Performance evaluation of scale-free graph algorithms in low latency Non-Volatile Memory

Manu Shantharam, Pietro Cicotti, Laura Carrington (SDSC)
Roger Pearce, Keita Iwabuchi, Maya Gokhale (LLNL)

HPBDC 2017
May 29, 2017

LLNL-PRES-732425

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

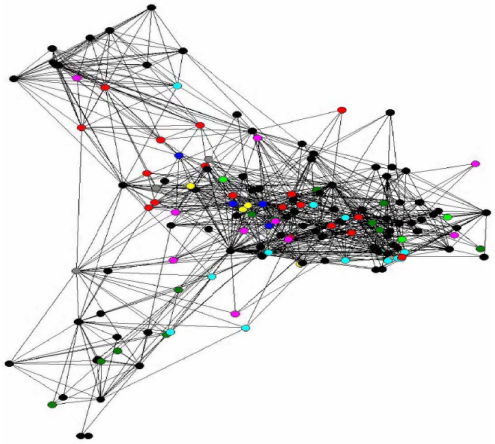
SDSC SAN DIEGO SUPERCOMPUTER CENTER

Lawrence Livermore National Laboratory

Process large graphs using advanced architectures

Facilitate the processing of Massive Real-World Graphs

- ~~Billions~~ Trillions of vertices and edges (e.g., Social Networks, Web Graphs)
- Potentially too large to fit in DRAM of HPC clusters



Utilize emerging Non-volatile memory storage technologies (e.g., NAND Flash)

- Develop techniques that tolerate data latencies

We have developed:

- Asynchronous vertex-centric computation model [SC'10]
- Communication optimizations: asynchronous message routing, aggregation & filtering [IPDPS'13]
- High-degree vertex partitioning & parallel processing [SC'14]
- 17+ Trillion edges using NVRAM storage [Graph500]

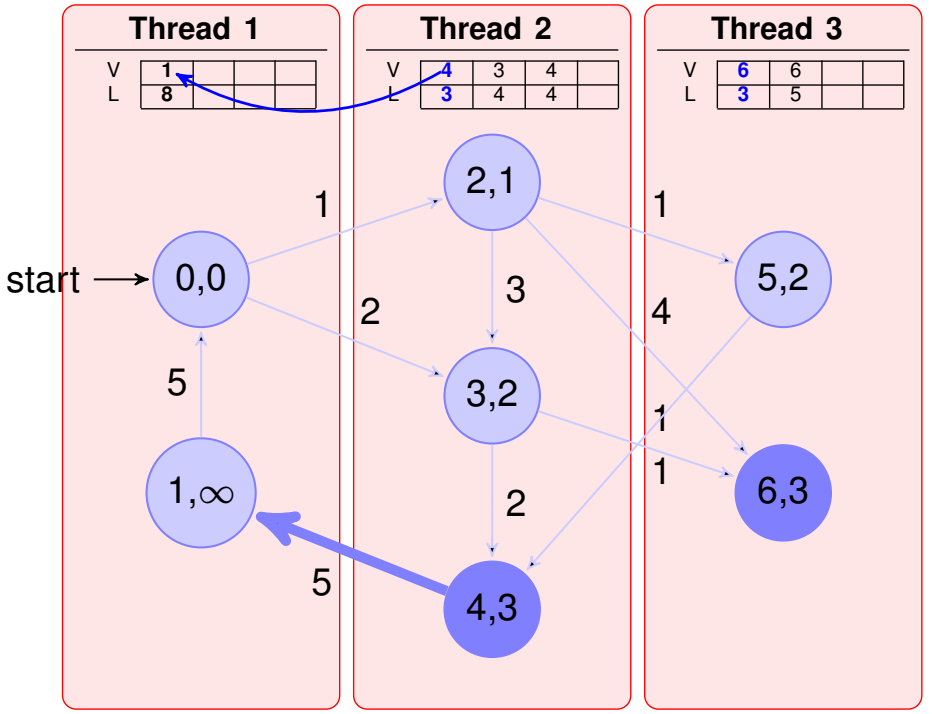
Target Platform:

- HPC + node-local NVRAM (Catalyst @ LLNL 128GB DRAM + 700GB NVRAM per node)
- **Future NVM technologies!**

