Update on Triangle Counting on GPU

Carl Pearson [1], Mohammad Almasri [1], Omer Anjum [1], Vikram Mailthody [1], Zaid Qureshi [2], Rakesh Nagi [3], Jinjun Xiong [4], Wen-mei Hwu [1]

[1] Electrical and Computer Engineering / [2] Computer Science / [3] Industrial and Systems Engineering, University of Illinois Urbana-Champaign, Urbana, IL

[4] IBM T.J. Watson Research, Yorktown Heights, NY

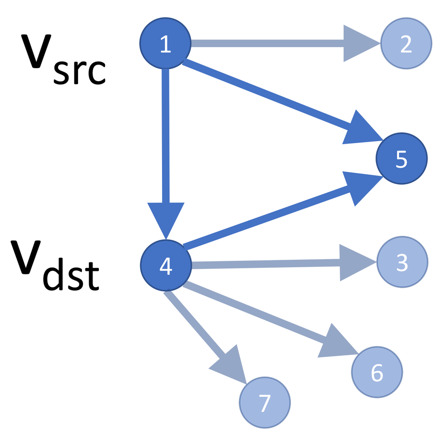
Introduction

This poster prevents the current status of several techniques being studied to achieve high-performance triangle counting, including

- persistent-block dynamic algorithm selection kernel designed to retain high performance on a variety of graph structures without an analysis phase
- Foundational techniques for multi-GPU scaling on unified memory system
- Initial design of a task-partitioning strategy for triangle counting to improve multi-GPU scaling

We are applying lessons learned from this work in more complicated graph analytics problems.

Triangle Counting

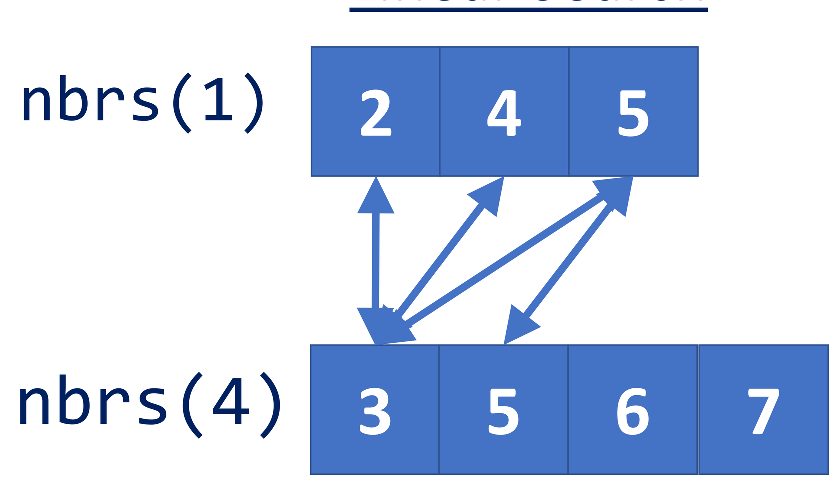


count triangles for directed edge $v_{src} \rightarrow v_{dst}$

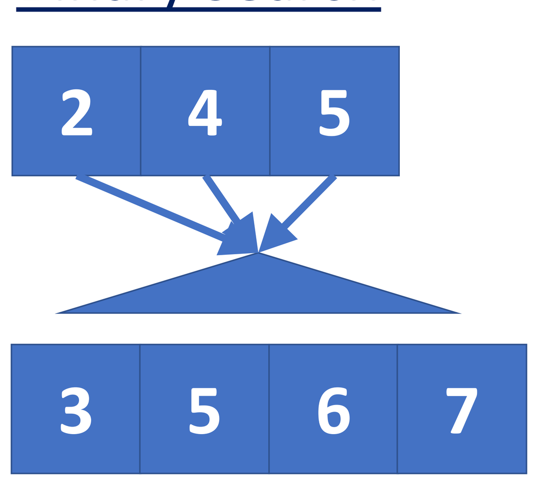
```
let nbrs(v) -> { neighbors of vertex v }
let len(s)  -> size of set s
count = 0
for all v_dst in nbrs(v_src):
  count += len( nbrs(v_src) ∩ nbrs(v_dst) )
```

Triangles for each directed edge are the number of common neighbors the source and destination vertex share. Graph is stored in CSR (compressed sparse row) format, with sorted rows. **Key triangle counting operation is a sorted set intersection.**

Linear Search



Binary Search



linear search (left), a single position is compared in each list, and the lower position is advanced. When values match, the count is incremented and both are increased. In binary search (right), a binary search through the longer list is execute in parallel for each element in the shorter list.

