# A Message-Driven, Multi-GPU Parallel Sparse Triangular Solver

*Nan Ding*, Samuel Williams, Yang Liu, Xiaoye S. Li
nanding@lbl.gov

U.S. DEPARTMENT OF ENERGY
UNIVERSITY OF CALIFORNIA
BERKELEY LAB
Lawrence Berkeley National Laboratory

# Bigger problems + not enough GPU memory -> multiple GPUs

- Demand for ever finer-resolution problems
- Can not always fit into a single GPU's memory
- GPUs have become a first-class compute citizen
  - 110/147 system use NVIDIA Volta chips in 2020, Top500 list[1]

[1] https://www.top500.org/

# Highlights

- **Multi-GPU SpTRSV using CUDA streams**
  - Up to 6x obtained for multi-GPU SpTRSV
  - kernel specialization on GPUs for DAG-based computations
  - Critical path model to explain/predict the performance

- **One-sided communications enabled distributed tasking on GPUs**
  - One-sided messaging libraries can vary substantially
    - ➢ Cray's one-sided implementation is 2.7x slower than Cray's two-sided yet ETH's foMPI is 3x faster than Cray's two-sided
    - ➢ NVSHMEM is 2.3x slower than IBM Spectrum On the Summit InfiniBand network
  - Need inter-node network performance improvement

- **Future work**
  - Port to other accelerators, e.g., AMD GPU with ROC_SHMEM
  - Use critical path model to identify potentially superior process mappings

# Acknowledgements

This research is supported in part by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) programs under Contract No. DE-AC02-05CH11231 at Lawrence Berkeley National Laboratory. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

U.S. DEPARTMENT OF ENERGY

UNIVERSITY OF CALIFORNIA

BERKELEY LAB
Lawrence Berkeley National Laboratory

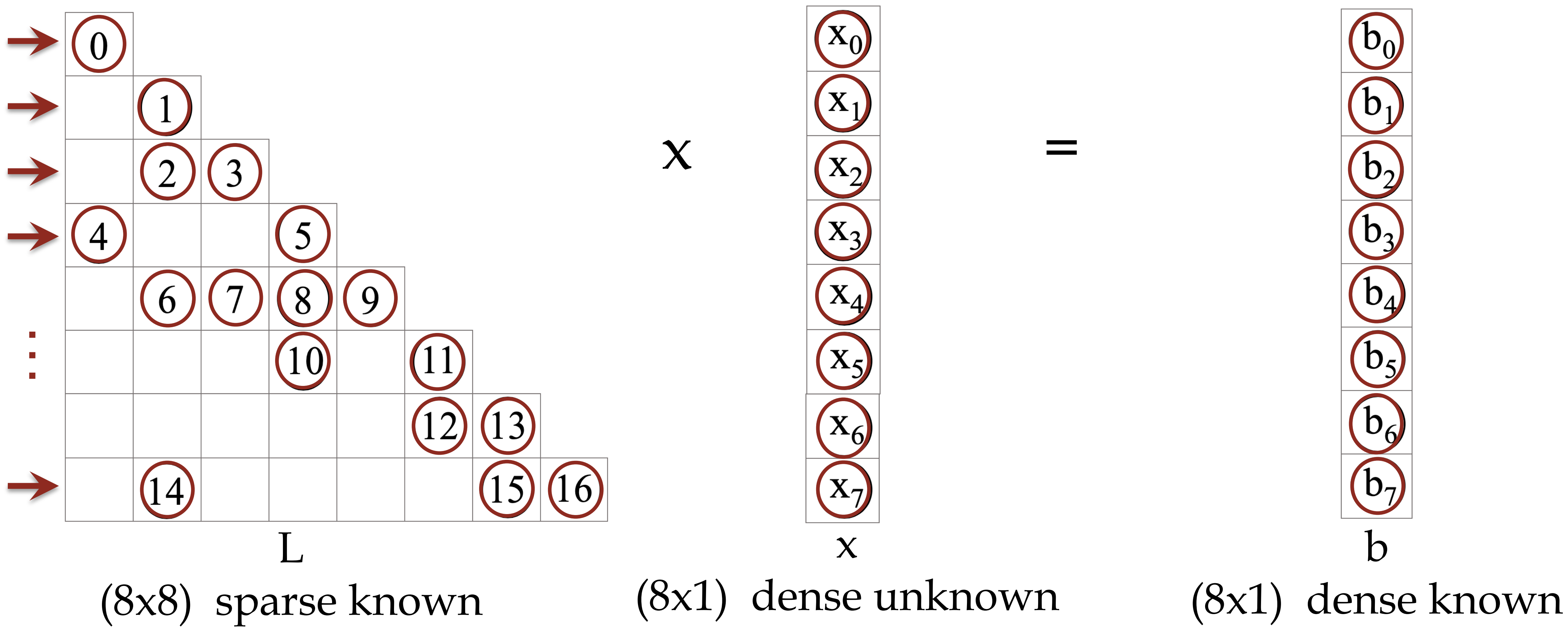
# Sparse Direct Solvers

- Sparse direct solvers
  - Block Jacobi preconditioning
    - LU factorization (a simplified/approximate system)
      - Factor once and use as a preconditioner across multiple solves
    - L- and U- solve (SpTRSV)
      - Shifts the focus to SpTRSV performance

- Challenging:
  - Low arithmetic intensity
  - Complex data dependencies
  - High inter-node communication

