Evaluating the Performance of One-sided Communication on CPUs and GPUs

Nan Ding, Muhammad Haseeb, Taylor Groves, Samuel Williams
Lawrence Berkeley National Laboratory
nanding@lbl.gov

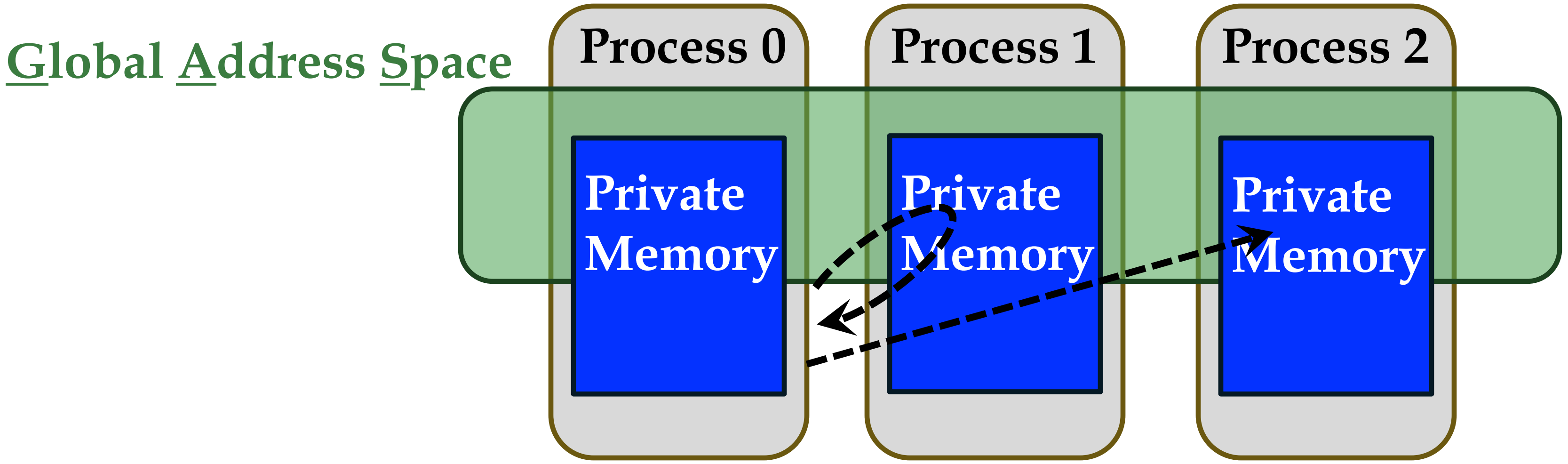
Outline

- Benefit and Challenges
- Message Roofline Model
- Results

What is One-Sided MPI?

- Two-Sided MPI: both sender and receiver are involved in data transfer
  - Example :MPI_Send/MPI_Recv

- One-Sided MPI: decouple data movement with process synchronization
  - PGAS model: one process can directly access other processes' memory
  - move data without requiring that the remote process synchronize
  - Example: MPI_Put



Benefits and Challenges

- Common way to communication on multiple GPUs

CPU (control flow) | GPU

Loop:

    <<<…>>> do computation
    ** synchronization**
    ** do communication **

- **Increased algorithm complexity and decreased program productivity**
- **Hard to scale DAG-like computation**

- GPU-initiated communication (One-Sided): NVSHMEM/ROC_SHMEM

CPU | GPU (control flow)

<<<…>>> everything

Loop:

    ** do computation **
    ** do communication **

- **Program like on traditional CPU nodes**
- **Makes scaling DAG-like computation more feasible**
- **Preserve portability by using a common SHMEM interface that could be applied to CPUs and GPUs**
- **Highlights the use of one-sided communication on CPUs**

