Grazelle

Hardware-Optimized In-Memory Graph Processing

Samuel Grossman, Heiner Litz, and Christos Kozyrakis

Existing Work

Properties of Graph Problems

- Irregular graph data
- Difficult to partition
- Unpredictable access pattern

Scalability Optimizations

- Partitioning algorithms
- Dynamic scheduling, load balancing
- Sharing and synchronization optimizations

Existing Work

Properties of Graph Problems

- Irregular graph data
- Difficult to partition
- Unpredictable access pattern

Modern Hardware Features

- ✗ Vector processing units
- ✗ Sequential memory accesses
- ✗ Prefetchers
- ✗ NUMA

Grazelle

Properties of Graph Problems

- Irregular graph data
- ✓ **Simple and easy** to partition
- ✓ **Predictable** access pattern

Modern Hardware Features

- ✓ Vector processing units
- ✓ Sequential memory accesses
- ✓ Prefetchers
- ✓ NUMA

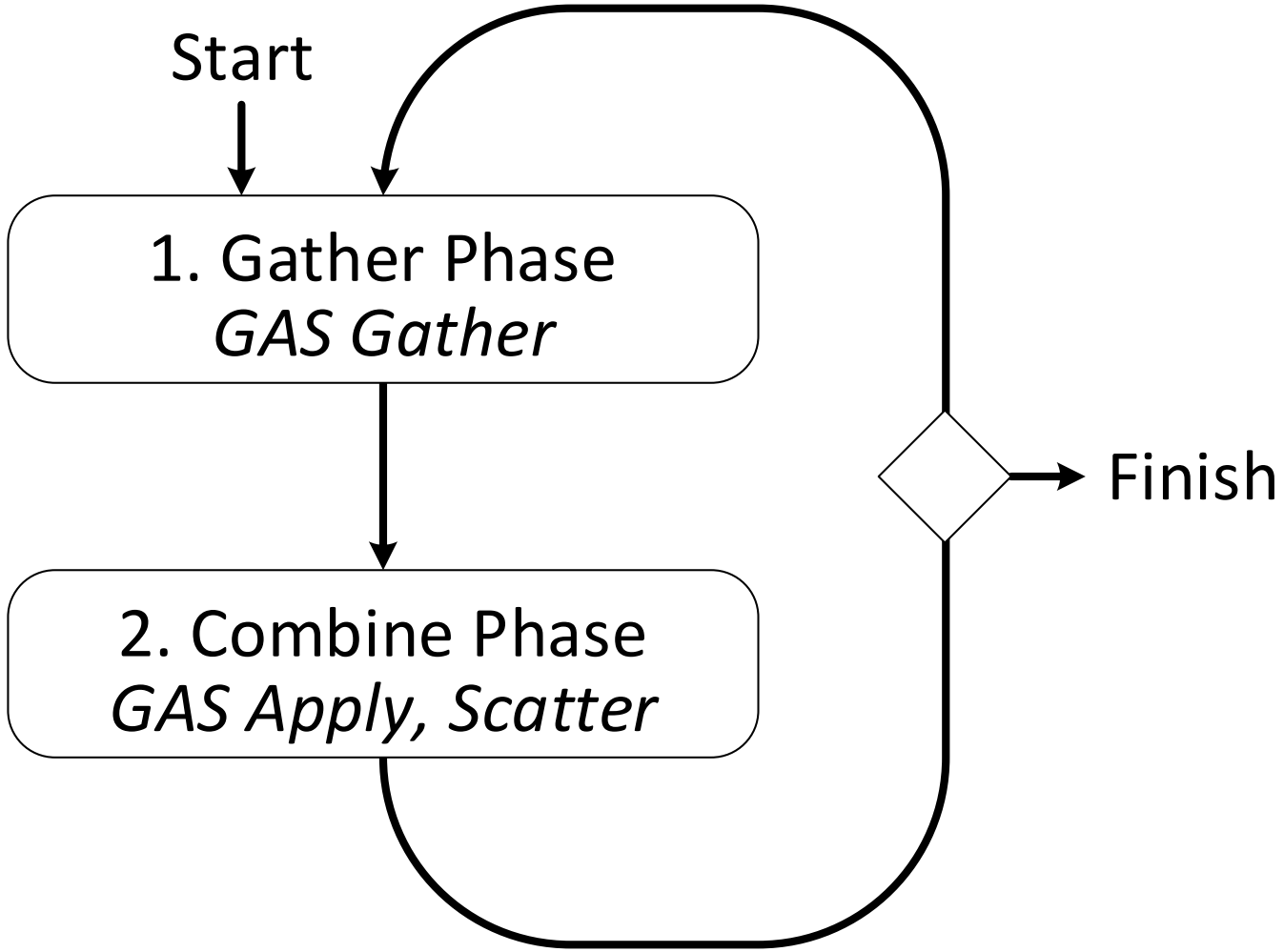
Grazelle

Grazelle is a **single-machine**, **in-memory** Gather-Apply-Scatter (GAS) graph processing engine that:

- Leverages modern hardware features
- Improves throughput by 4.4× to 36.2× over existing work

Grazelle is **not** a complete graph analytics framework.