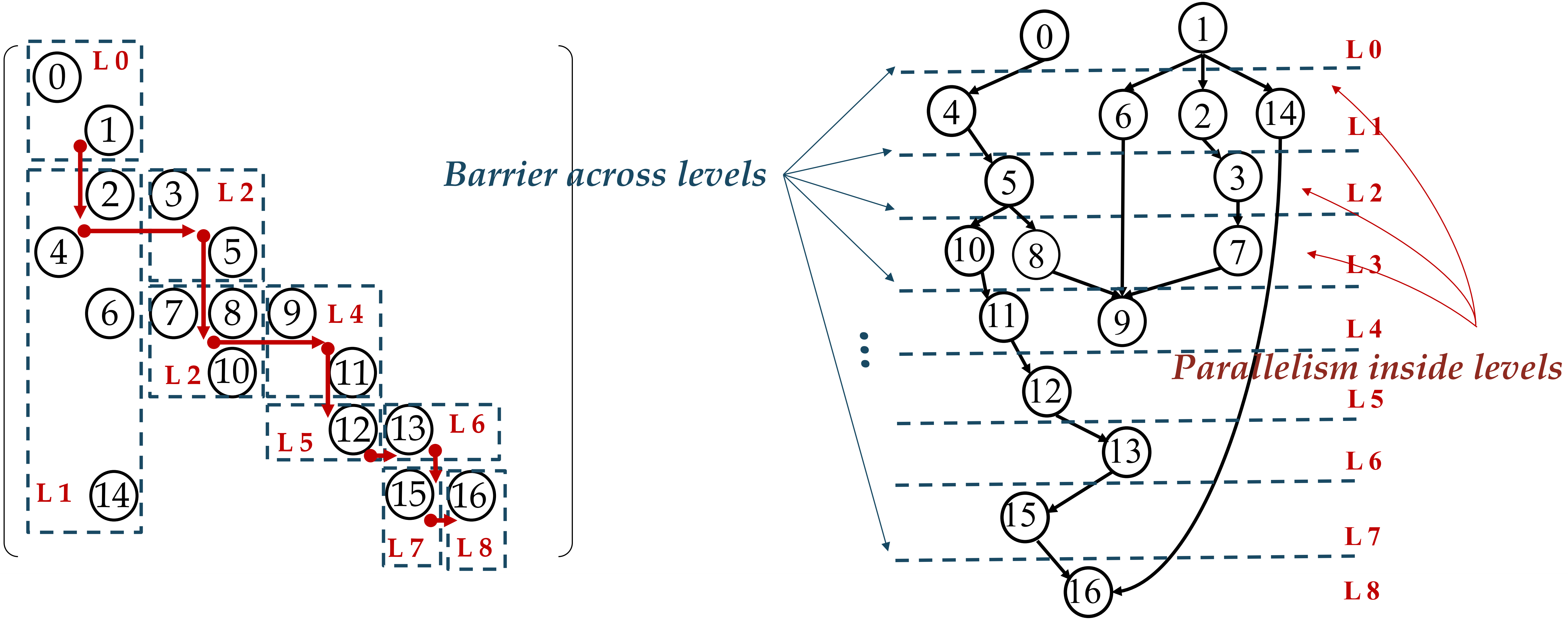
# Naïve BSP SpTRSV

- Compute solution vector x from a sparse linear system, Lx=b
- Naive approach:
  - solve the system one equation (row) at a time,
  - can be optimized to (selectively) parallelize over column updates or row reductions



# Recast SpTRSV as a DAG

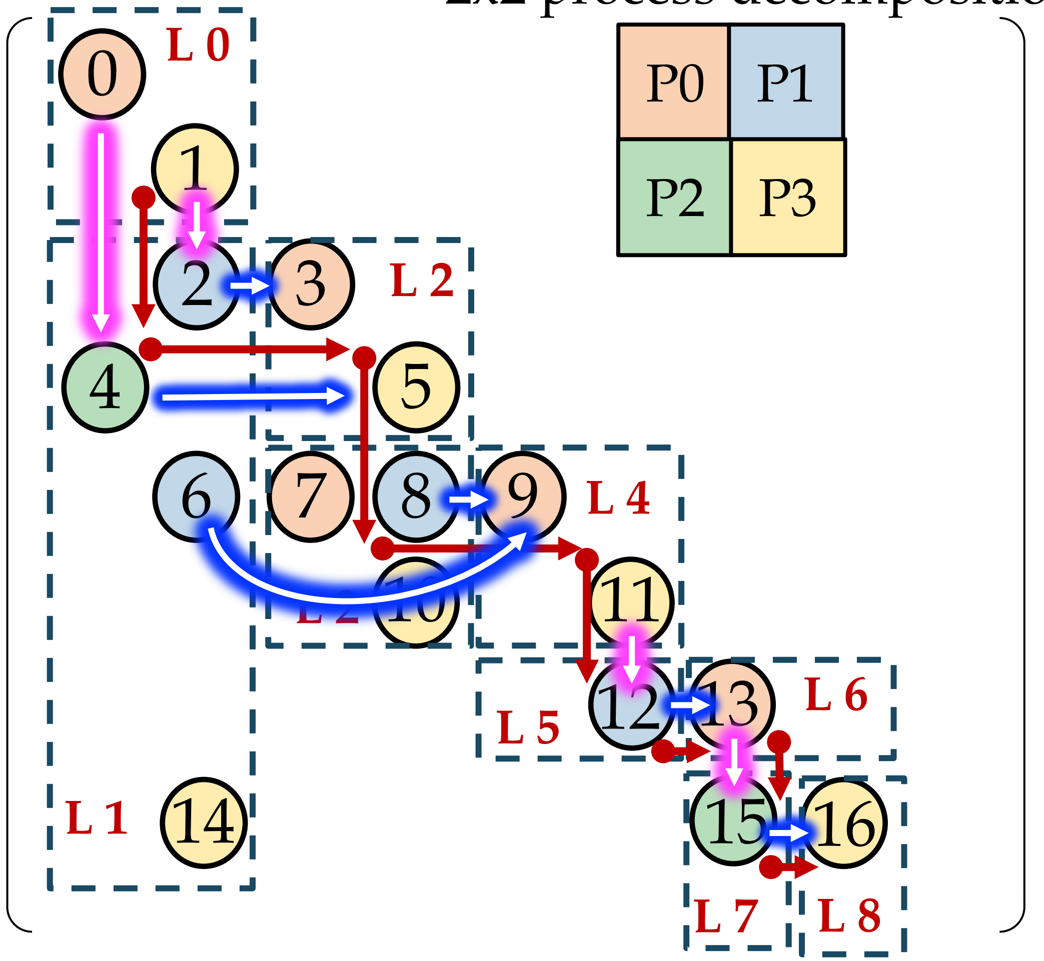
- Computation = Directed Acyclic Graph (based on level sets)
- Each node in the DAG is a small dense matrix-vector
- Parallelism is sacrificed in the bulk synchronous approach (data dependencies satisfied, but will not be executed until all previous levels have been executed)