Parallel graph traversal framework
Asynchronous Dataflow Algorithms

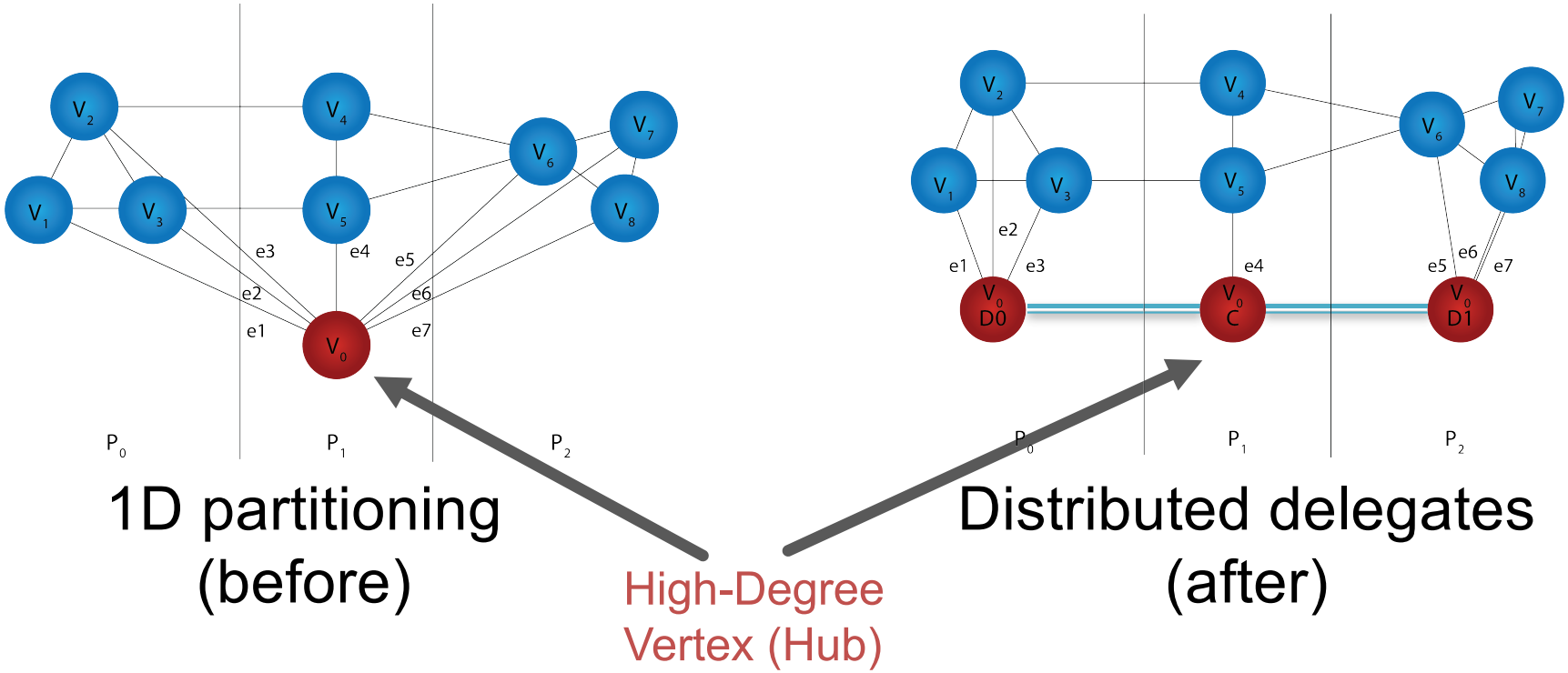
- Developed parallel, latency tolerant asynchronous traversal framework
  - Independent threads or processors schedule work on others' work queues
  - Large number of concurrent, independent I/O requests to NVRAM
  - Competing techniques are level-synchronous

- Uses visitor abstraction
  - Visitor is an application-specific kernel that the framework applies to each graph vertex
  - Visit results in traversal of graph edges to queue work on target vertices
  - Visitor is queued to the vertex using priority queue
  - Visitor execution scheduler orders to exploit page-level data reuse



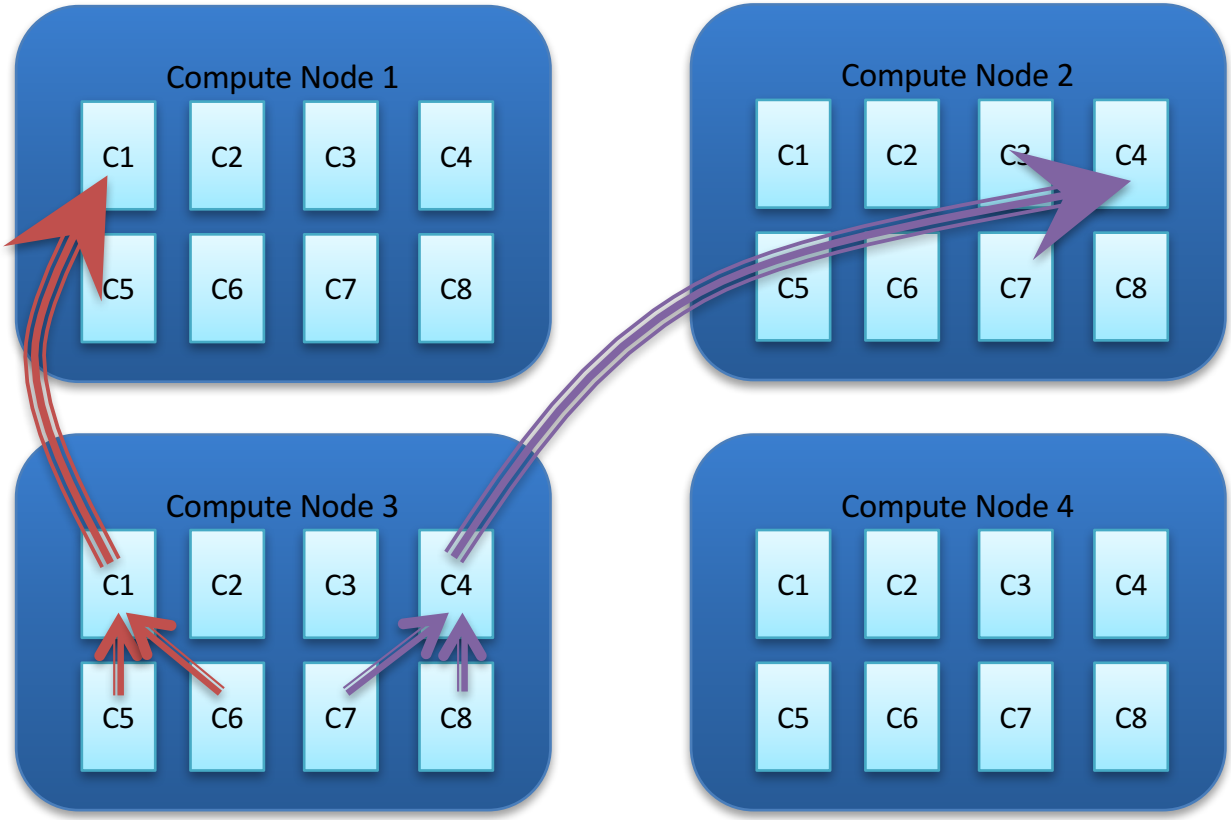
- Demonstrated with Breadth first search, Single source shortest path, (Strongly) Connected components, Triangle counting, K-th core, pagerank, coloring