Top-Level Execution Flow



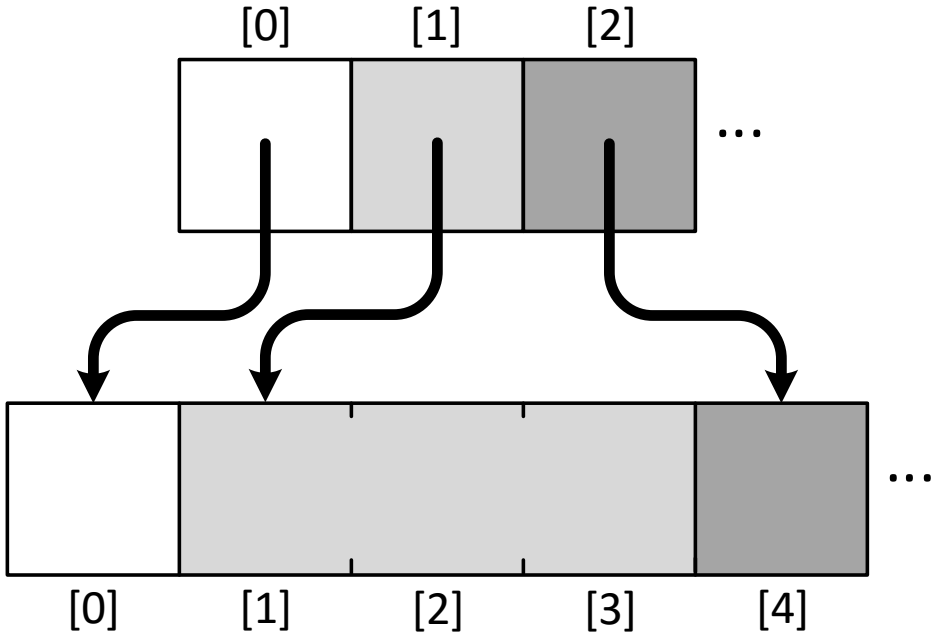
Key Design Principles

- Vector-optimized data structures with minimal indirection
- Thread-private memory writes
- Mostly sequential memory accesses
- Simple, static partitioning and scheduling
- Synchronization via thread barriers between phases

Gather: Topology Data Structures

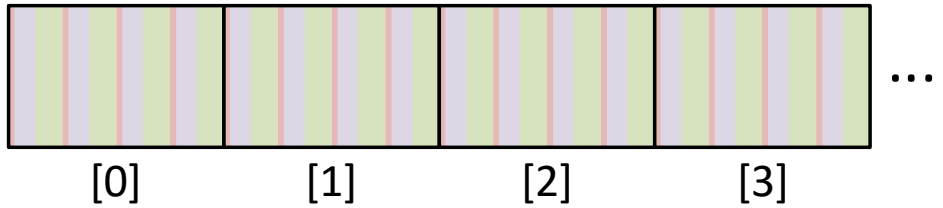
Existing Work

- "Compressed Sparse Row"