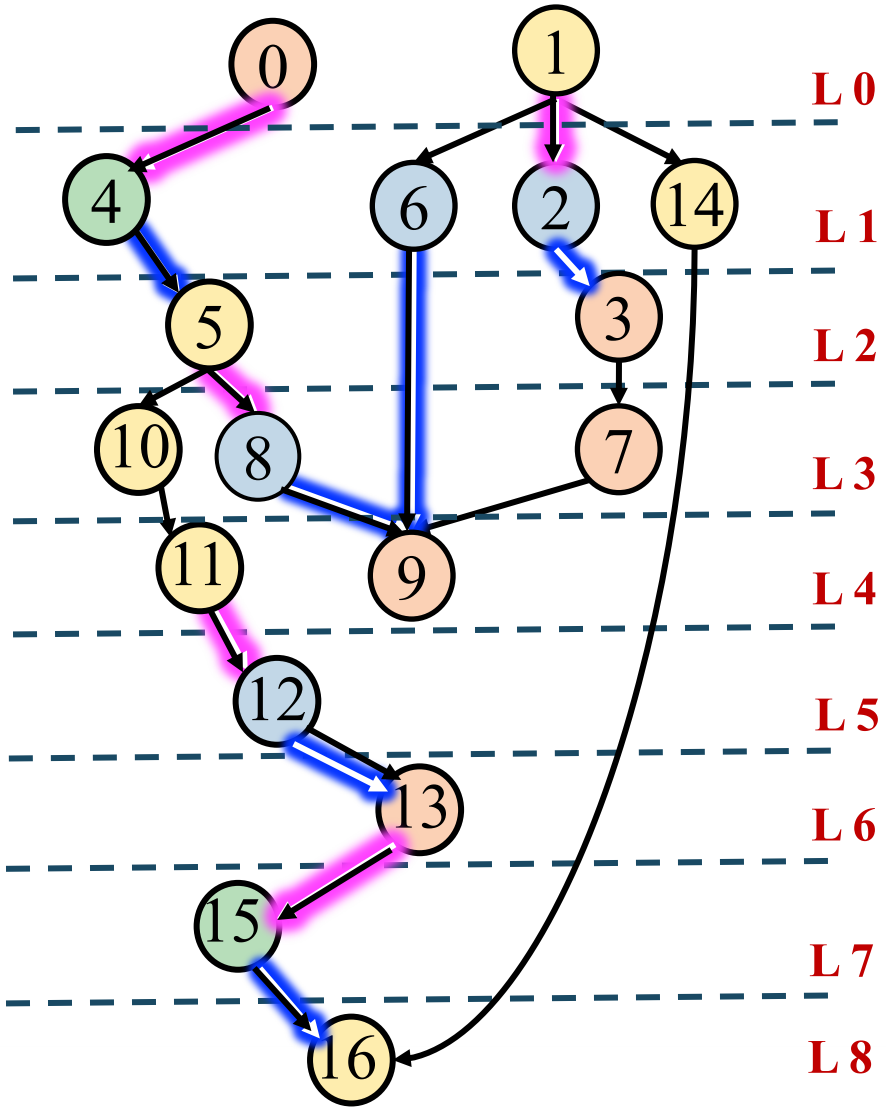


# SpTRSV in SuperLU: Message Driven

- A 2D block cyclic process layout
- Asynchronous communications: no barrier across levels, edges are inter-process communications
- Two types of computation: Solves (on-diagonal blocks), MatVec (off-diagonal blocks)
- Two types of communication: Block column broadcast, Block row reduction
- Typical message size: 256 -1024 bytes
- Demand high messaging performance

2x2 process decomposition



# Previous Messaging Solutions in SuperLU

- Two-sided MPI on CPUs [1]
  - MPI_Isend/Recv

- One-sided MPI on CPUs [2]
  - Computations remain the same with the two-sided solution
  - MPI_Put (non-blocking), each message= data + payload
  - Payload: user-coded checksum for receivers to check data arrival
  - Up to 2.4x vs. the Two-sided MPI solution from 64 to 4096 cores with foMPI[3] library on Cray Aries network

[1] Liu, Yang, et al. "Highly scalable distributed-memory sparse triangular solution algorithms." *2018 Proceedings of the Seventh SIAM Workshop on Combinatorial Scientific Computing*. Society for Industrial and Applied Mathematics, 2018
[2] Ding, Nan, et al. "Leveraging One-Sided Communication for Sparse Triangular Solvers." *Proceedings of the 2020 SIAM Conference on Parallel Processing for Scientific Computing*. Society for Industrial and Applied Mathematics, 2020.
[3] Gerstenberger, Robert, Maciej Besta, and Torsten Hoefler. "Enabling highly-scalable remote memory access programming with MPI-3 one sided." *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 2013.

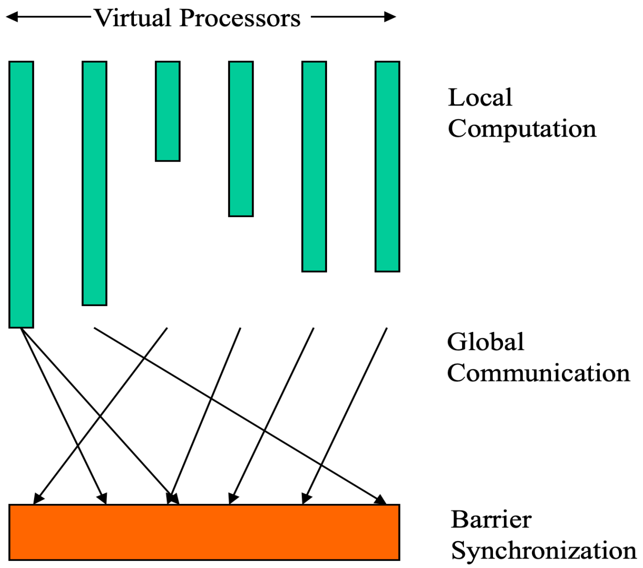
# NVSHMEM has potential
## (but bad implementations can destroy it)

<u>Other MPI,  e.g.,  cuda-aware MPI</u>

✗ initiate communications on CPU
not good for DAG-Based computations
but may satisfy BSP computations
(stencil)

solve<<<...>>>(...)

