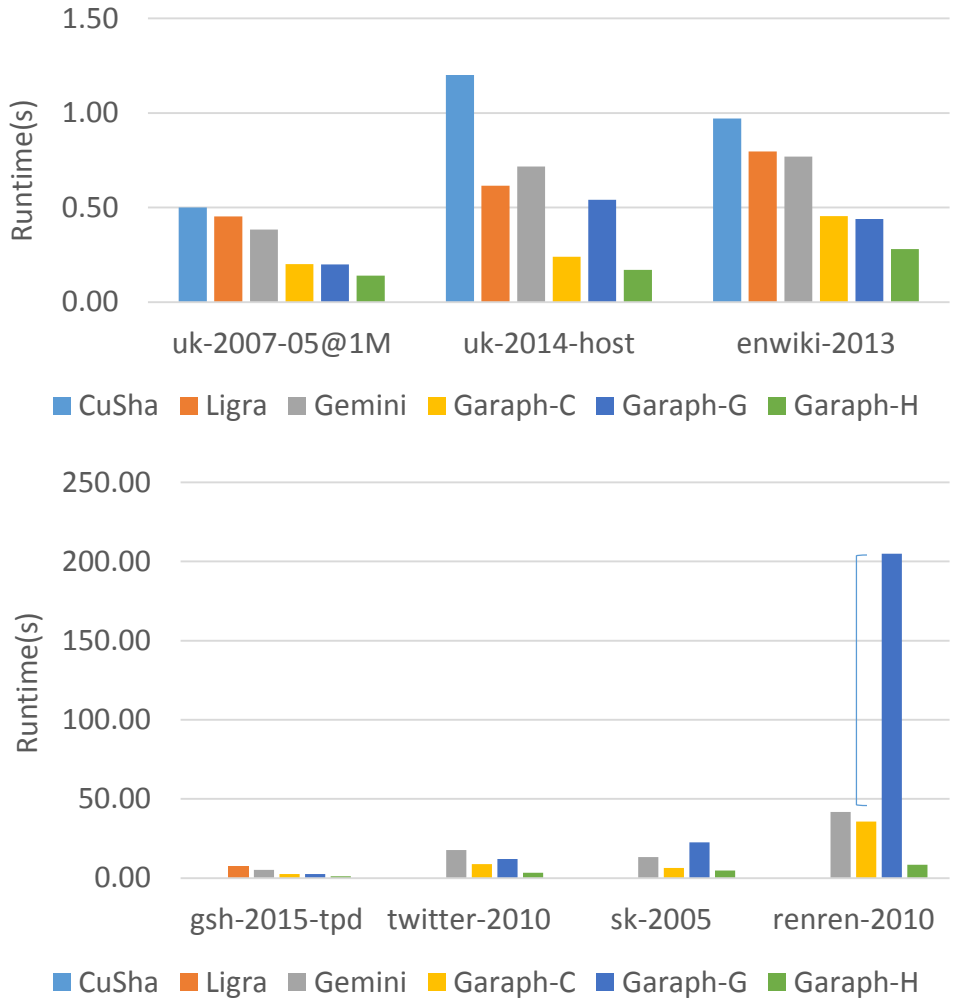
- Run 10 iterations of PR
- Up to 4.05x faster than the fastest one



Evaluation: Overall Performance

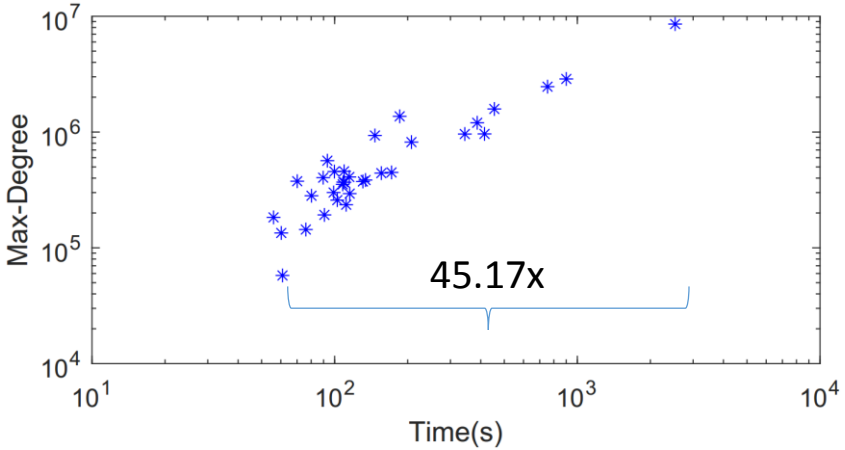
```
ConnectedComponent:
  gather(D_u, D_(u,v), D_v):
    return min(D_u.id, D_v.id)
  sum(Accum left, Accum right):
    return min(left, right)
  apply(D_u, Accum):
    D_u^new.id = Accum
  activate(D_u^new, D_u):
    if(D_u^new.id != D_u.id)
      return true
```