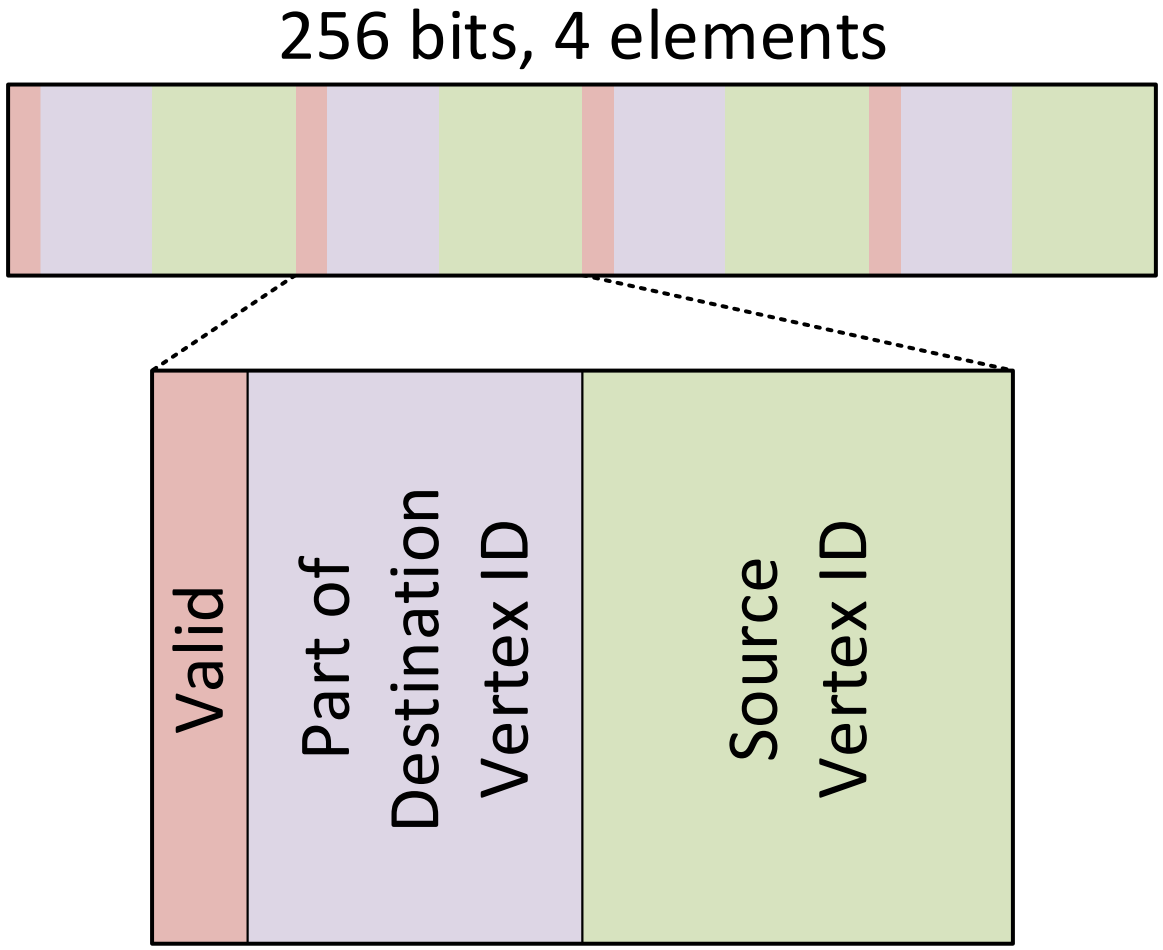


Grazelle

- Vector-encoded edge list

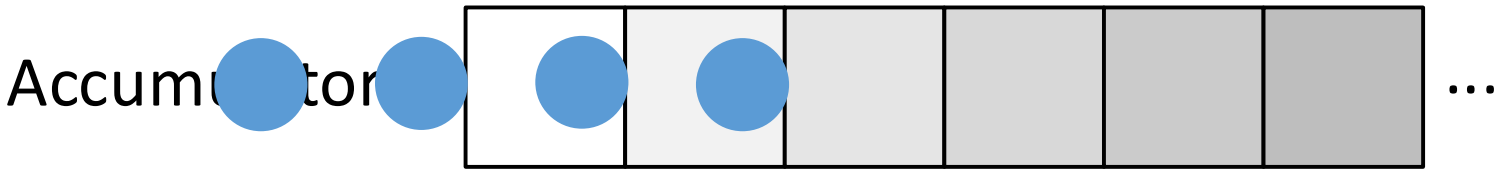


Gather: Topology Data Structures



Gather: Execution

Combine: Execution

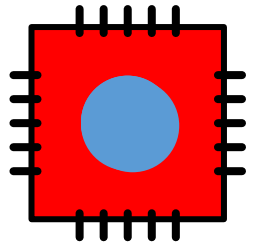


✓ Private, read-only

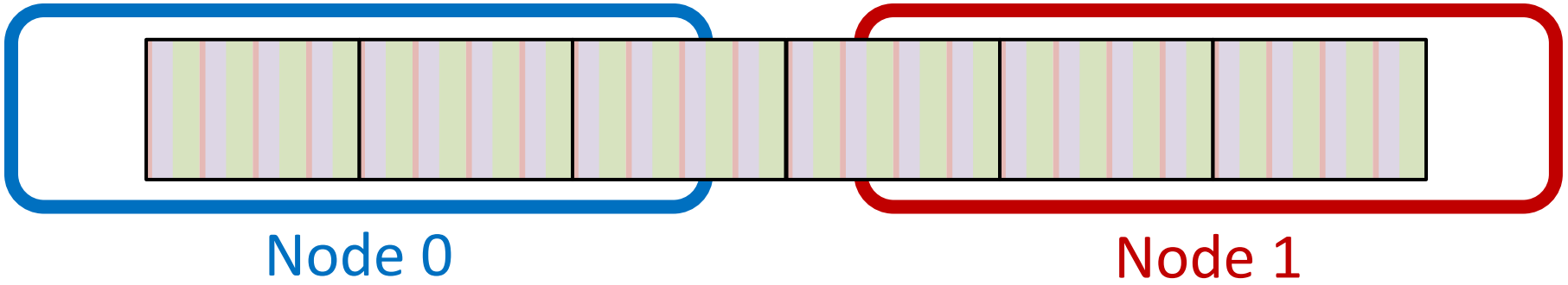
vector Load

✓ Private, write-only

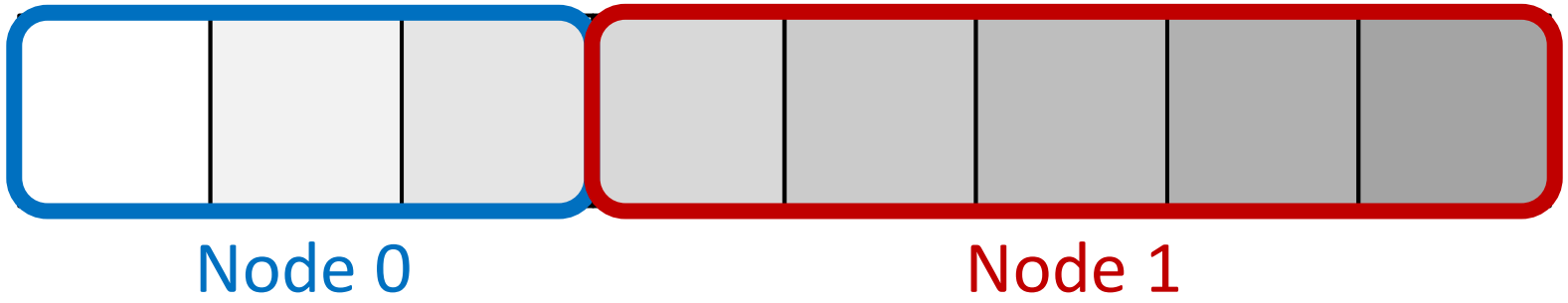
vector Store



NUMA Partitioning



Edges always NUMA-local



Accumulators always NUMA-local

Vertices sometimes NUMA-remote

Evaluation

Processor:	4× Intel Xeon E7-4850 (14 cores, 2-way SMT, 35 MB LLC)
RAM:	1 TB total, 256 GB per socket
Storage:	12× 6 TB magnetic disks, RAID-10
OS:	Ubuntu 14.04 LTS
Compiler:	GCC 4.8