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

Benefits and Challenges

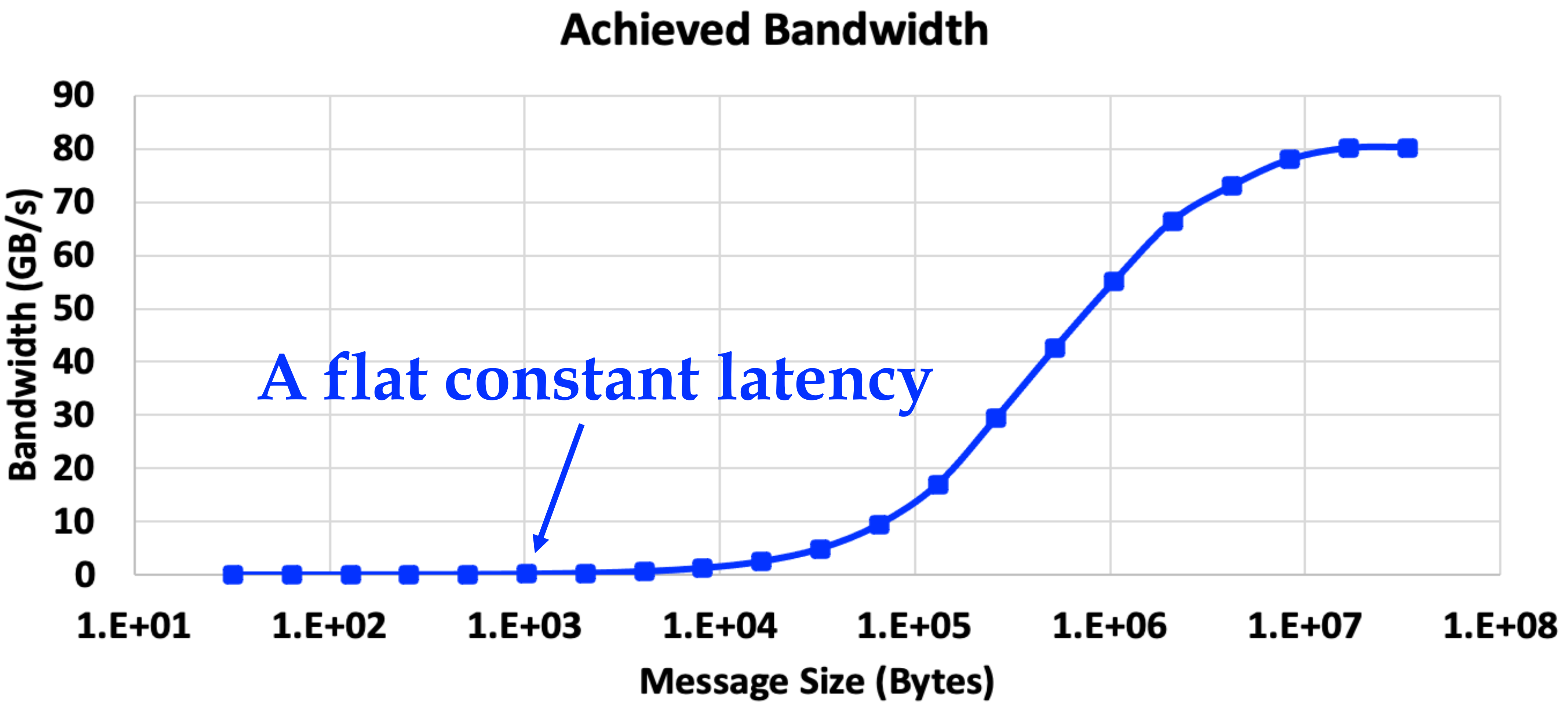
- Challenges:
  - Requires more careful management of data placement and synchronization
    - Two-Sided communication: MPI_Recv handles everything
      - Data transfer is complete at the receiver side
      - Receive buffer can be easily re-usable
    - One-Sided communication: NA
      - Need user effort to manage data placement and receiver notification

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

What’s the Achieved Communication Performance?