Partitioning Optimizations for Scale-Free Graphs



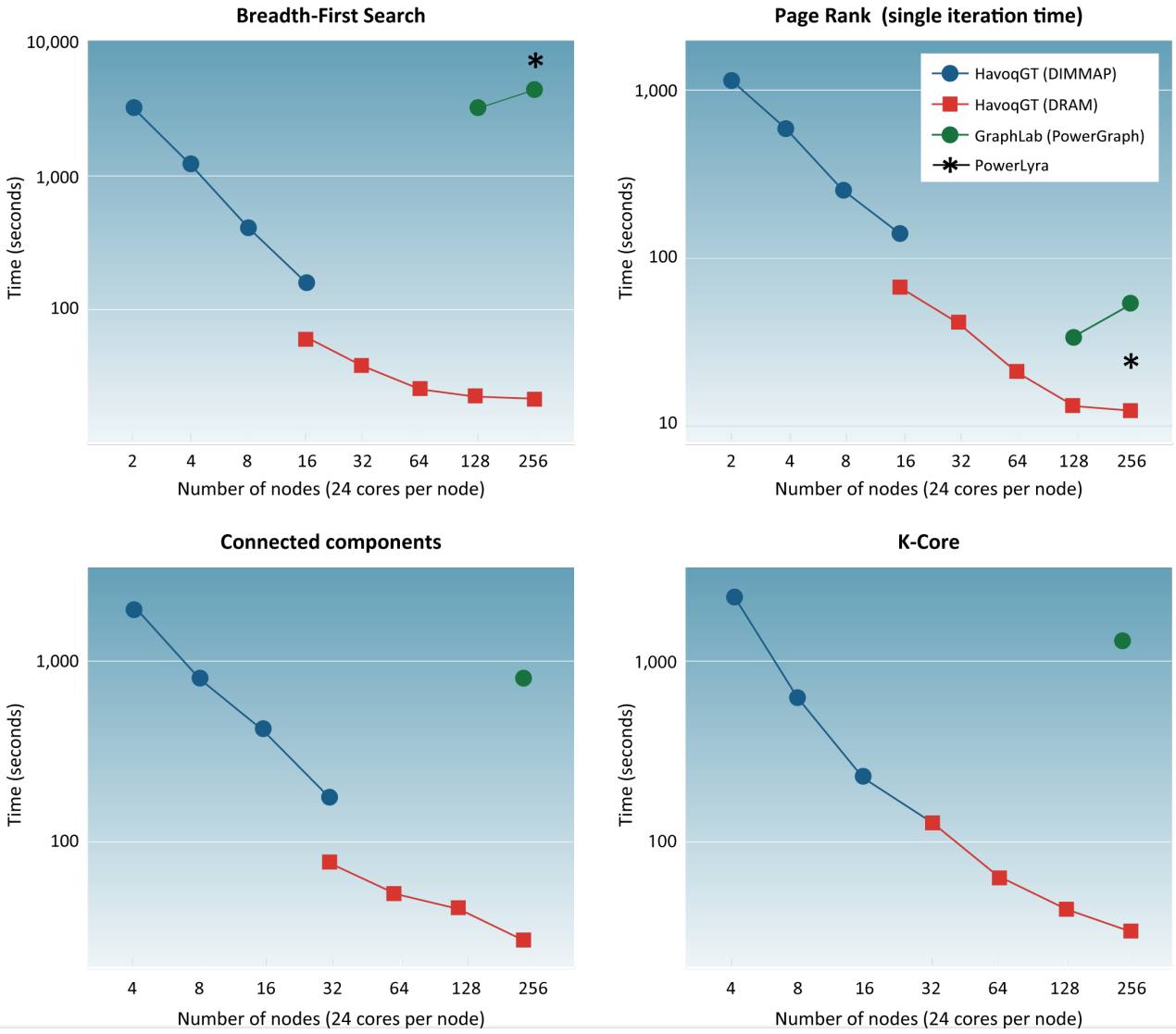
High-degree vertices create load imbalances (left). Our partitioning and parallelization techniques mitigate the imbalances of irregular data common to real-world datasets by distributing edges of high-degree vertices (right).

