Introducing Lucata's GraphBLAS

v1.0 is tagged! And tested!

Jason Riedy and Shannon Kuntz

13 Oct 2021

Lucata Corporation

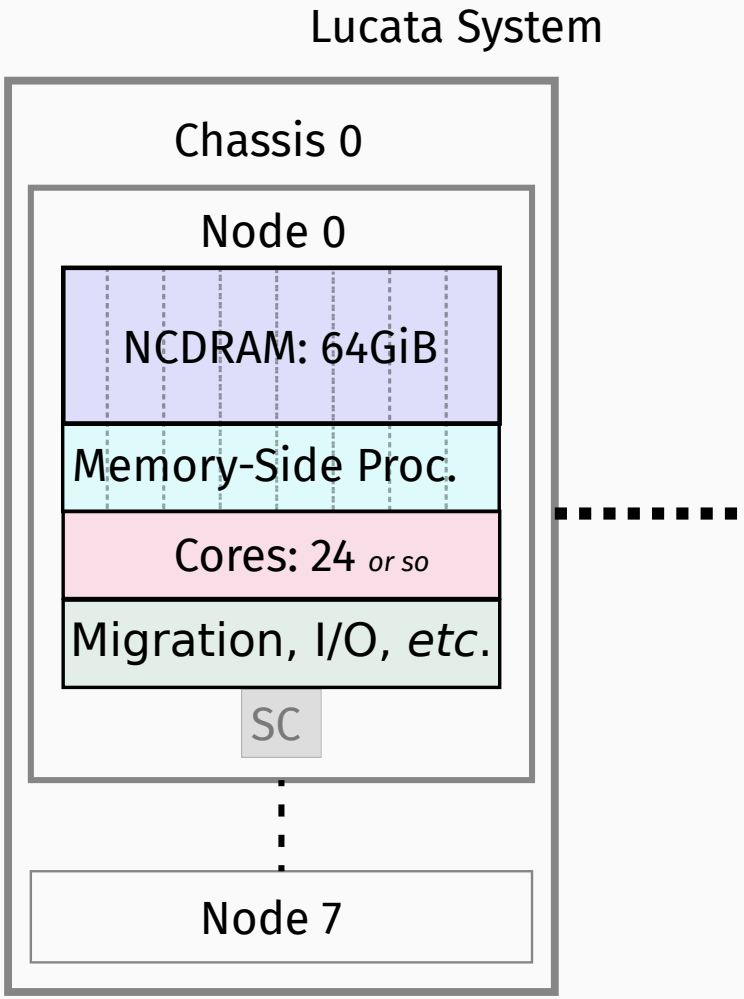
Parallel Graph Analysis

Known issues:

- Scattered memory access w/ small(?) seq. bursts
  - Cache lines provide fraction of avail. BW
  - Prefetechers fire up then mis-predict
- Large bursts (high degree) $\Rightarrow$ load imbalance
  - Combined with 90% diameter ≤ 8.
  - Plenty of load-balancing pre-processing...
- Streaming: The graph is changing.
  - Pre-processing can hurt where and when changes are interesting.

CPU+cache systems have one set of coping mechanisms. GPGPUs / flex. vectors another. *And Lucata has yet more...*