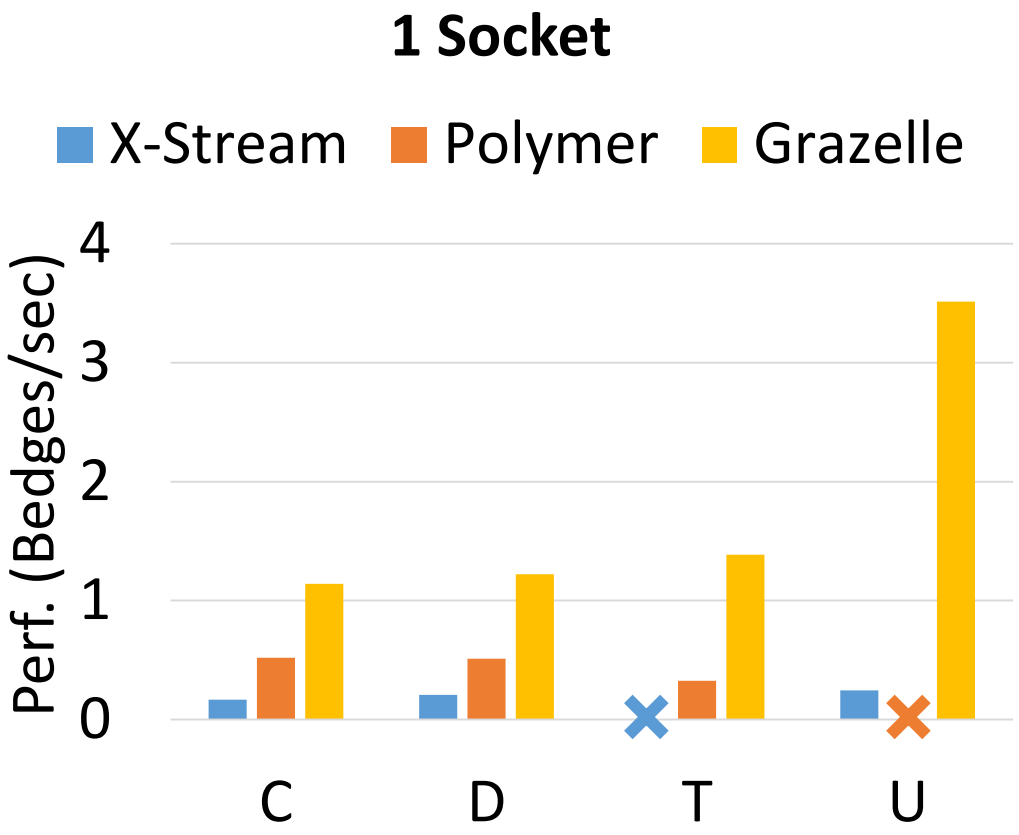
Evaluation

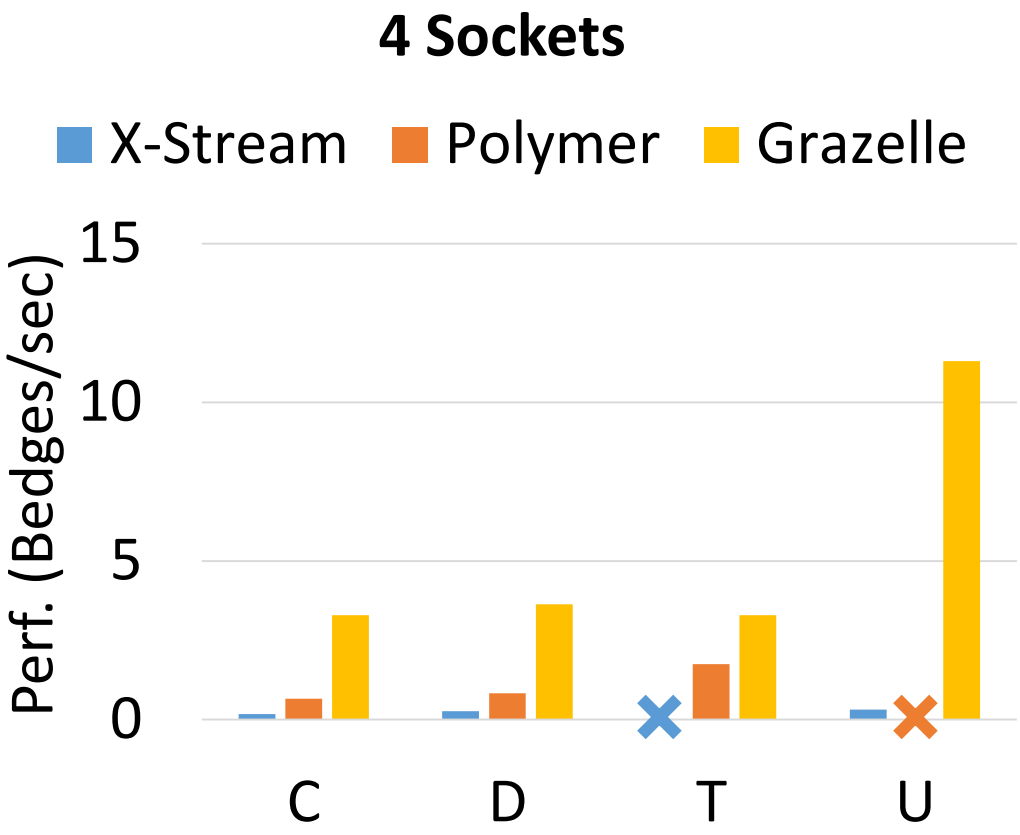
Name	Abbreviation	Vertices	Edges	Size	Domain
cit-Patents	**C**	3.7 M	16.5 M	250 MB	Citations web
dimacs-usa	**D**	23.9 M	58.3 M	900 MB	Road network
twitter-2010	**T**	41.7 M	1.47 B	20 GB	Social
uk-2007	**U**	105.9 M	3.74 B	60 GB	Internet
(skewed synthetic)		≤ 134 M	≤ 17 B	≤ 250 GB	

Comparison

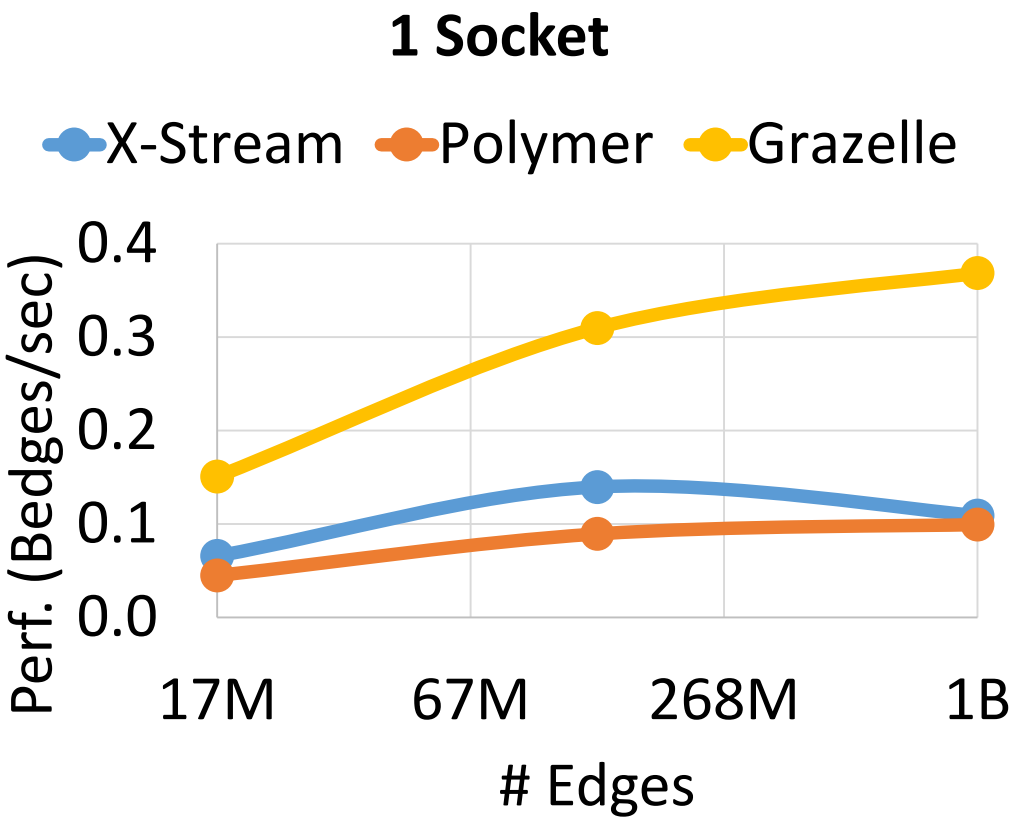
	X-Stream	Polymer	Grazelle
Vector processing units	No	No	Yes
Sequential memory accesses	Yes	Yes	Yes
Prefetching	No	No	Yes
NUMA awareness	No	Yes	Yes
Caching overheads	Yes	Partial	Yes
Simultaneous multithreading	No	No	Yes