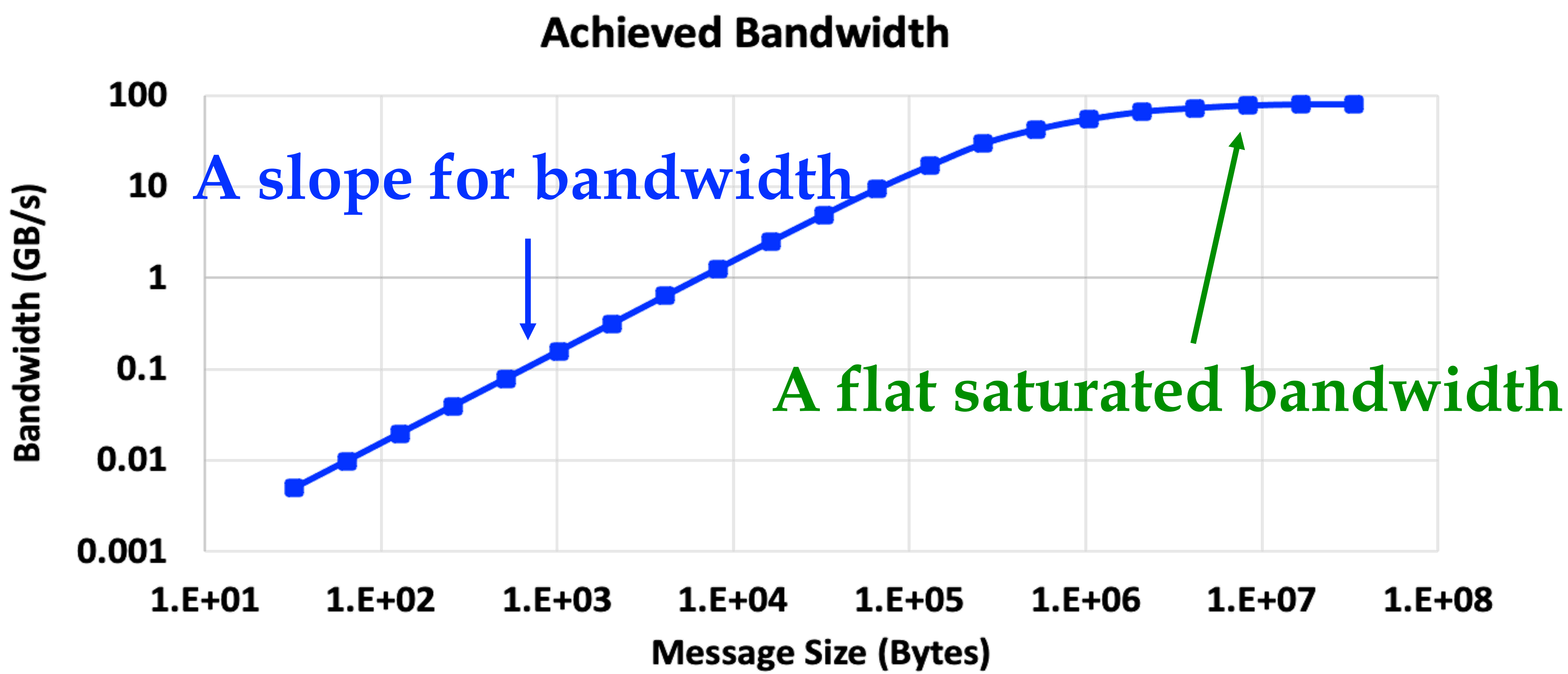
- Message Roofline Model provides a realistic bound on the communication performance based on the number of messages per synchronization