Communication Optimizations for Scale-Free Graphs



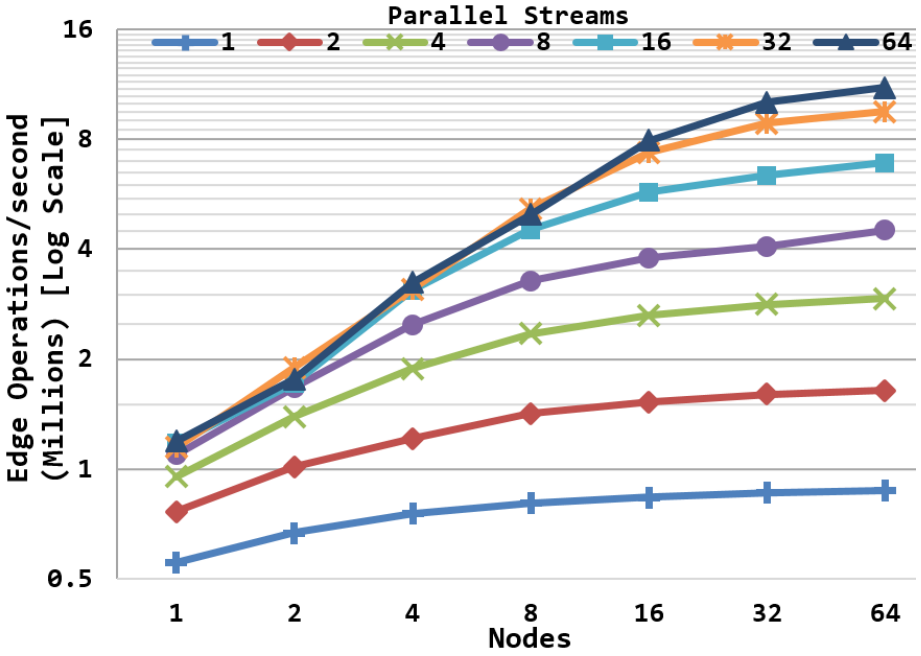
The synthetic message routing network alleviates dense processor−processor communication. Messages are first routed and aggregated locally on the compute node, then communicated to their final destination

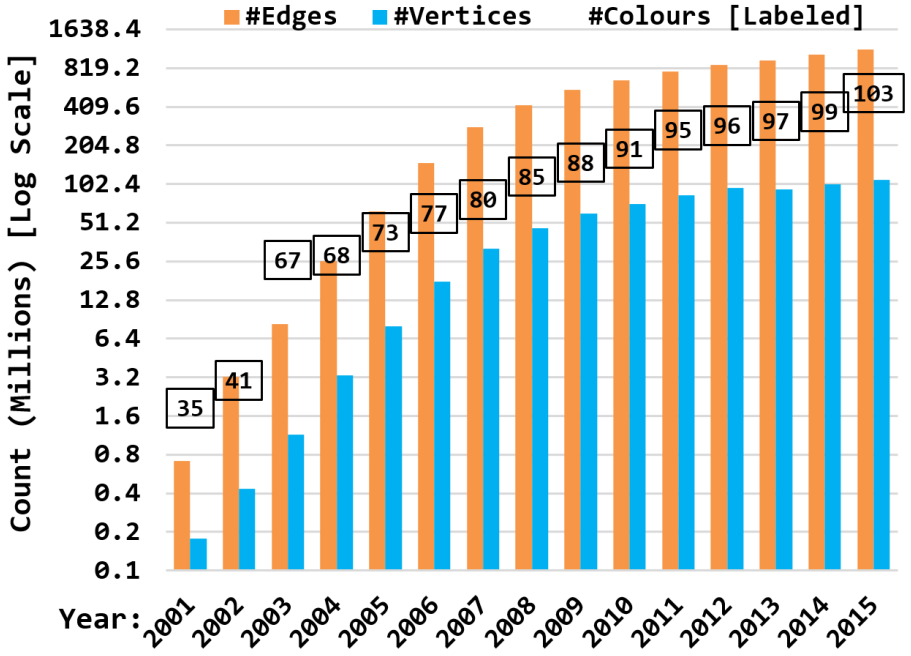
128B edge Webgraph on Catalyst (Strong Scaling)



Static graph processing on 128 billion directed edges webgraph from largest known open-source graph dataset. Our approach outperforms the cloud-based solution GraphLab, both in running times and memory resources.

