Design of a NVRAM Specialized Dynamic Graph Data Structure

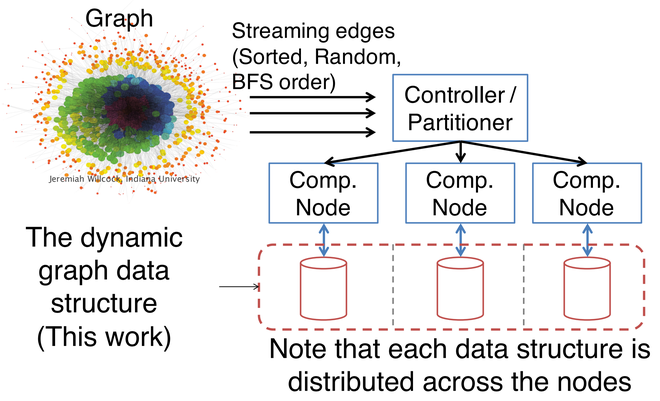
Keita Iwabuchi[1,2], Roger A Pearce[2], Brian Van Essen[2], Maya B Gokhale[2], Satoshi Matsuoka[1]
1. Tokyo Institute of Technology (Tokyo Tech) 2. Lawrence Livermore National Laboratory (LLNL)

Motivation

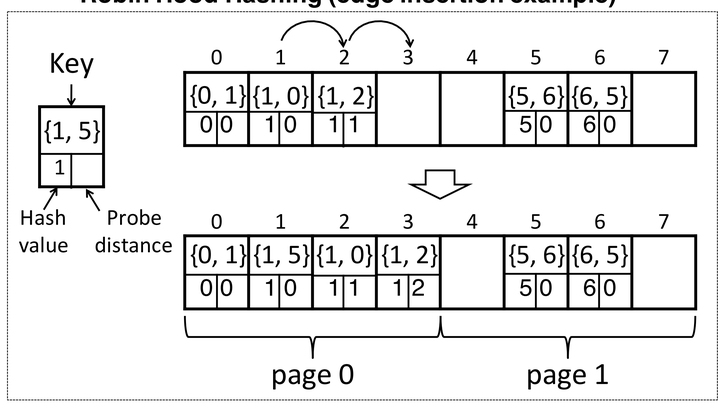
Store and Process Large Dynamic Graphs
- Social network, genome analysis, WWW, etc.
- Streaming graph updates (insert or delete edges or vertices)
- Efficiently store sparse scale-free graphs

Leverage Emerging NVRAM in HPC Systems
- NVRAM has lower cost and power consumption than DRAM
- Persistently store distributed graph database across compute nodes with attached NVRAM
- Extends node's memory capacity

Goal
High performance:
- Insertion and deletion of vertices and edges
- search for a specific edge based on edge meta data



Robin Hood Hashing (edge insertion example)