Log-linear Plot: CAN NOT Interpret Small Message Performance

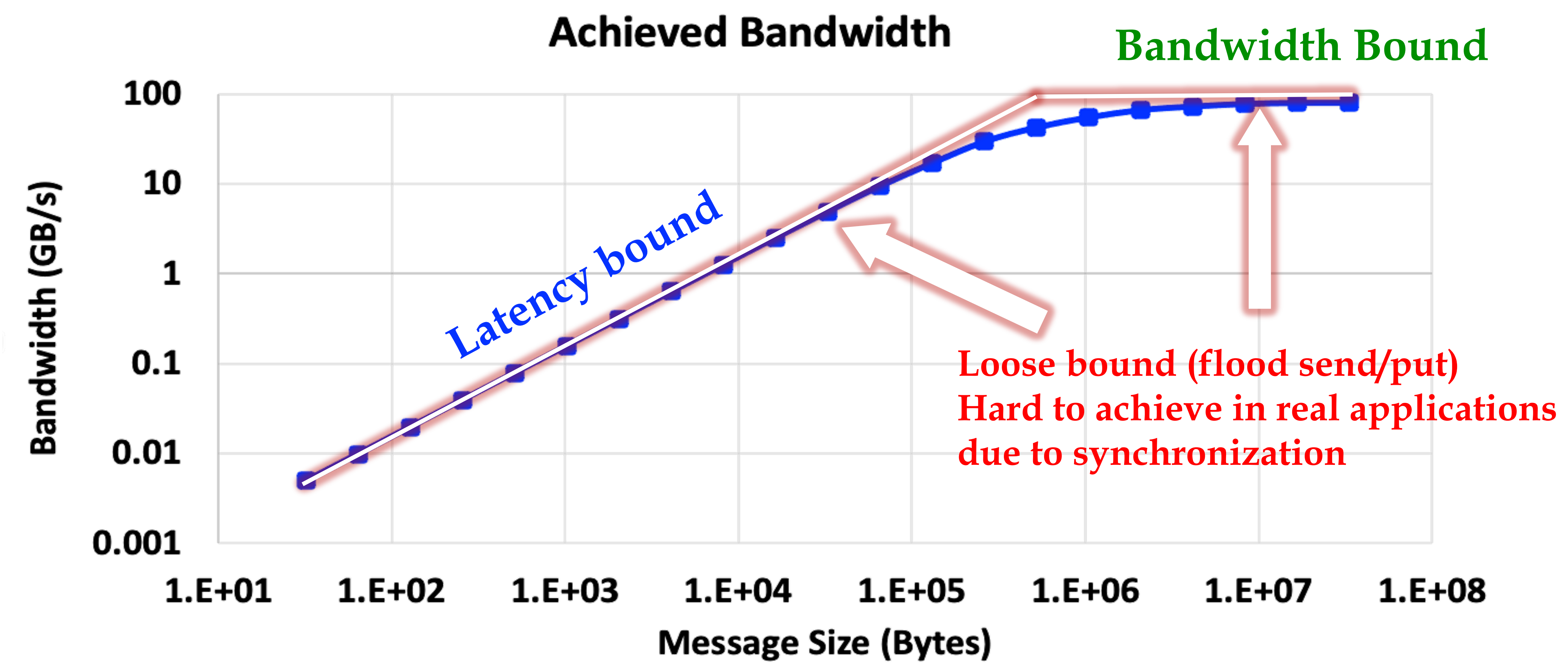


Log-Log Plot: CAN Interpret Small Message Performance

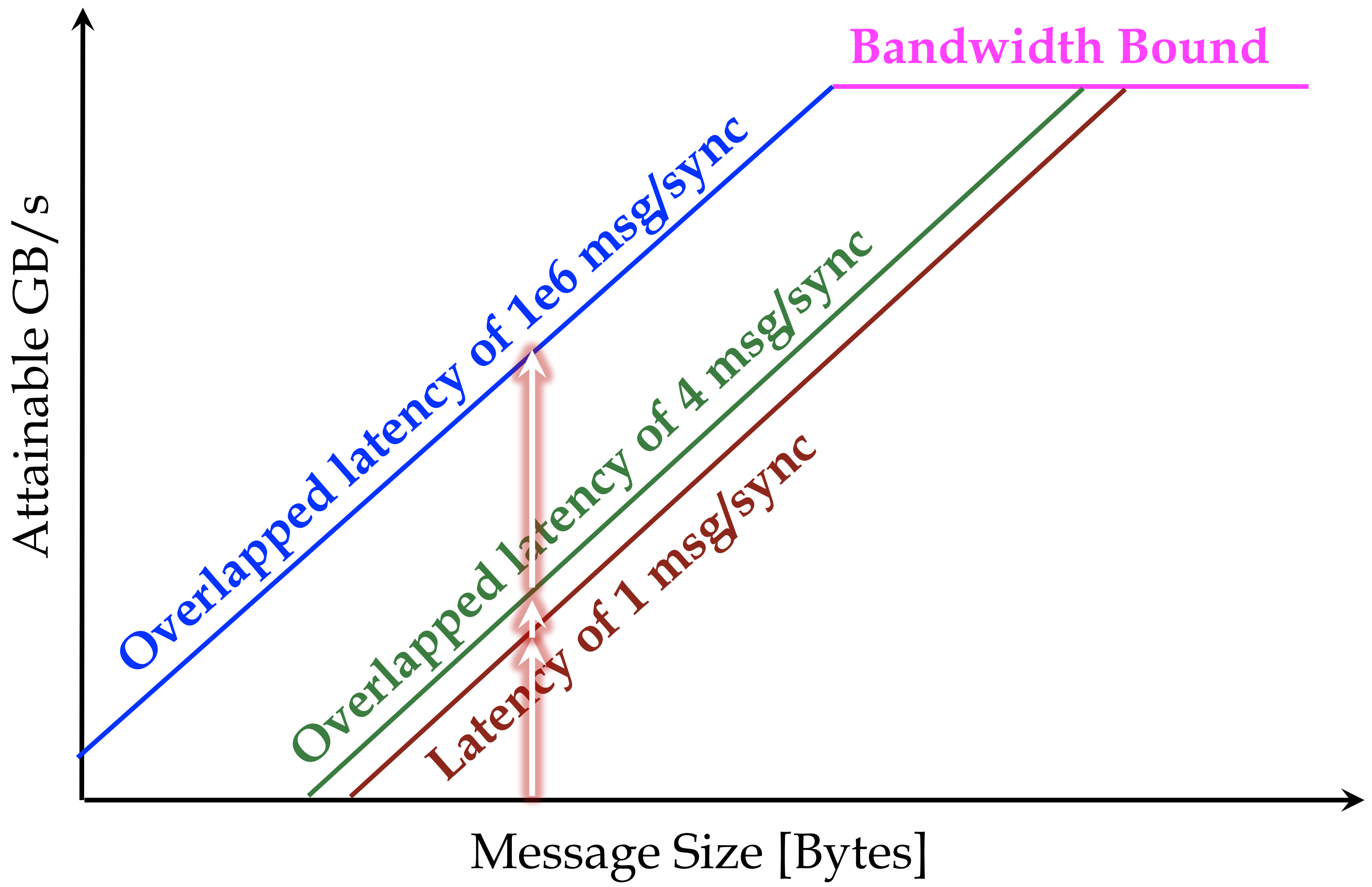
Achieved Bandwidth = F(message size)