Streaming Coloring on Wikipedia Graph [SC'16]





- Dataset extracted from 13+ years of Wikipedia edit history
- Coloring updated as edges are inserted or deleted from multiple streams

[Scott Sallinen (UBC)]

How will low-latency memory-attached NVM impact large scale graph analytics?

- Are DRAM caches necessary?
  - Previously used ‘mmap’ for DRAM cache to NAND Flash
  - Significant prior work to develop efficient mmap runtimes [DI-MMAP]
- Are scratchpads, or manually managed fast temp memory, necessary?
  - Previously manually managed vertex set in DRAM with graph stored in NAND Flash
  - Semi-External Approach
- Will NVMs improve streaming graph processing?

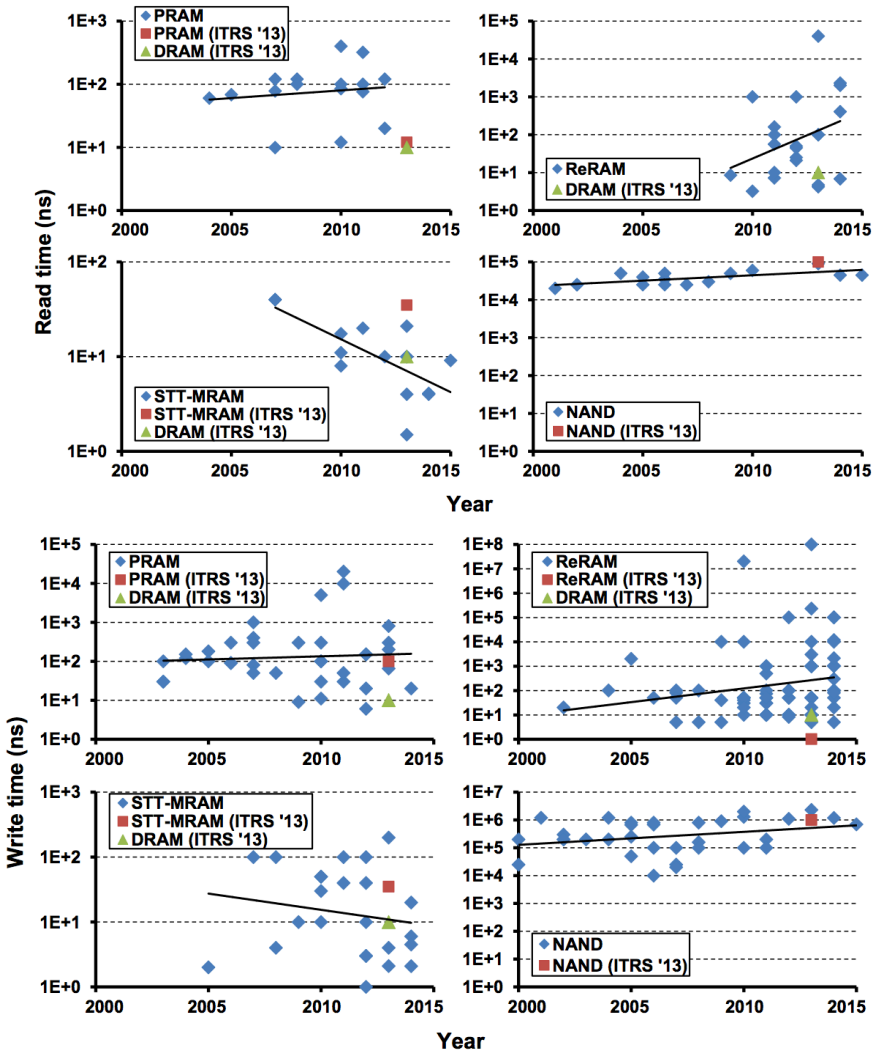
Low-latency NVM

- NVMs attractive choice for graph processing algorithms
- Emerging NVM technologies expected to have a range of latencies
- What is the impact of such latencies on the performance of locality optimized graph processing algorithms?

Table 1: Characteristics of different memory technologies

Memory Technology	Read delay (ns)	Write delay (ns)	Read energy (pJ/bit)	Write energy (pJ/bit)
RAM	10	10	10	10
PCM	21	100	12.4	210.3
STTRAM	35	35	58.5	67.7
FeRAM	40	65	12.4	210

Ref: Evaluation of emerging memory technologies for HPC, data intensive applications