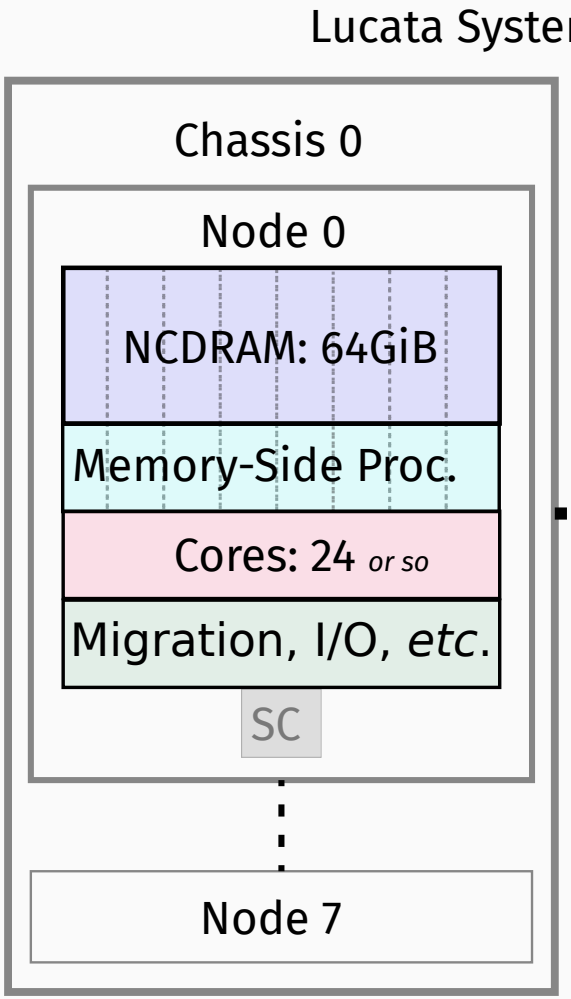
The Lucata Pathfinder PGAS architecture



Four chassis system is a 2 TiB NSF CCRI resource at crnch.gatech.edu

- **Optimized for weak locality**
  - Scattered jumps and seq. access
- Stationary core for OS per node + SSDs
- Hardware partitioned global address space (PGAS) with a twist
  - Plenty of network BW, low latency
  - Details in a moment...
- Multithreaded multicore LCE (or GC)
  - Currently 1536 threads per node, 12k per chassis, 50k per 4 chassis.
  - "Helps" with load balance
  - No cache.

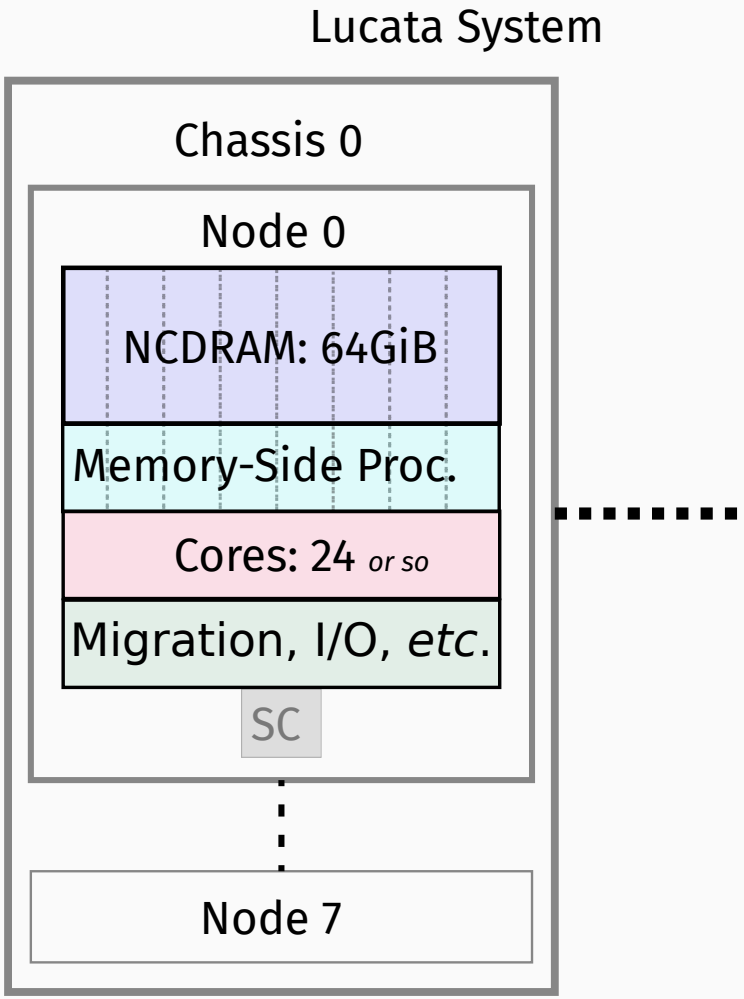
Lucata's PGAS Twist for Weak Locality



Four chassis system is a 2 TiB NSF CCRI resource at crnch.gatech.edu

- Threads write **remotely**, always read **locally**
- Writes: 8 Memory-side processors (MSPs)
  - Writes+ don't touch the cores.
  - Handle some arithmetic ops. (FPADD)
  - Deep queue, no control flow
- Reads ⇒migrate⇐.
  - *Hardware*: Remote read ⇒ package and send the thread context
  - Read latency is local up to migration.
  - Control flow depends on reads.
  - Contrast with Tera MTA / Cray XMT: Need far fewer threads, far less network bandwidth.