Can you achieve the peak?

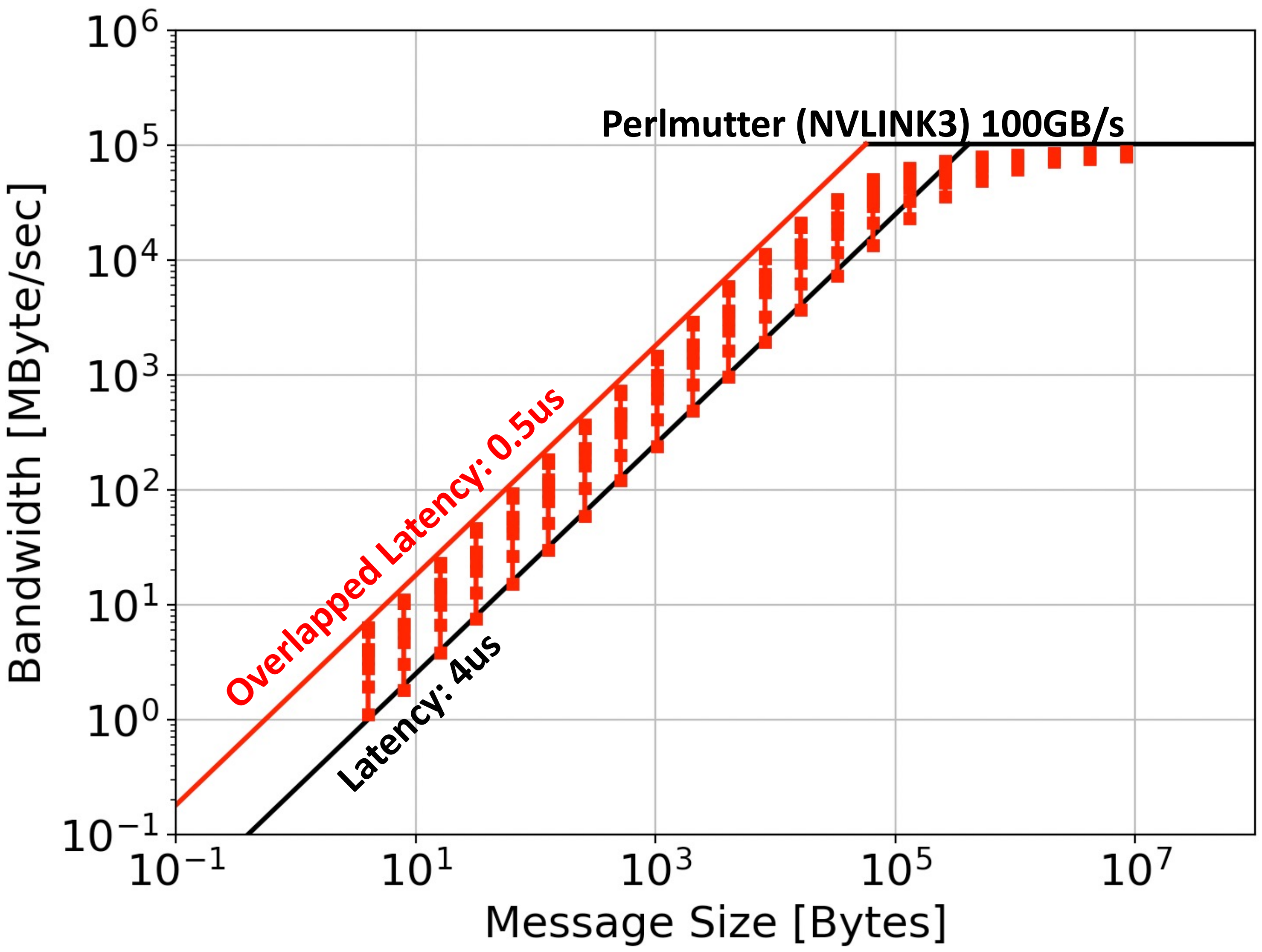


Msg/sync Tells A Tight Communication Upper Bound



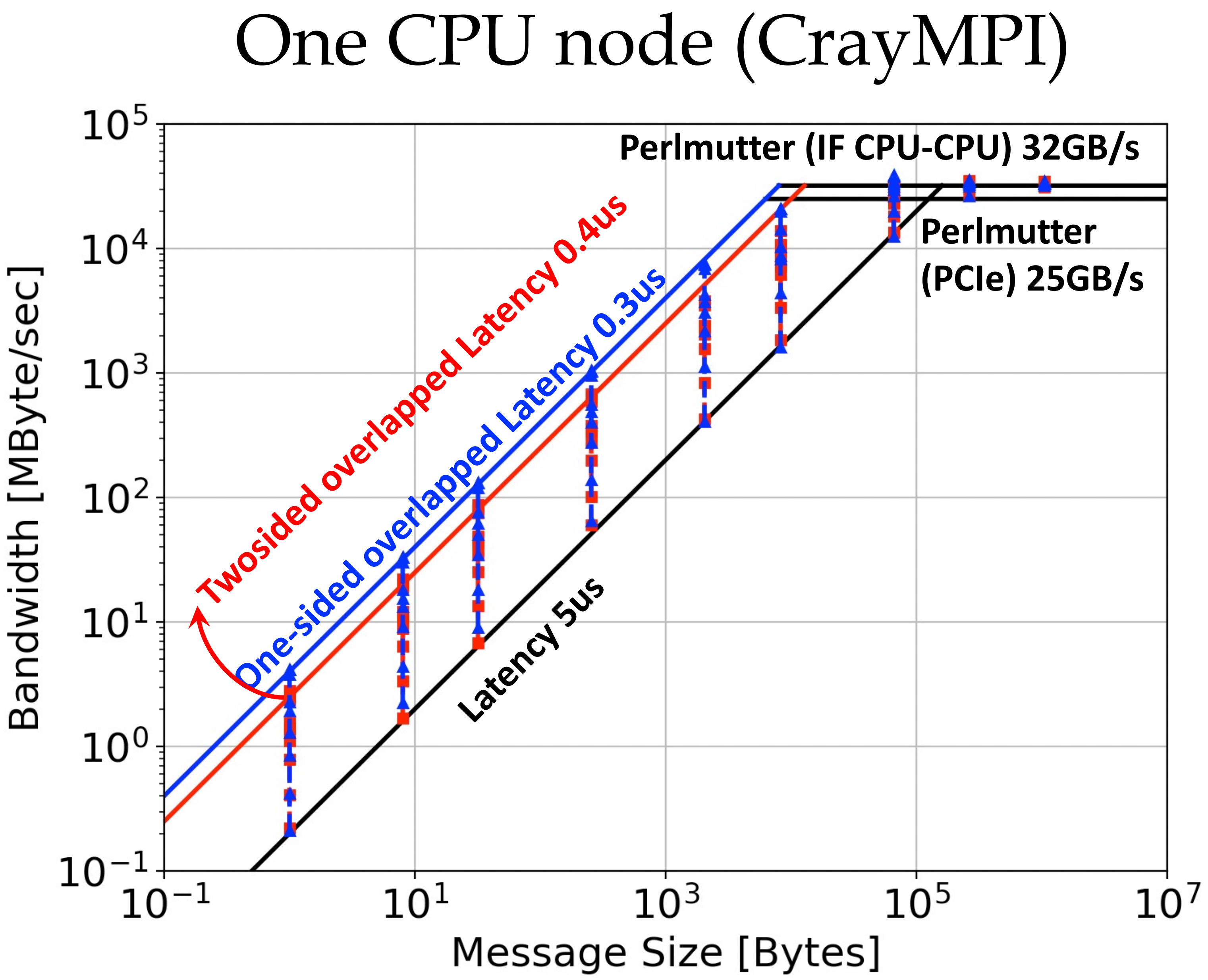
Communication performance on Perlmutter GPUs

One GPU node (NVSHMEM)



- Sender: put-with-signal and nvshmem_quiet to ensure the data transfer is completed at the receiver side.