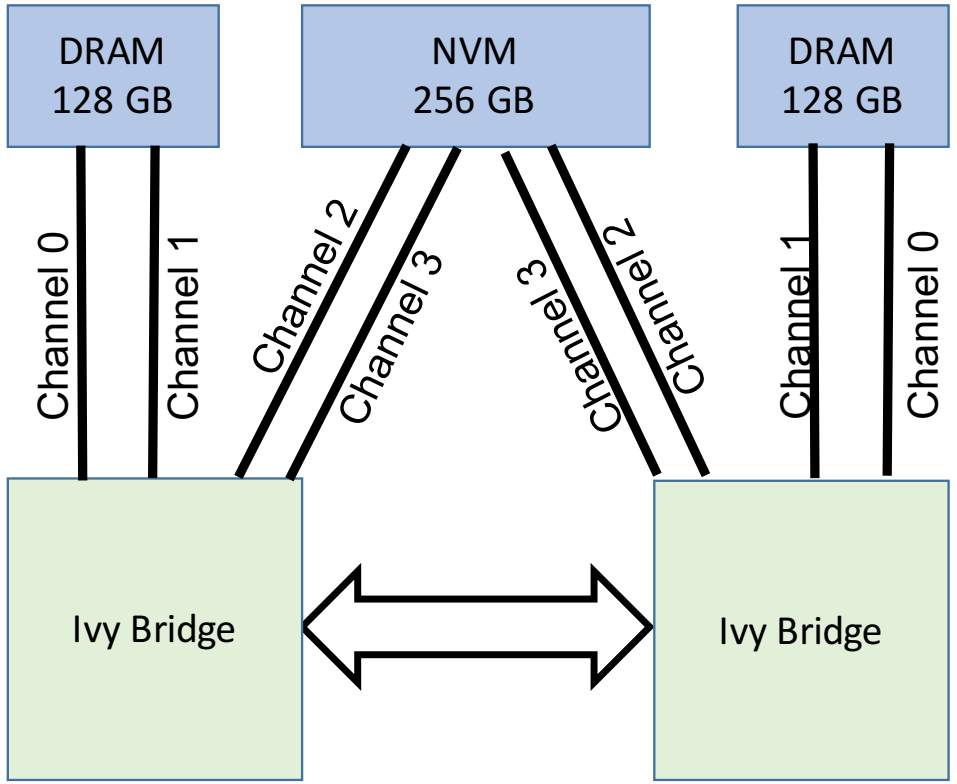


Ref:The Non-Volatile Memory Technology Database

Intel's Software Emulation Platform (SEP)

@ SDSC

- SEP is a dual socket Ivy Bridge (E5-4620 v2)
- 512 GB of DDR3 1600MHz DRAM
  - 2 NUMA DRAM pools: 128GB per numa node
  - 256 GB UMA NVM-like pool
- Can vary read latency from 100ns – 350ns

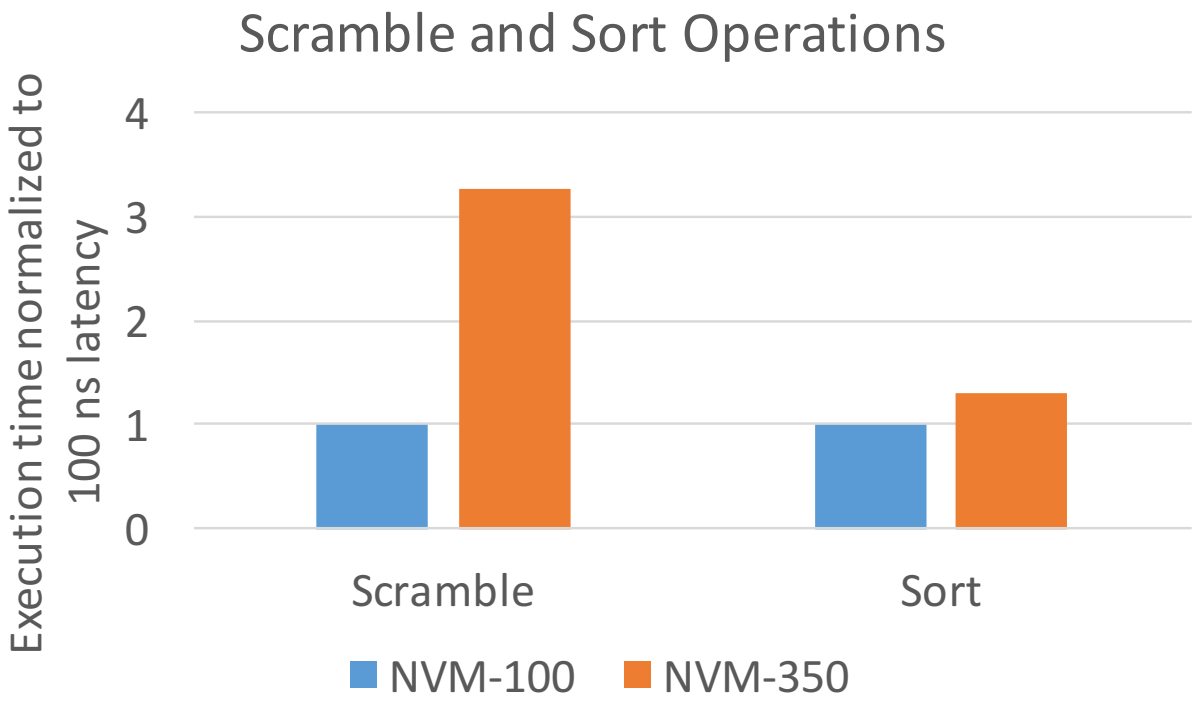


Experimental Setup

- System: Intel's SEP system
- 8 MPI ranks, varying read latencies from 100ns – 350ns
- Algorithms
  - Static: BFS, PageRank, Connected Components, k-core
  - Dynamic: Graph update operations
- Input graphs
  - RMAT, Twitter, SK2005, Wikipedia
- Evaluate two memory modes
  - Semi-external (SEM)
    - Graphs in NVM
    - All dynamic intermediate data structures in DRAM
  - Fully external (EM)
    - All data in NVM
    - Use LD_PRELOAD to redirect memory allocation calls to NVM