Persistent-Warp Kernel

Persistent-warp kernel fills the GPUs with long-lived warps. Warps use atomics to claim groups of 32 edges, estimate fastest algorithm, and use that algorithm to count triangles for all 32 edges.

a ∩ b	Expected Global Memory Accesses
Linear	len(a) + len(b)
Binary	len(b) × $\log_2$(len(a))

The estimated binary search cost is multiplied by a scaling factor *sf* to account for dynamic cost of each algorithm. Low *sf* favors binary search, high *sf* linear search. Based on empirical evaluation of large graphs, *sf = 2* is chosen for this platform.

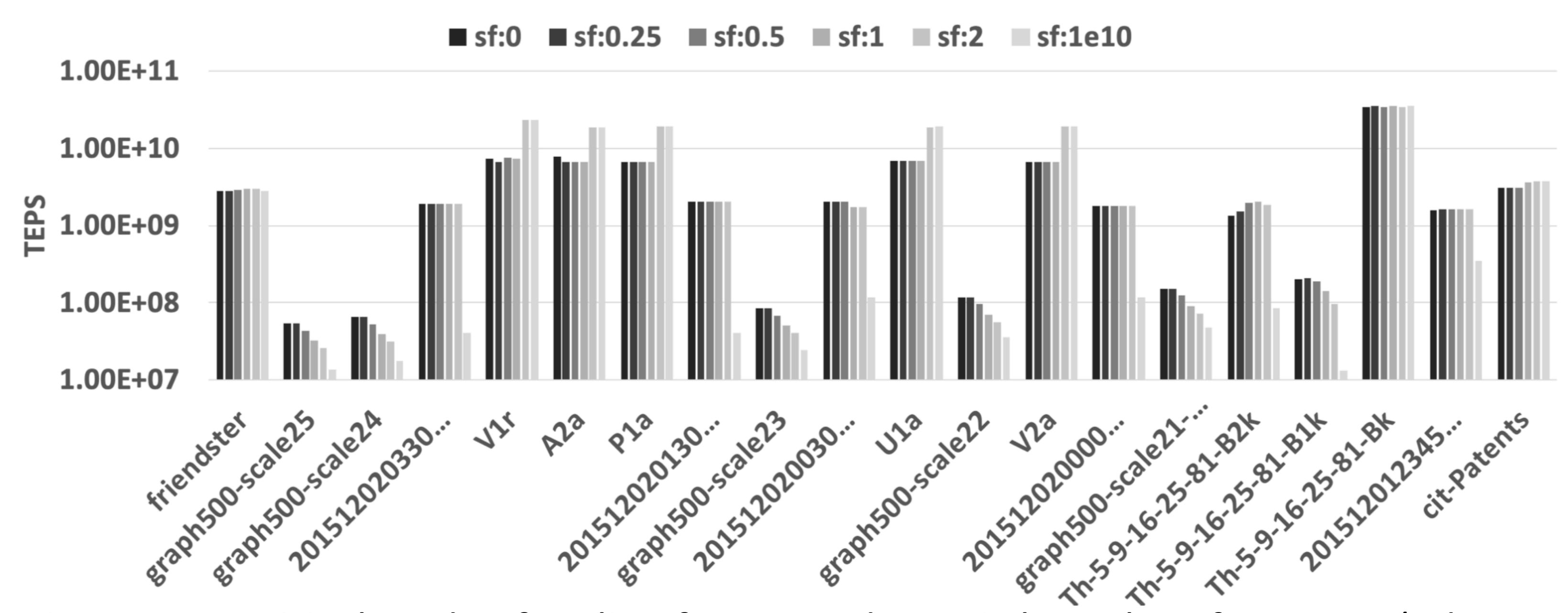


Figure 1: Empirical study of scaling factor on dynamic kernel performance (edges per second).

Ultimately, the dynamic algorithm is typically able to achieve performance of the best of either the linear or binary algorithm.