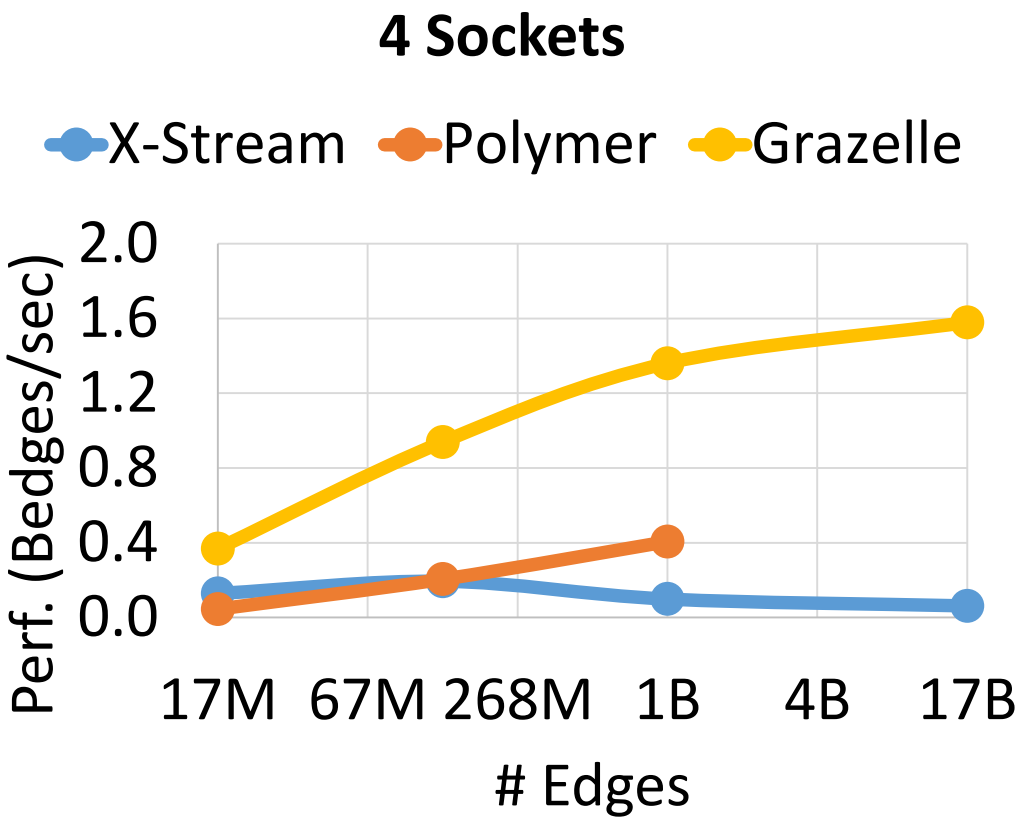
Comparison: Throughput (Real Graphs)