Initial Benchmarking

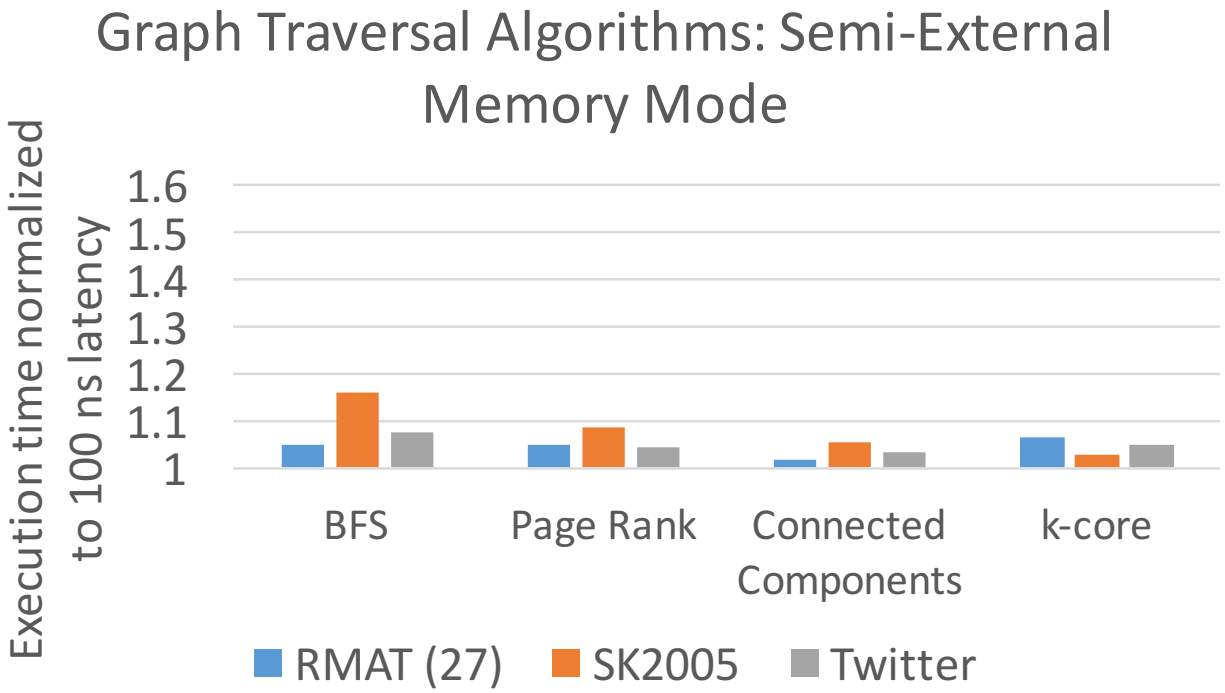
Scramble & Sort



- 3x slowdown for cache-unfriendly scramble operation
- Only 30% slowdown for sorting (c++ quicksort)