MPI_Send(...)
MPI_Recv(...)



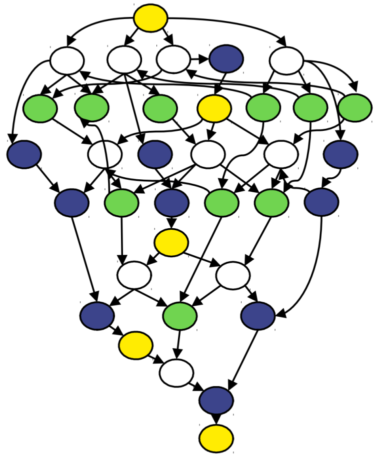

✓ no limitation on #thread blocks

<u>NVSHMEM (based on OpenSHMEM)</u>

✓ uses GPU-initiated data transfers

-> all work can be done in one single CUDA kernel

```
__device__  device_function()
{
  /* computations*/
  nvshmem_double_put_nbi_block(...)
}
```



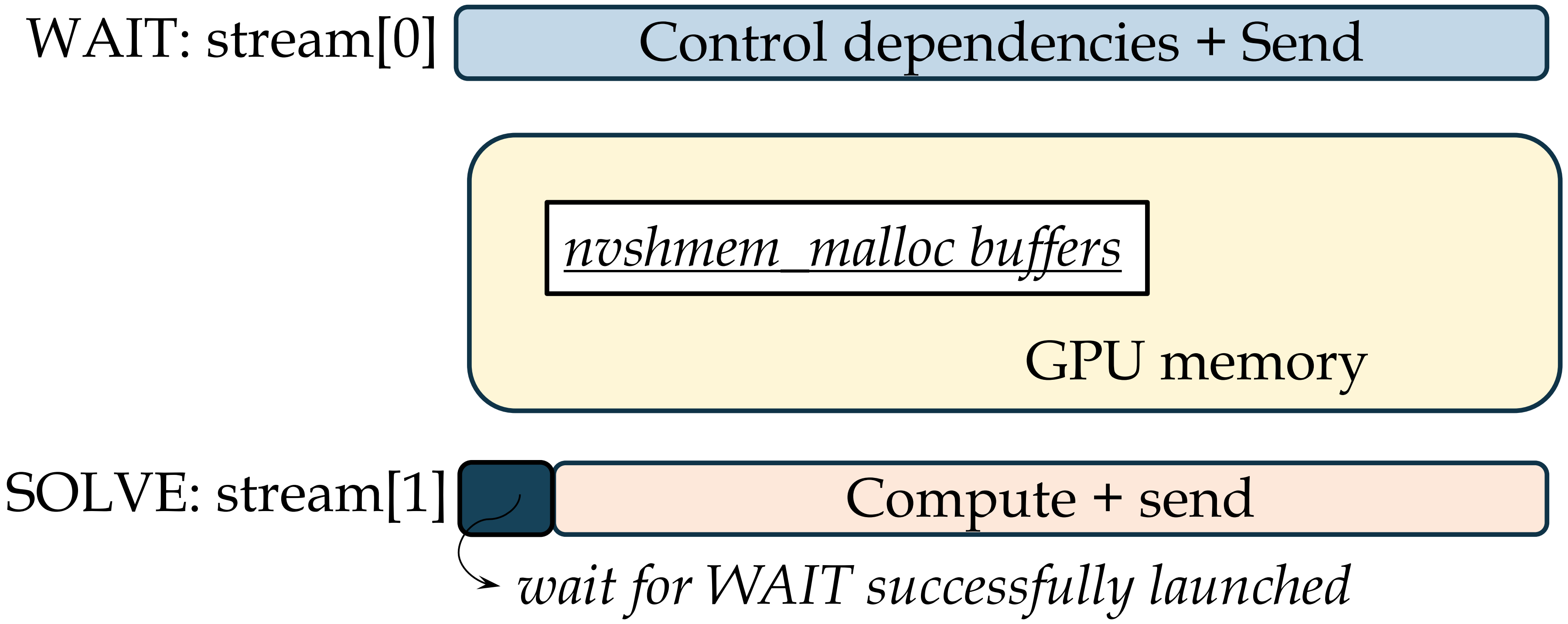
✓ provides signaling operations and point-to-point synchronization operations to notify receivers

✗ limited number of thread blocks that can be launched

# Multi-GPU SpTRSV using two CUDA streams

- Point-to-point communications can happen at any time between any two processes with no strict barrier synchronization
  - depending on the sparsity pattern and the process decomposition

- Leverage high concurrency: processes can proceed its local computations whose data dependencies are satisfied