Communication performance on Perlmutter CPUs



Two-Sided:

- MPI_Isend
- MPI_Recv

One-Sided:

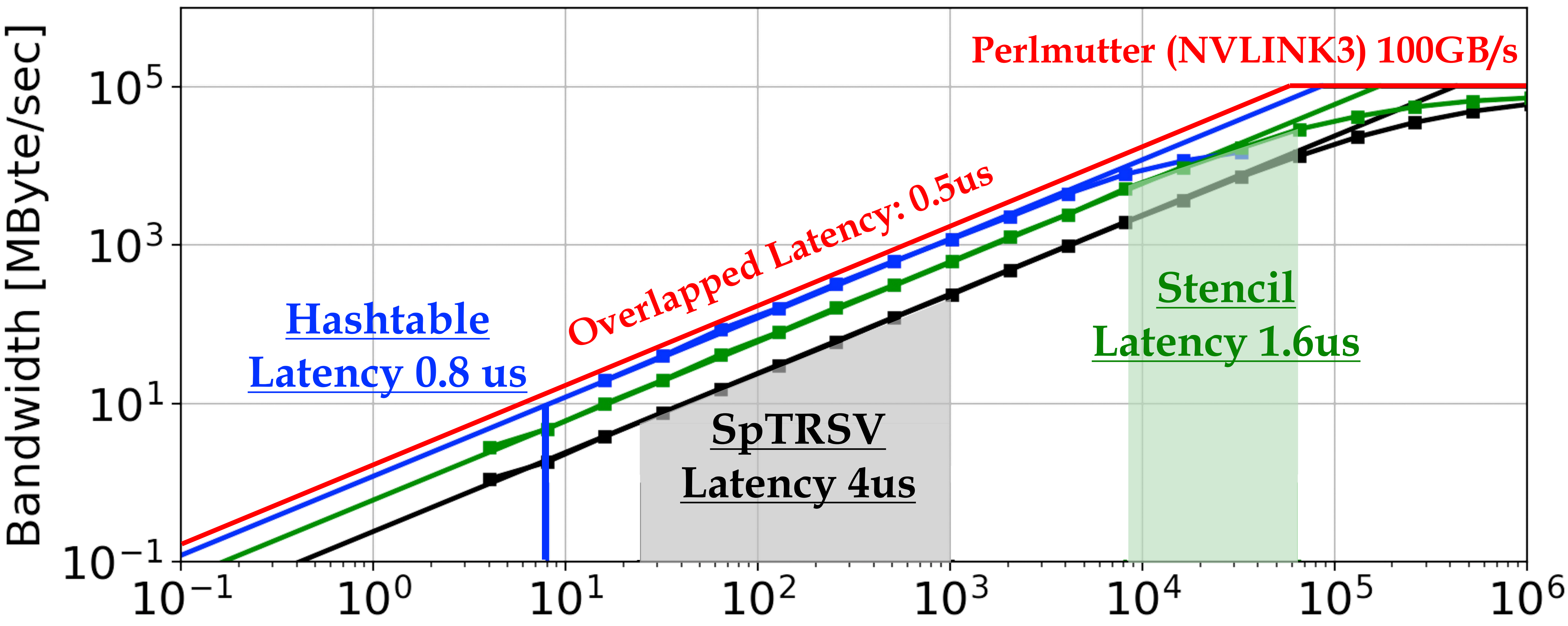
- MPI_Put (data)
- MPI_Win_flush /* memory order */
- MPI_Put (signal)
- MPI_Win_flush /* avoid a delayed signal */

CPU one-sided MPI has potential to outperform the two-sided by supporting put-with-signal and receiver notification operations.