Semi-External Memory Mode

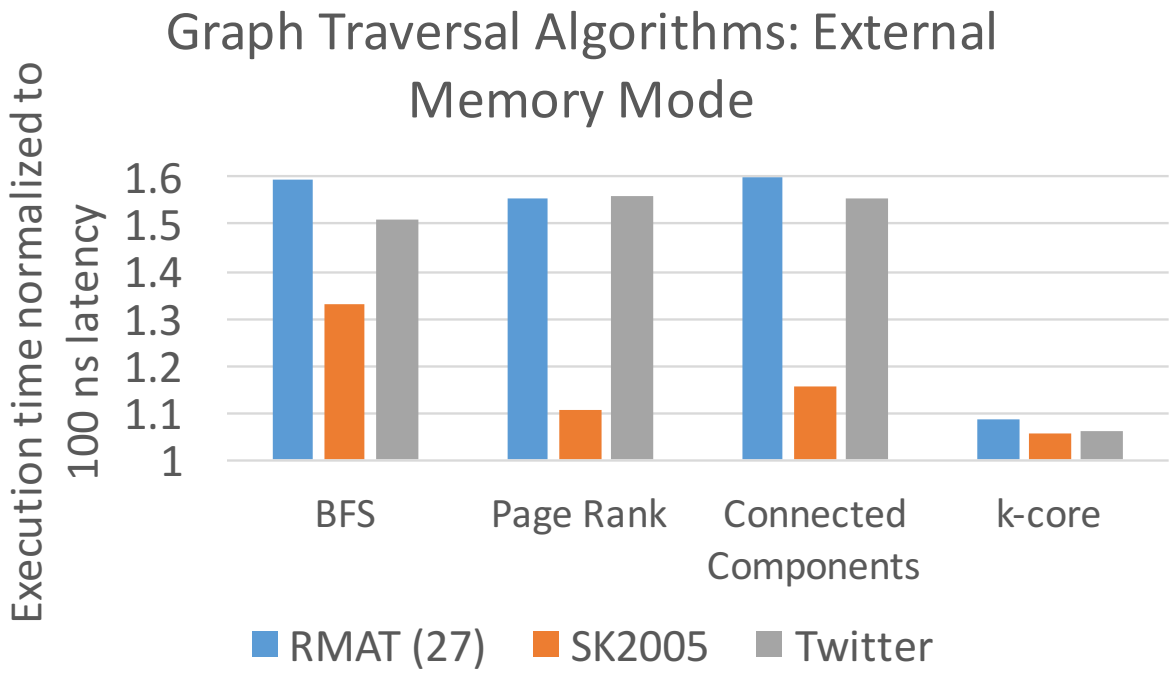
Graph in NVM, Alg & temp data in DRAM



Most tests show less than a 10% performance penalty with increased read latency (350 ns)

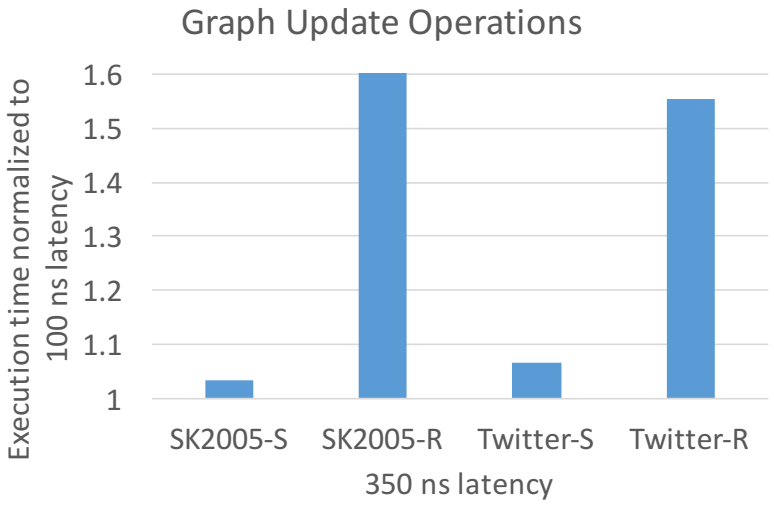
Fully External Memory Mode

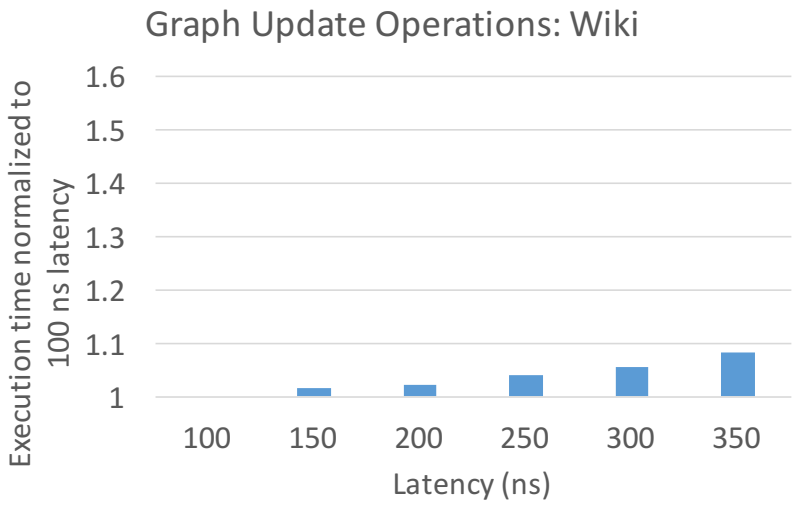
Graph & Alg data in NVM



- Performance impact on real graphs range from 5% (for k-core) to 59.6% for 350 ns read latency
- Performance related to algorithm and the input graph structure

Performance of Streaming Graph Operations





- Random orderings impact performance at higher latencies
- Low-latency NVM useful for dynamic streaming graph applications, especially for **real-life** scenarios
- Caveat: Write latencies are not modeled with Intel SEP system
- Streaming graph storage designed by Keita Iwabuchi (Tokyo Tech & LLNL)

Emerging NVMs are an attractive choice for realistic graph processing algorithms

- DRAM related scratchpads and/or caches likely required
- Hybrid memory configurations
  - Application specific scratchpad beneficial
  - Application aware data structure placement
- Future work:
  - Evaluate DRAM hardware cache for these algorithms
  - Investigate additional storage tiers such as Flash SSD in addition to low latency NVM
- Open Source Graph Framework: http://software.llnl.gov/havoqgt/
- **Students Welcome!**