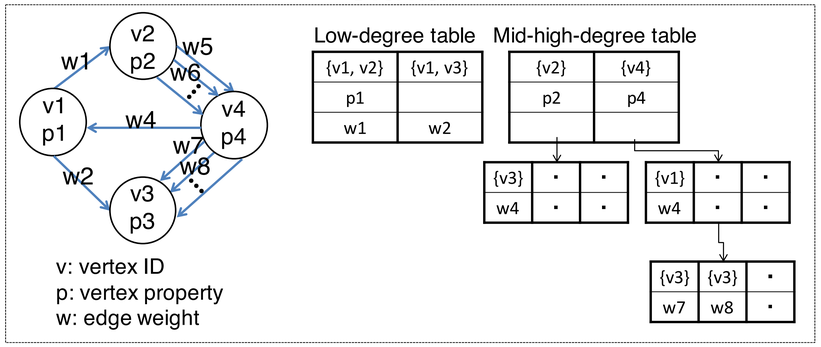


A NVRAM Specialized Degree-Aware Dynamic Graph Data Structure

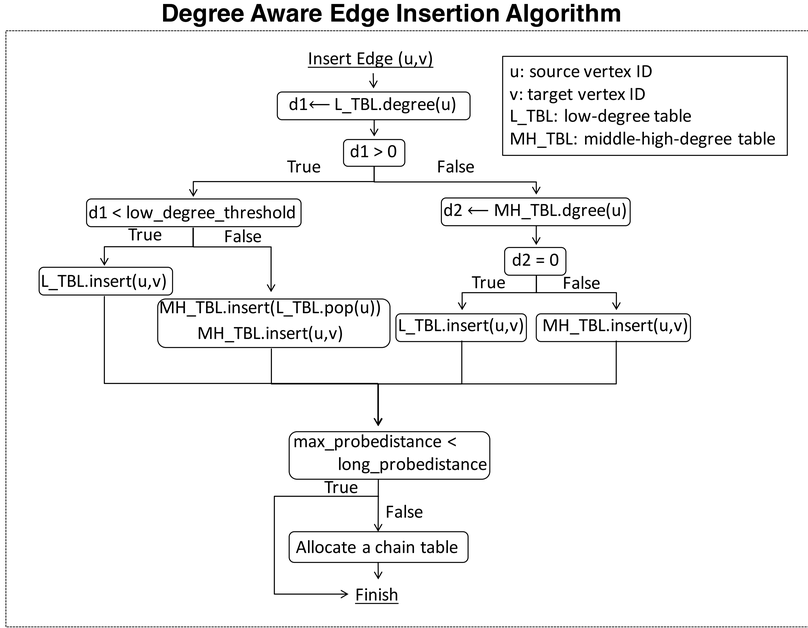
Key Design Objectives
- Increase page-level locality of data stored in NVRAM
- Optimize for low degree vertices
- Efficiently search and retrieve vertices, edges, and metadata
- Quickly locate a specific edge matching topological and metadata constraints

Our Approach
- Degree aware data structures, where low-degree vertices are compactly represented
- Use Robin Hood Hashing [Celis '86] because of its locality properties

Overview



Degree Aware Edge Insertion Algorithm



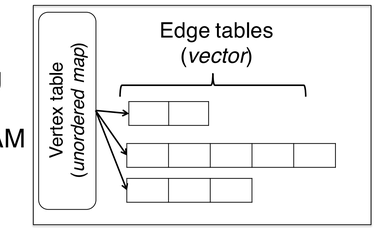
Experiments

Configuration
- Catalyst cluster at LLNL with 800GB of NVRAM per node (single node)
- Memory mapped I/O using DI-MMAP as an interface to NVRAM, limiting the DRAM resident portion of graph DB (page buffer) to 4GB
- *Boost.Interprocess* to allocate data structures in memory-mapped region

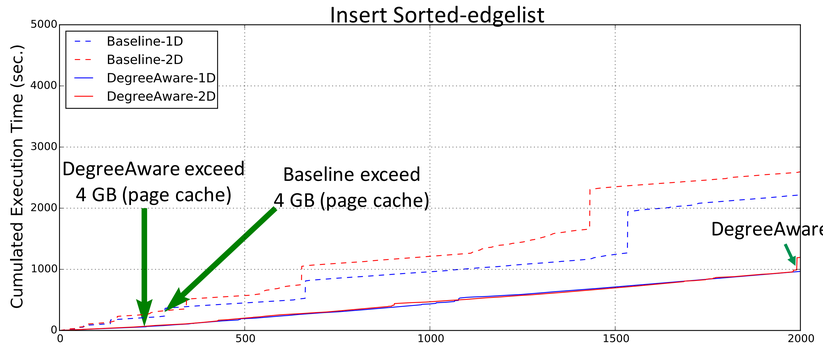
Dataset
WebGraph 2012 [Lehmberg'14]
- Largest open source webgraph to our knowledge, 120 billion of edges
- Graph is 1D or 2D partitioned, modeling 64 partitions
- Vertex: webpage, Edge: hyperlink, Weight: N/A

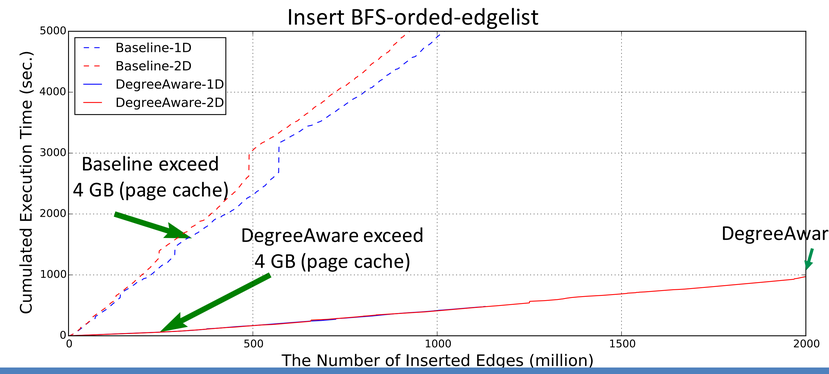
Streaming edge insertion
1. Streaming edges are ingested in Sorted, Random, or BFS order
2. Partition the edges into 1D or 2D partitioning (#partitions is 64)
3. Buffer a subset of edges (1 million) into DRAM
4. Insert edge buffer into the graph data structure sequentially

Baseline model (Boost)



Results
- Degree aware data structures scale near-linearly with the number of edges inserted (up to 2 billion edges)
- Robin Hood Hashing improves page-level locality and overall performance when graph database grows beyond 4GB page cache





This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-POST-676096