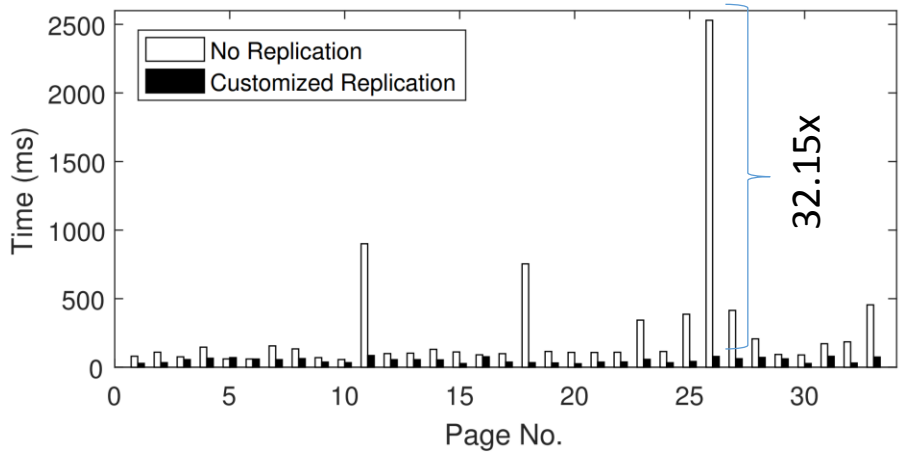
- Run CC to convergence
- GPU is much slower than CPU without activation scheme
- Up to 5.36x faster than the fastest one

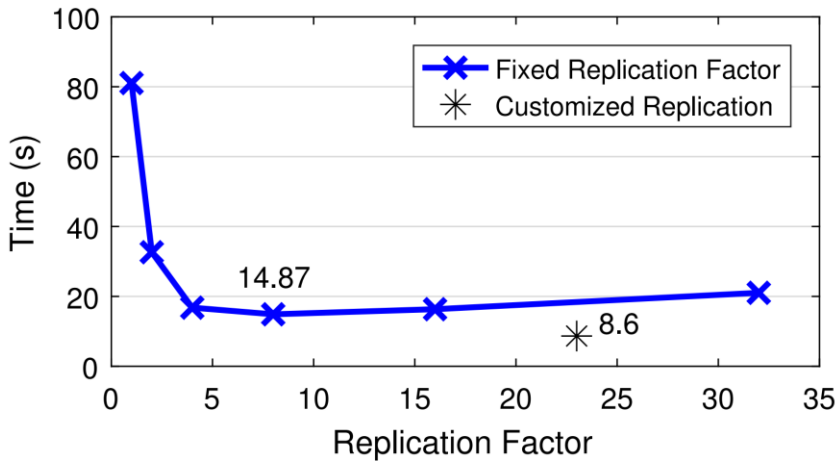


Evaluation: Customized Replication

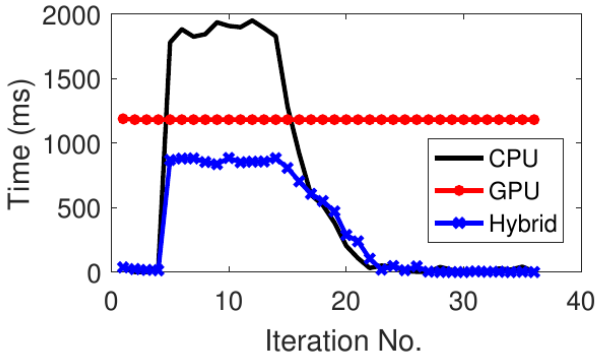
- SK-2005 dataset
  - Slowest is 45.17x slower than the fastest one
- Correlation: 0.9853
  - => vertices of high degree
- Customized replication
  - Up to 32.15x speedup



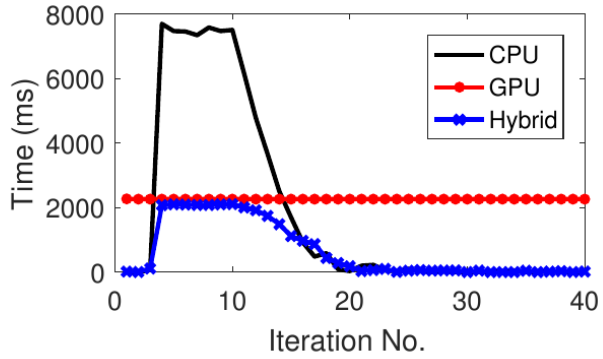




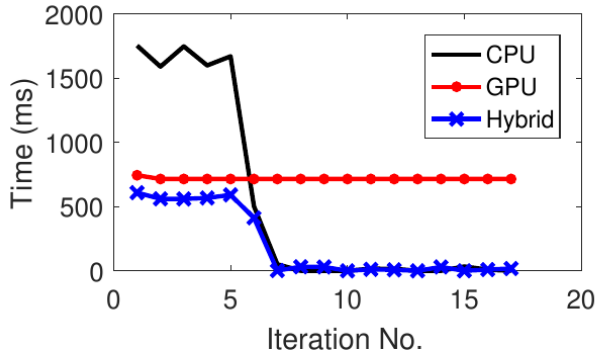
Evaluation: Hybrid CPU-GPU Scheduling



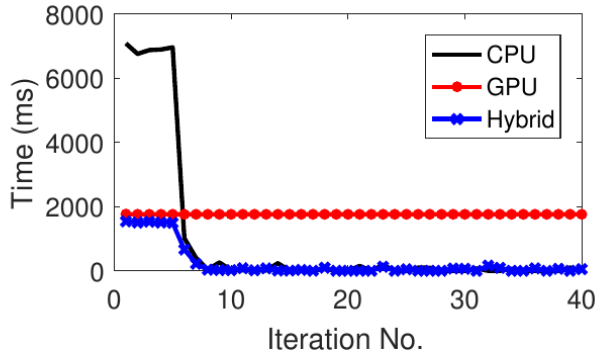
(a) SSSP: twitter-2010



(b) SSSP: renren-2010



(c) CC: twitter-2010



(d) CC: renren-2010

Conclusions

Garaph: efficient GPU-accelerated graph processing on a single machine

- Replication-based GPU computation kernel.
- Dual modes replication-based CPU computation kernel.
- Scheduler for hybrid CPU and GPU.

Q & A

Thanks!