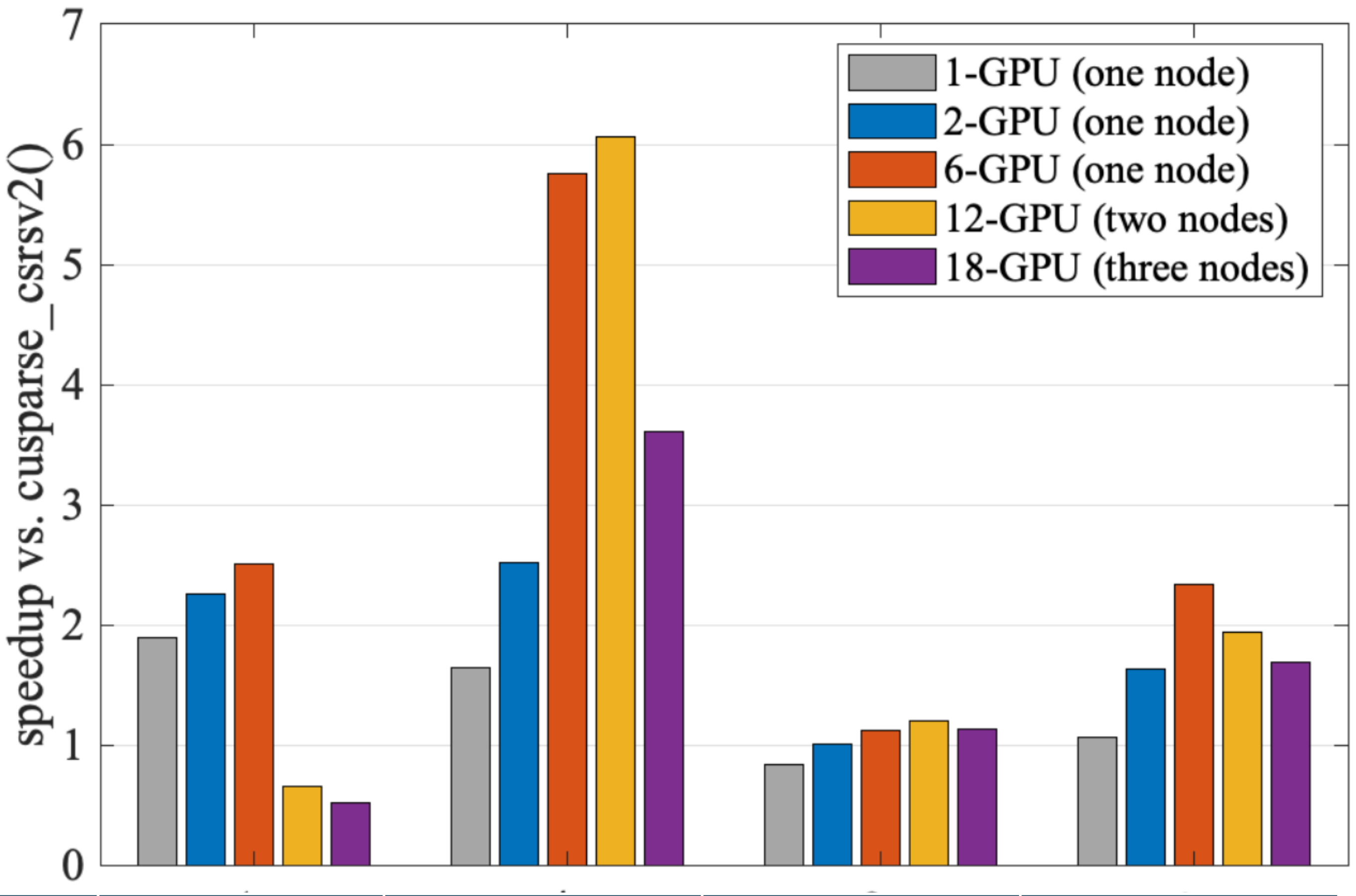
# Multi-GPU SpTRSV vs. cusparse_csrsv2()
# up to 6x speedup (L-solve)

Experimented on Summit:

- Cuda 10, Nvshmem 1.1.3 with Grdcopy 2.0

- bind one process to one GPU
- Px1 process layout (column broadcast)
- use nvshmem_double_put_nbi_block()

- S1 is from M3DC1
- Other matrices are from SuiteSparse Matrix Collection
- factorized via SuperLU_DIST with METIS ordering for fill-in reduction