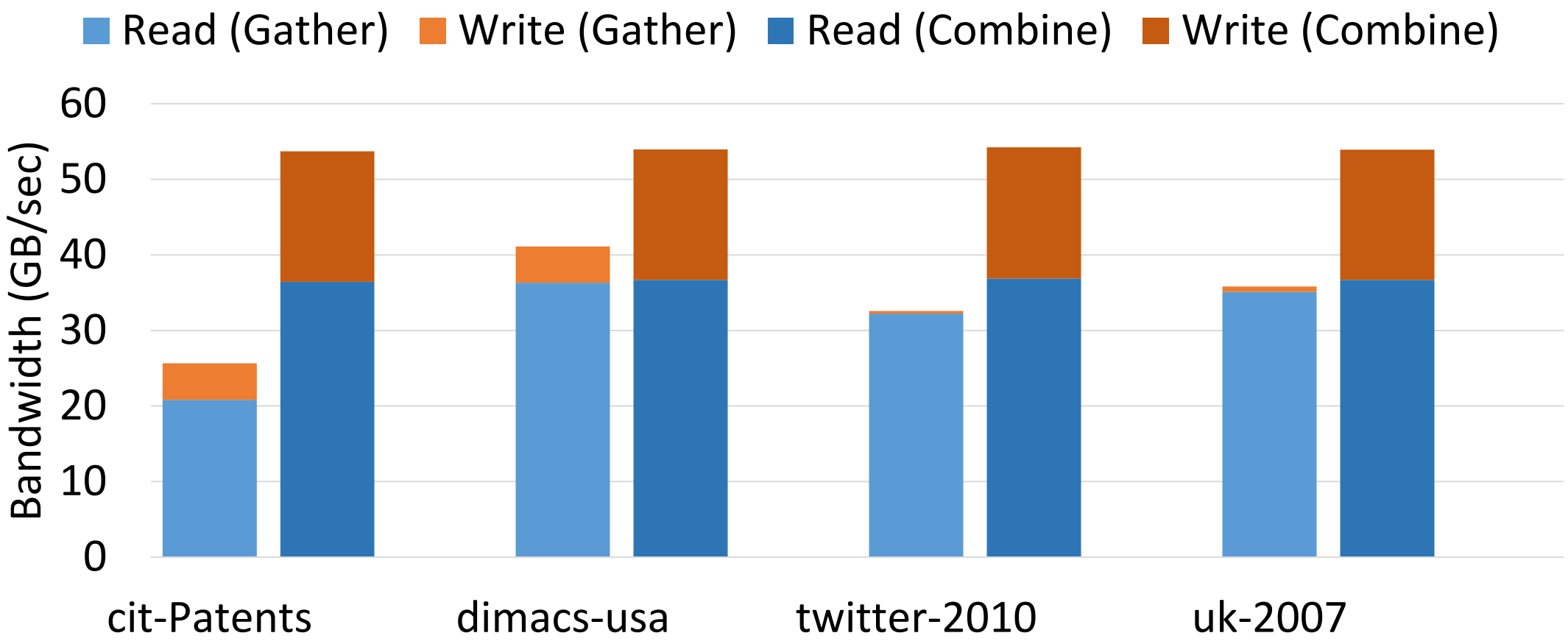


Comparison: Throughput (Synthetic Graphs)