Programming the not-Beast: Not painful.



Four chassis system is a 2 TiB NSF CCRI resource at crnch.gatech.edu

- PGAS: Read and write directly.
- Memory *views* implemented in hardware
  - Intra-node `malloc`
  - Node-striped `mw_malloc1d`
  - Node-blocked `mw_malloc2d` ...
  - Implemented by pointer bitfields
- Fork/join parallelism: Cilk+ + extensions
  - Yes, Cilk+ is alive: OpenCilk
  - Fast: Spawning a thread ≈ function
  - Composes: “Serial elision.”
  - Collectives? In progress.
    - Some Cilk+ reducers map perfectly.

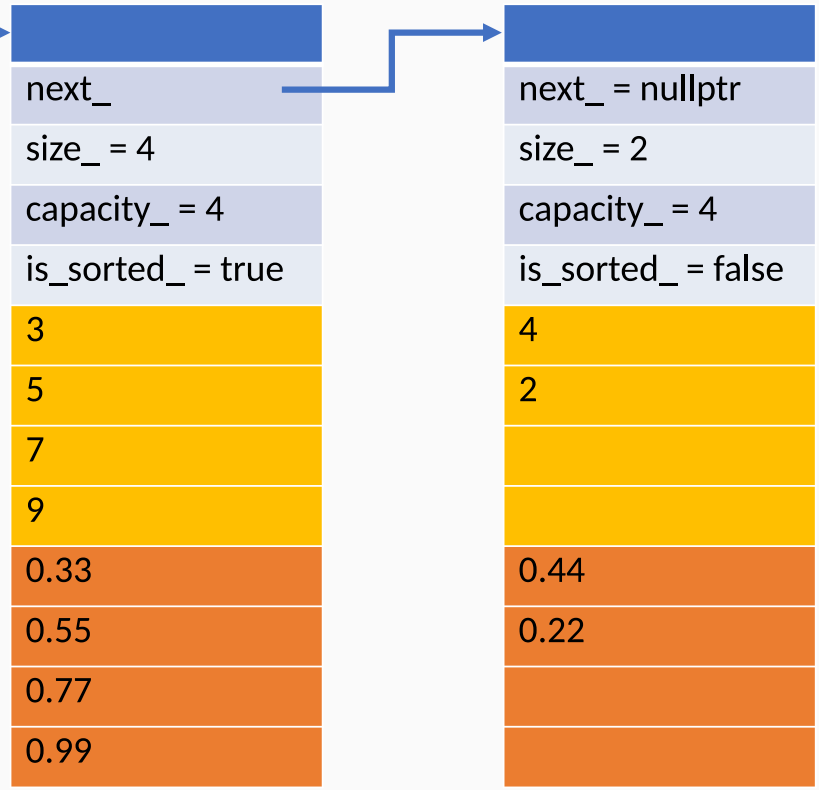
Challenges in Implementing the GraphBLAS

- Naïve kernels may migrate for every edge.
  - Gustavson's SpGEMM
  - Treat with a form of message aggregation
- Collectives like prefix-scan...
  - Fork/join is cheap, but not free.
  - Prefix-scan must join, (re-fork, re-join,)^2 join
  - (*e.g.* Existing code from Arch Robinson.)
- Separate memory spaces: SC $\cap$ Lucata $= \emptyset$
  - From SC: Data structures fully opaque
  - From LCE/GC: Trigger but not fulfill transfer
  - Anything that operates edge-by-edge via or outside the GraphBLAS: NO

Current LGB Data Structure Perspective

Rows are striped across nodes, four here.

i_	rows_
0	nullptr
1	nullptr
2	
3	nullptr
4	nullptr
5	nullptr
6	nullptr
7	nullptr
…	