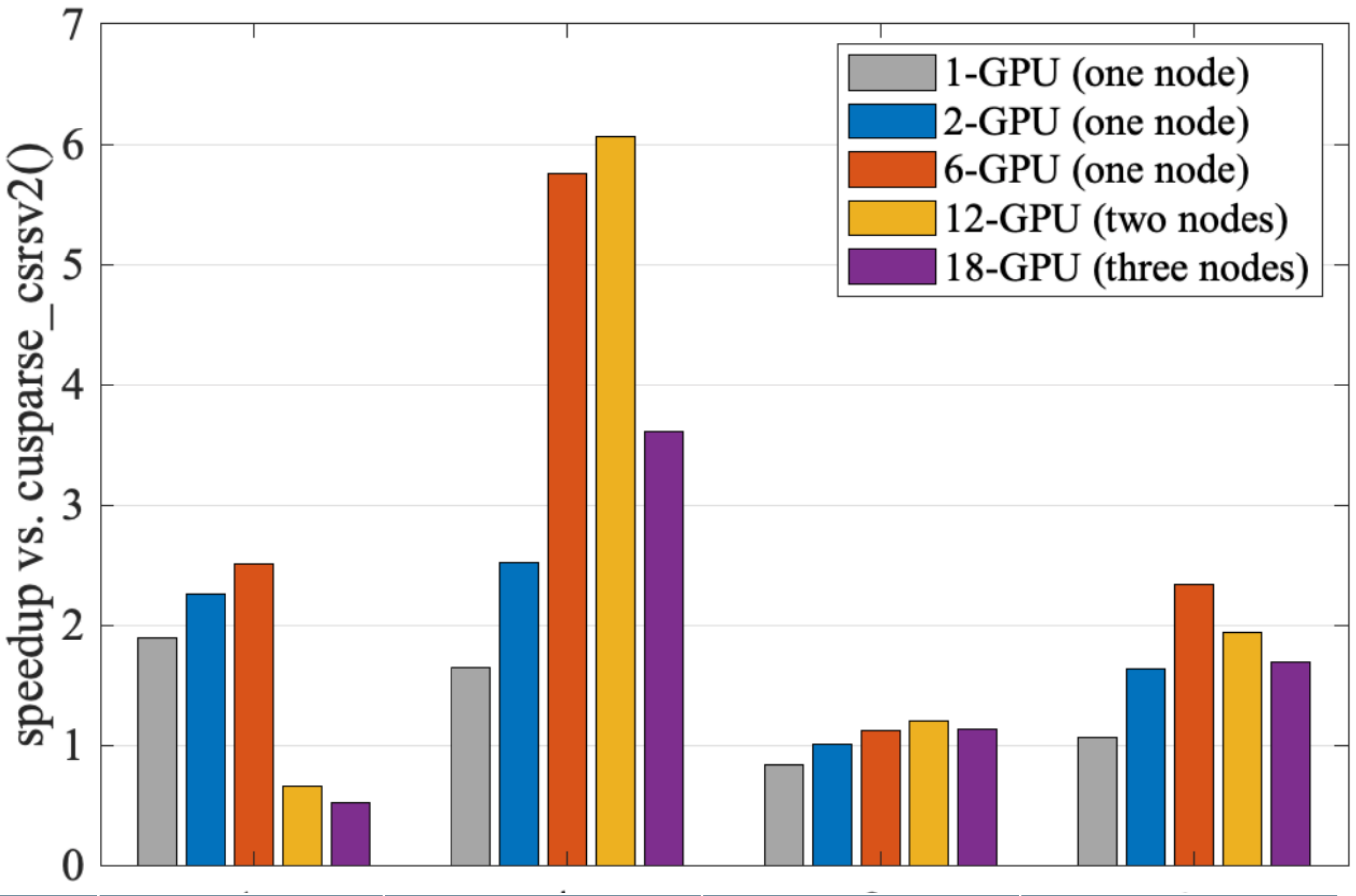



| | S1 | Li | DG | LU |
|---|---|---|---|---|
| nnz | 8.80e+08 | 5.18e+08 | 9.66e+08 | 8.54e+08 |
| DAG levels | 388 | 188 | 199 | 264 |

# Multi-GPU SpTRSV vs. cusparse_csrsv2() up to 6x speedup (L-solve)

Interesting Observations:

- DG has a similar number of DAG levels with Li but more nonzeros -> DG scales better than Li but it's not
- Exploit multiple GPUs on one node, performance is challenged when using multi-nodes




| | S1 | Li | DG | LU |
|---|---|---|---|---|
| nnz | 8.80e+08 | 5.18e+08 | 9.66e+08 | 8.54e+08 |
| DAG levels | 388 | 188 | 199 | 264 |

# It's important to understand what constraint the performance

- Some numerical methods lend themselves to simple performance analysis

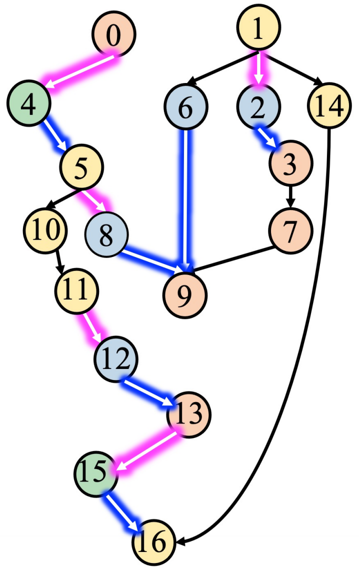
- DAG-based SpTRSV demands more sophistication

- Solution:
  - construct a critical path performance model
  - assess our observed performance relative to machine capabilities.

# Critical path Performance model

- **SpTRSV Characterization**
  - Initial Critical path: based on level-set using BFS
  - Refined Critical path: process decomposition

Inter-process column broadcast
Inter-process row reduction
Intra-process execution order



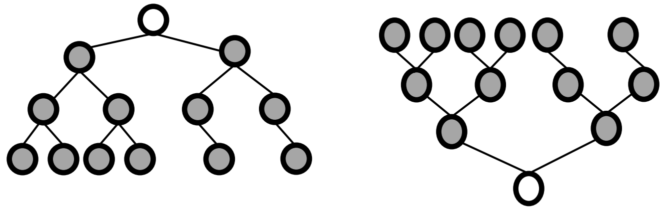

- **Architecture Characterization**
  - *Memory bandwidth* scales with the number of blocks (GEMV/TRSV) in the same level until the aggregate bandwidth reach the peak:

$$T_{mat-vec\ per\ gpu} = \frac{accumulated\ Bytes}{aggrated\ bw}$$

  - *Communication:* binary communication tree, latency-bandwidth model

$$T_{comm\ per\ gpu} = \sum_{\text{levels}} \left( \mathrm{L}_{\text{net}} + \frac{\log2(\#\text{out}) * \text{sz}}{\mathrm{BW}_{\text{net}}} \right) + \sum_{levels} \left( log2(\#in) * (L_{net} + \frac{sz}{BW_{net}}) \right)$$