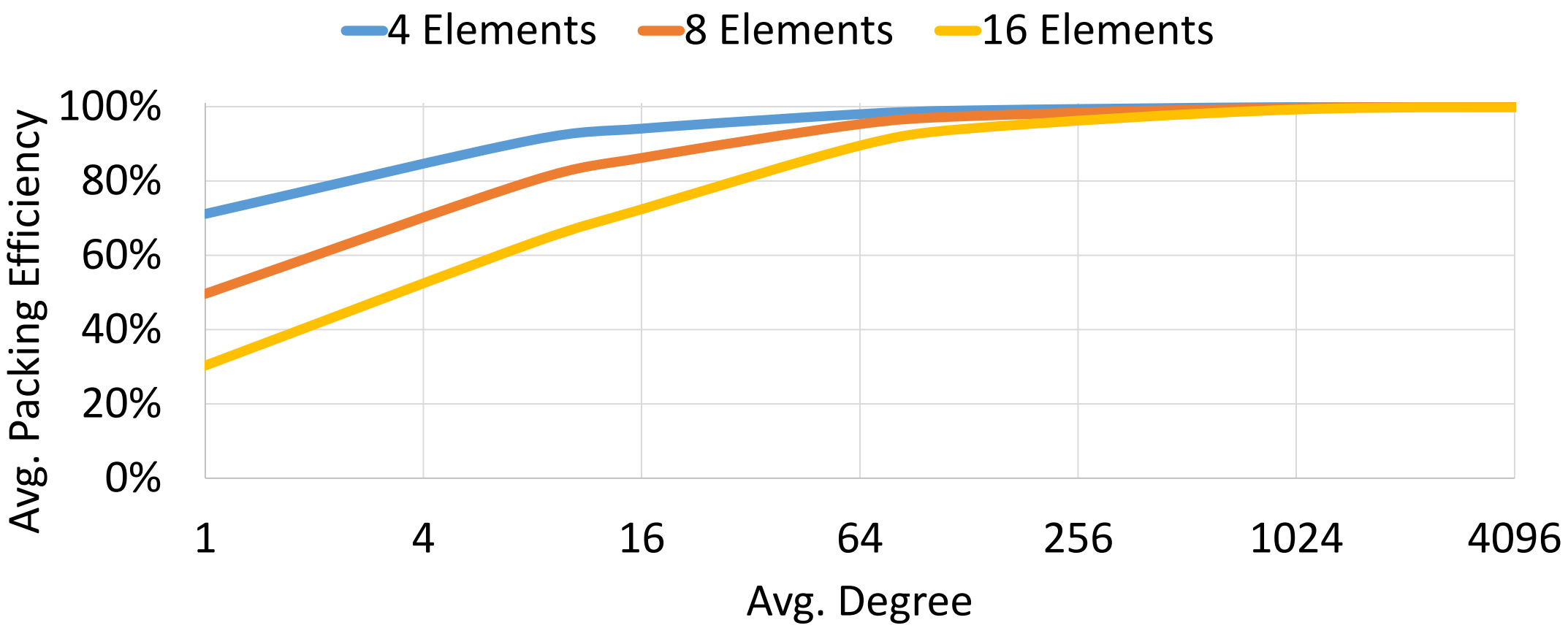




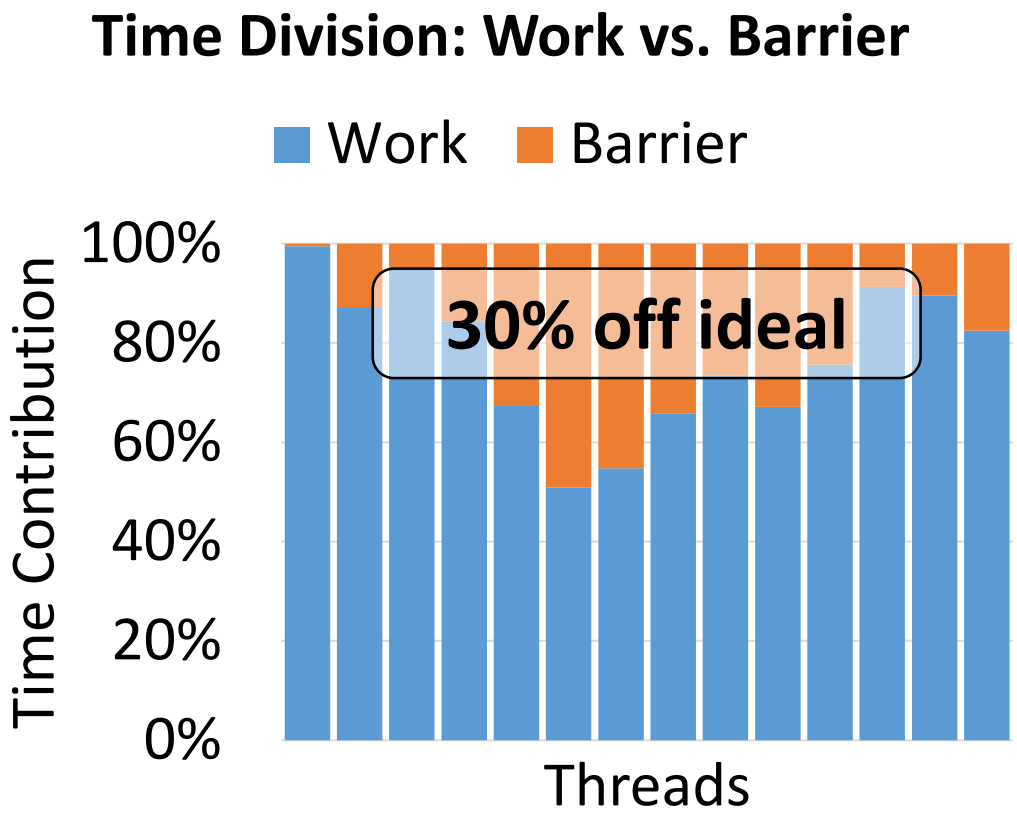
Memory Bandwidth Utilization

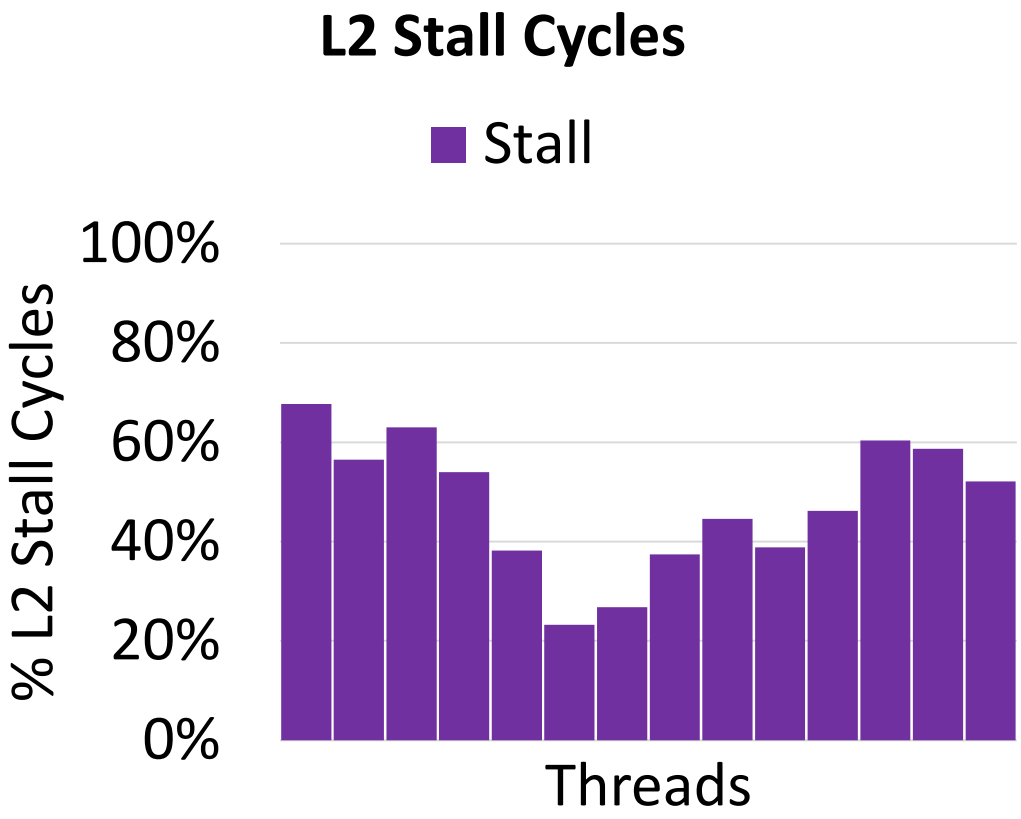


Edge Vector Packing Efficiency



Load Balance Effectiveness





Conclusion

- Grazelle maps graph problems to a *regular* and *predictable* software implementation without sacrificing scalability or balance

- Grazelle effectively leverages modern hardware and significantly outperforms the state-of-the-art

- Future work:
  - Expand to secondary storage devices like flash
  - Build higher-level optimizations on top of Grazelle