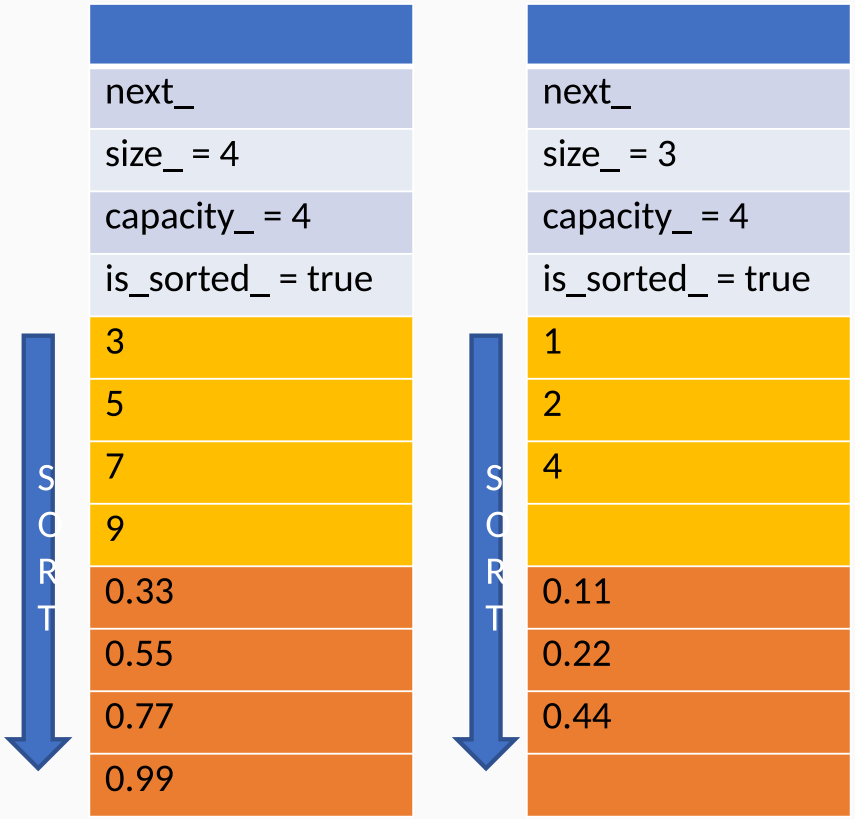
Nonzeros:
A[2, 3] = 0.33
A[2, 5] = 0.55
A[2, 7] = 0.77
A[2, 9] = 0.99
A[2, 4] = 0.44
A[2, 2] = 0.22

type_	x_size_
double	8

Nonzeros are stored in a linked list of blocks. Type information is stored separately in the matrix. Each block consists of a header, a list of indices (yellow), and a list of values (orange).

Current LGB Data Structure before Operating

i_	rows_
0	nullptr
1	nullptr
2	
3	nullptr
4	nullptr
5	nullptr
6	nullptr
7	nullptr
...	



type_	x_size_
double	8

sort_and_merge() creates a single sorted block. First each block is sorted individually, then blocks are pairwise merged until we have a single sorted block.

Future Aspects (Student Opportunities?)

- Add a fast, fixed-size block pool per matrix
  - Fast de-allocation for temporaries
  - Or ignore the pool, CSR-ish for marked temps
  - Can alloc. consecutive blocks, next = count

- Opportunities for large-degree vertices!
  - Can **stripe** across nodes, views are transparent
  - *Composing* parallelism ⇒ more options than current big/small splits (*e.g.* MTGL)
  - Feed back into deeper Lucata spawning / flow control decisions
- Extending CMU/SEI's GBTLX (C++ with SPIRAL) for PGAS, migratory threads

And Streaming! (NSF SBIR 2105977)

- HW supports multi-Gig network ingest per node
- Large-scale locking or snapshotting is a no-go
  - Starting current streaming/updating algs...
- No locking: "Valid" algs reading edges atomically[1]
  - Starting graph + *some* subset of concurrent changes
  - BFS, connected components, linear algebra centralities (PageRank, Katz), triangle counting
  - Copying subgraphs also... Seed set expansion
  - In turn can be updated "validly."
- Other approaches: Aspen (already in Cilk+)

[1] Chunxing Yin and J.R. Concurrent Katz Centrality for Streaming Graphs. HPEC 2019. DOI 10.1109/HPEC.2019.8916572