**broadcast** **reduction**

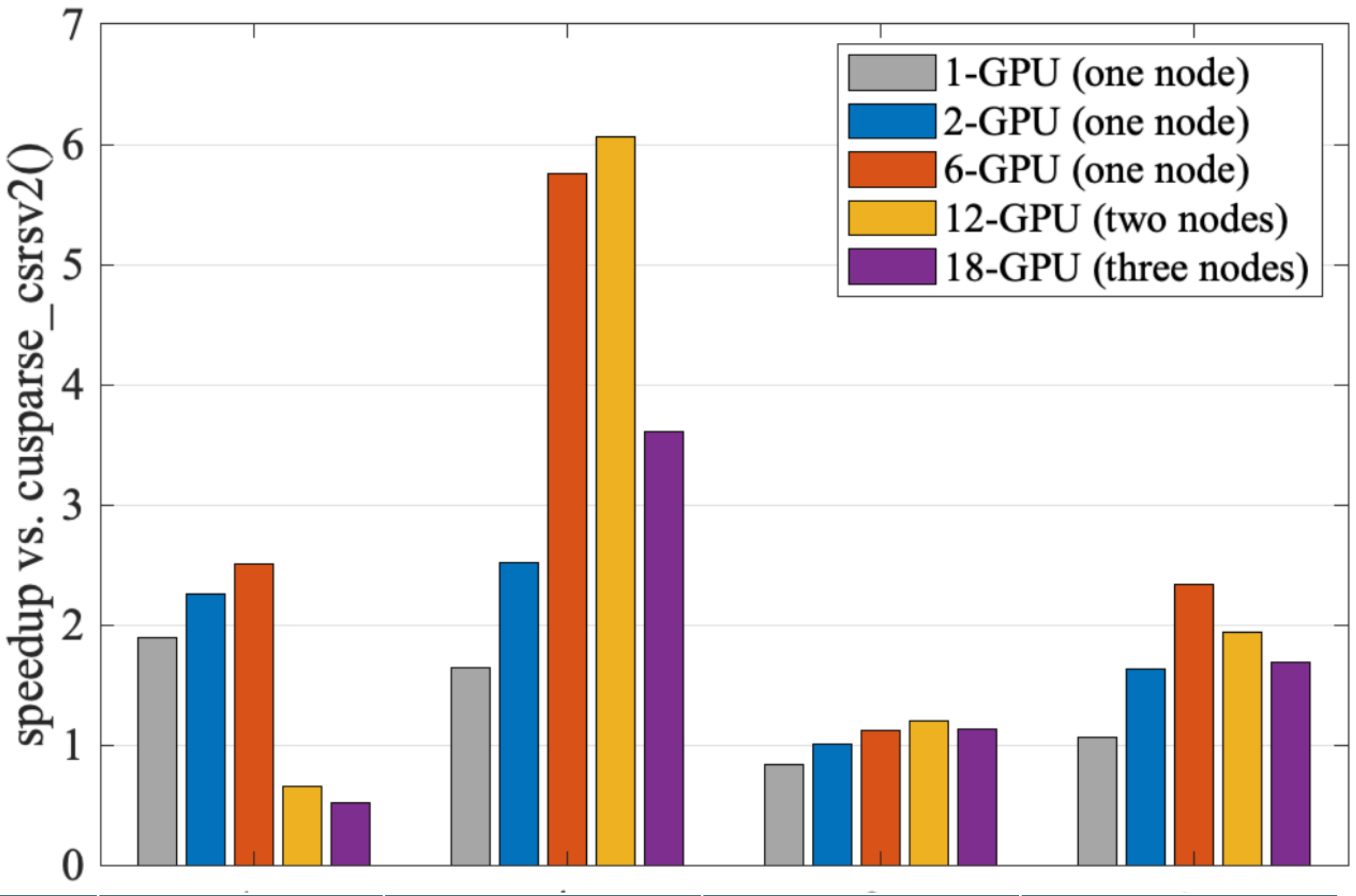


off-diag diag

# Large number of messages of DG makes its scaling performance worse than matrix Li.

Interesting Observations:

- DG has a similar number of DAG levels with Li but more nonzeros -> DG scales better than Li but it's not




@ 6 GPU (single node)

Li:

- 270 messages on the critical path

DG:

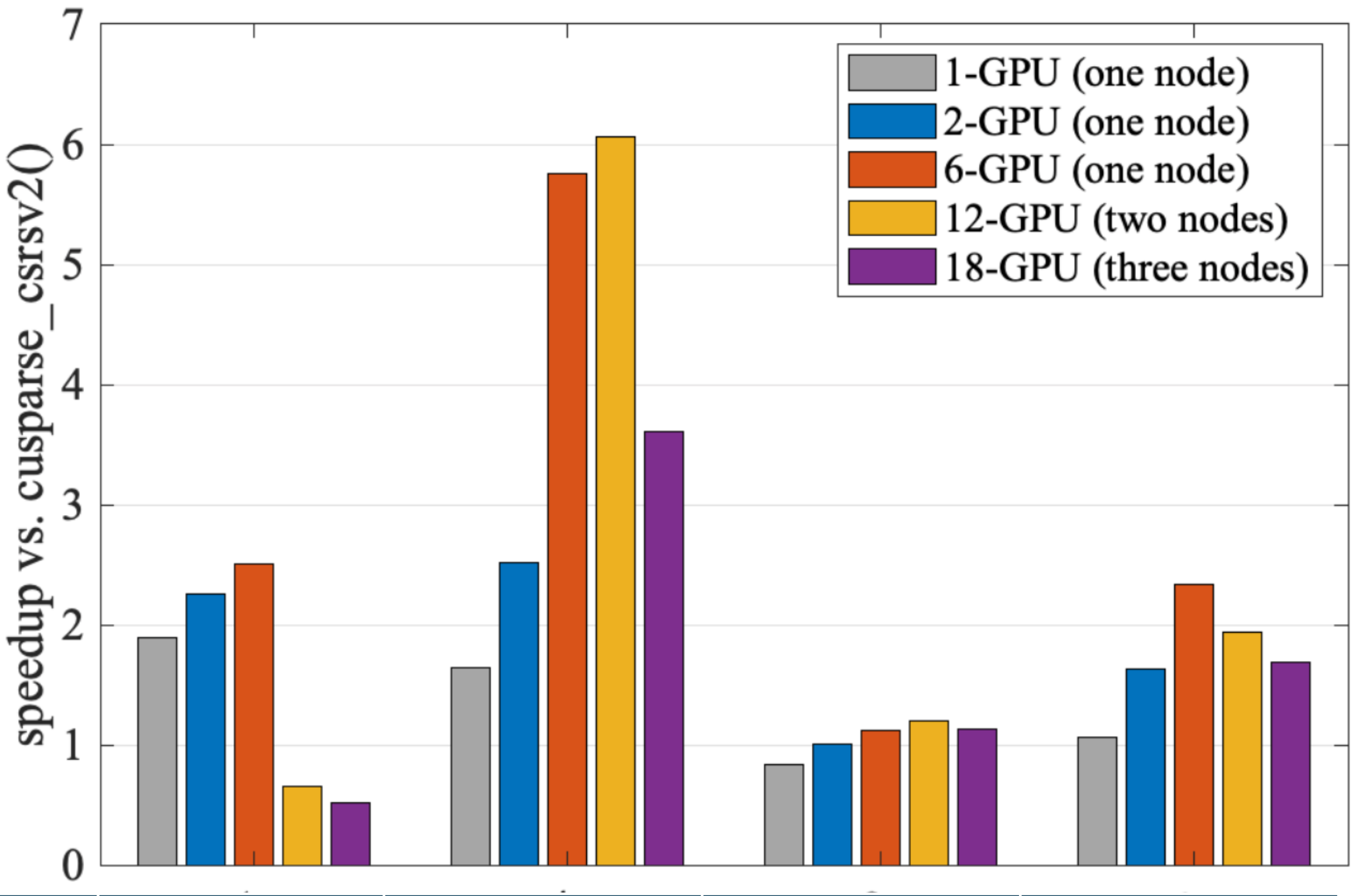
- 1000 messages on the critical path

| | S1 | Li | DG | LU |
|---|---|---|---|---|
| nnz | 8.80e+08 | 5.18e+08 | 9.66e+08 | 8.54e+08 |
| DAG levels | 388 | 188 | 199 | 264 |

# SpTRSV performance differs with critical paths

Interesting Observations:

- Exploit multiple GPUs on one node, performance is challenged when using multi-nodes




| | S1 | Li | DG | LU |
|---|---|---|---|---|
| nnz | 8.80e+08 | 5.18e+08 | 9.66e+08 | 8.54e+08 |
| DAG levels | 388 | 188 | 199 | 264 |

# SpTRSV performance differs with critical paths

@ 6 GPU (single node)

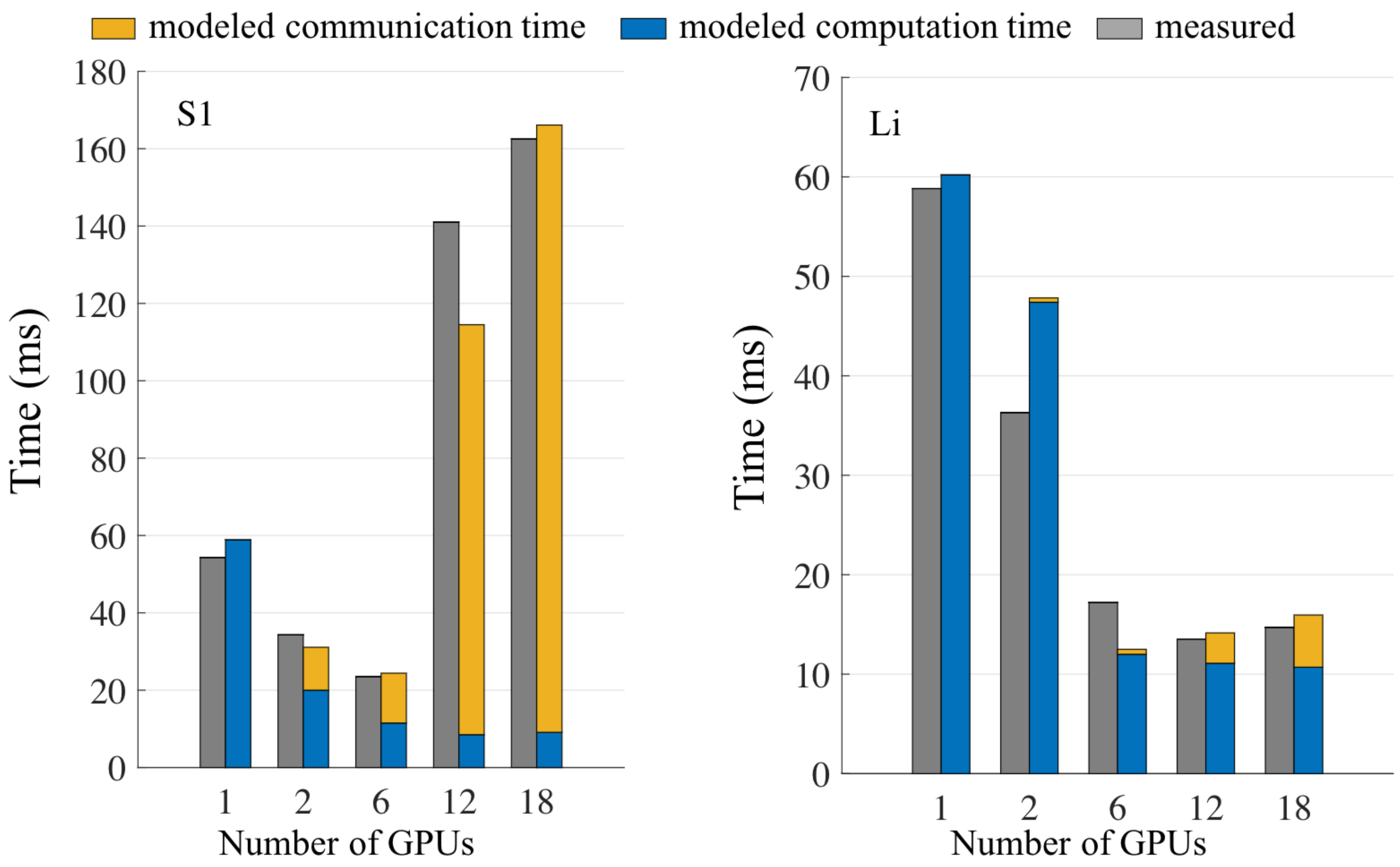
S1:

- 7,922 messages on the critical path
- 1.3 GB/s memory bandwidth

Li:

- 270 messages on the critical path
- 5.2 GB/s memory bandwidth

| Matrix | #supernodes | DAG levels | nnz L |
| --- | --- | --- | --- |
| s1_mat_0_507744 | 9,827 | 388 | 8.80E+08 |
| Li4244 | 362 | 188 | 5.18E+08 |



# Questions

U.S. DEPARTMENT OF ENERGY

UNIVERSITY OF CALIFORNIA

BERKELEY LAB
Lawrence Berkeley National Laboratory