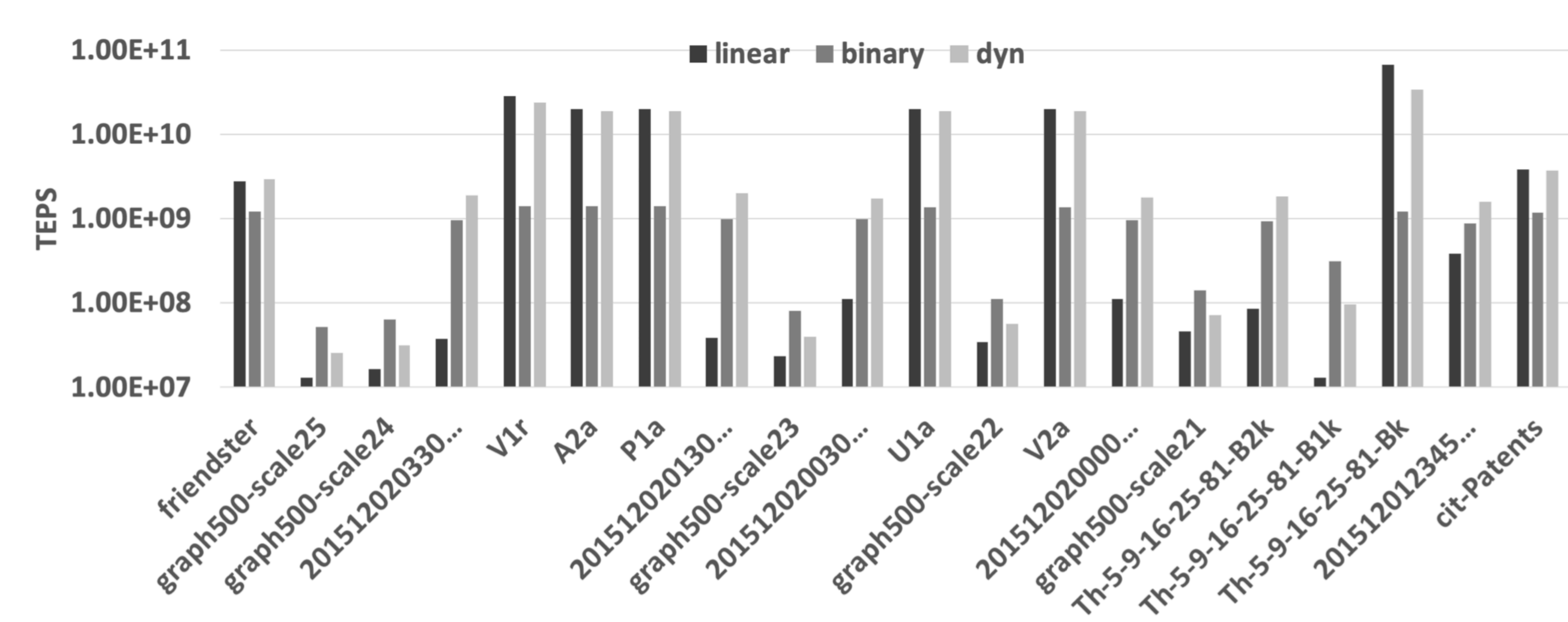
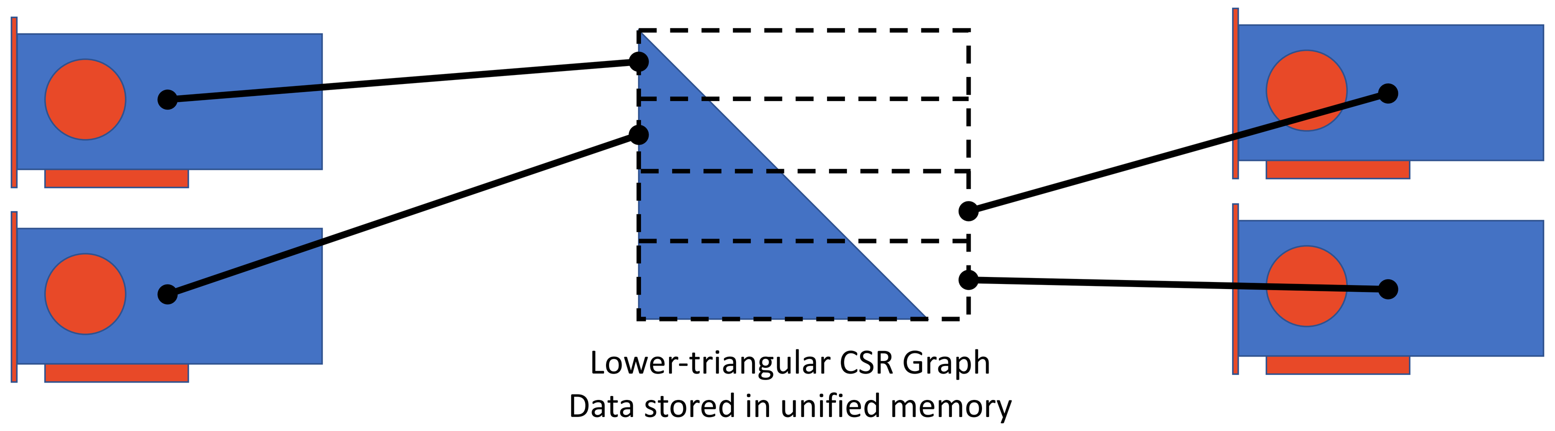