Characterize Applications using msg/sync

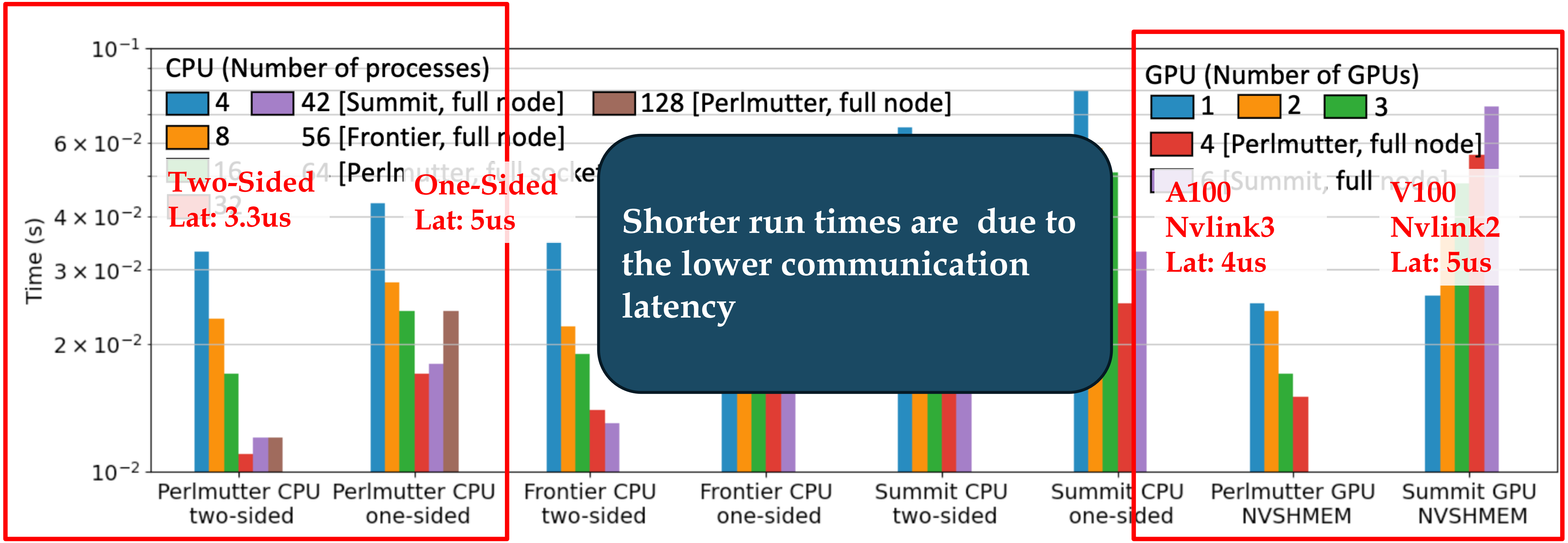
Workloads	Patterns	Need receiver Notify?	P2P pair	Msg/sync	Words/Msg
2D Stencil	BSP sync	Yes	Deterministic & fixed	4	Problem size/P
SpTRSV	DAG async	Yes	Deterministic & variable	1	Avg. 100
HashTable	Random async	No	indeterministic	Two-Sided: P	3
				One-Sided: 1e6	1

Varies Achieved Bandwidth due to different msg/sync



Case Study: SpTRSV

- Matrix (from M3D-C1): 126K x 126K, with 1E+8 non-zeros
- 1 msg/sync
- Message size: 24 bytes – 1040 bytes



Conclusion

- Propose a new metric -- the number of messages per synchronization -- to provide a tight upper bound of communication performance, and help reason performance

- Message Roofline Model can help with 3P: ***(1)Performance***: provide a tight upper bound of communication performance, ***(2) Productivity***: guide a proper communication model for applications, and ***(3) Portability (Performance):*** reason different performance trends across architectures.

- We demonstrate the potential of One-Sided MPI if put-with-signal and loose wait can be supported on CPUs.

Acknowledgements

This material is based upon work supported by the Advanced Scientific Computing Research Program in the U.S. Department of Energy, Office of Science, under Award Number DE-AC02-05CH11231 and used resources of the National Energy Research Scientific Computing Center (NERSC) which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research also used resources of the Oak Ridge Leadership Facility which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. This work was completed in part at the NERSC Open Hackathon, part of the Open Hackathons program. The authors would like to acknowledge OpenACC-Standard.org for their support.

U.S. DEPARTMENT OF ENERGY

UNIVERSITY OF CALIFORNIA

BERKELEY LAB
Lawrence Berkeley National Laboratory