Figure 2: Comparison of performance between linear search, binary search, and dynamic search kernel. The dynamic kernel uses *sf=2.*

Multi-GPU Unified Memory



The CSR data is stored in unified memory, which provides a coherent memory image to all devices. Base edges are split evenly among GPUs, leading to substantial load imbalance and access overlap. Data is marked cudaMemAdviseReadMostly to prevent thrashing for multiple devices.

Future 2D Partitioning

N2 partitions and N3 tasks. Task {i,j,k} counts a subsect triangles for edges between vertices in row partitions i and j (partition i,j). Non-zeros from partitions I,k and j,k compared for partial triangle count.

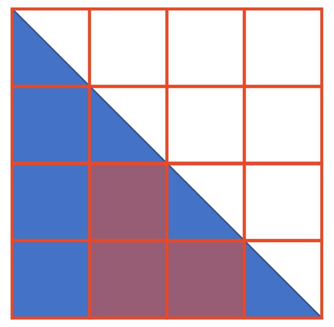
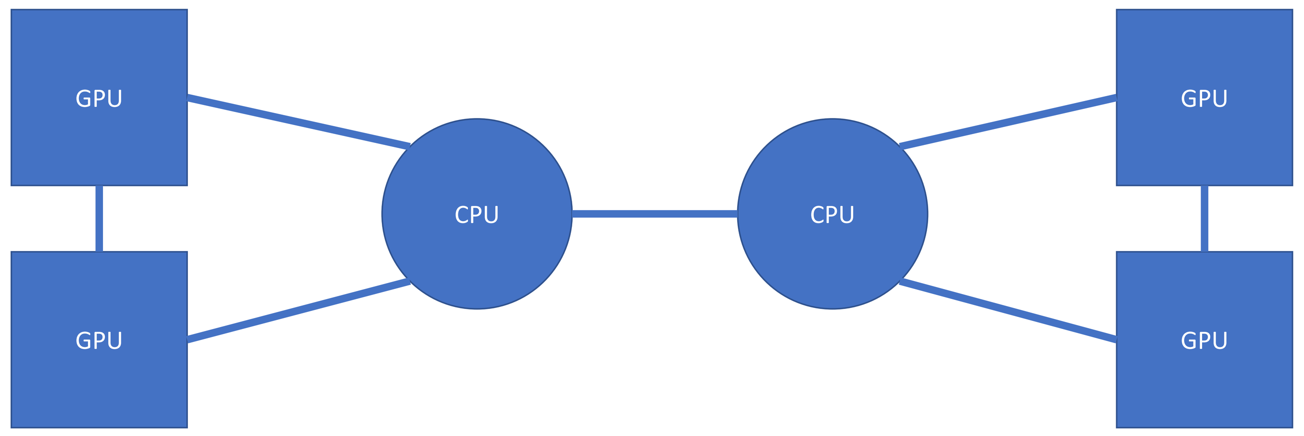


Figure 3: TASK (2, 2, 1}. PARTITION 3,2 HOLDS THE EDGES THAT START IN ROW PARTITION 3 AND END IN 2. PARITIONS 3,1 AND 2,1 HOLD THE PORTIONS OF THE ROWS FOR WHICH THE INTERSECTION WILL BE COMPUTED.

Partitions can be sized to fit in the local memory of a GPU even for large graphs. All N3 tasks can be executed independently.

IBM AC-922 Evaluation System



Two IBM POWER 9 CPUs, 512 GB RAM, 4 Nvidia V100 (16GB) GPUs, NVLink 2.0 150 GB/s triad interconnect, 64 GB/s CPU-CPU interconnect.

External Resources

Pangolin Graph library: github.com/c3sr/pangolin

Dataset Management: github.com/cwpearson/graph-datasets2

C3SR center for cognitive computing systems research

Acknowledgements: This work is supported by IBM-ILLINOIS Center for Cognitive Computing Systems Research (C3SR) - a research collaboration as part of the IBM AI Horizons Network. This work utilizes resources supported by the National Science Foundation's Major Research Instrumentation program, grant #1725729, and University of Illinois at Urbana-Champaign

ECE ILLINOIS