OAK RIDGE National Laboratory

Oak Ridge National Laboratory

An Initial Assessment of NVSHMEM for High Performance Computing

Chung-Hsing Hsu, Neena Imam
{hsuc,imamn}@ornl.gov

Akhil Langer, Sreeram Potluri, Chris J. Newburn
{alanger,spotluri,cnewburn}@nvidia.com

ORNL is managed by UT-Battelle, LLC
for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

APDCM'20
May 18, 2020

One-Page Summary

- NVSHMEM is an **experimental** programming library:
  - Developed by NVIDIA.
  - Alternative to the popular CUDA+MPI approach.
  - Enabling GPU-initiated data communication.
  - Supporting SHMEM for NVIDIA GPU clusters.
- We focus on evaluating NVSHMEM **at scale**:
  - NVIDIA incapable of doing it in house.
  - Testing NVSHMEM on ORNL's Summit supercomputer.
  - Eyeing on usability, functionality, and scalability.
  - Using math kernels at various optimization levels.
  - Helping make NVSHMEM a viable option for HPC.

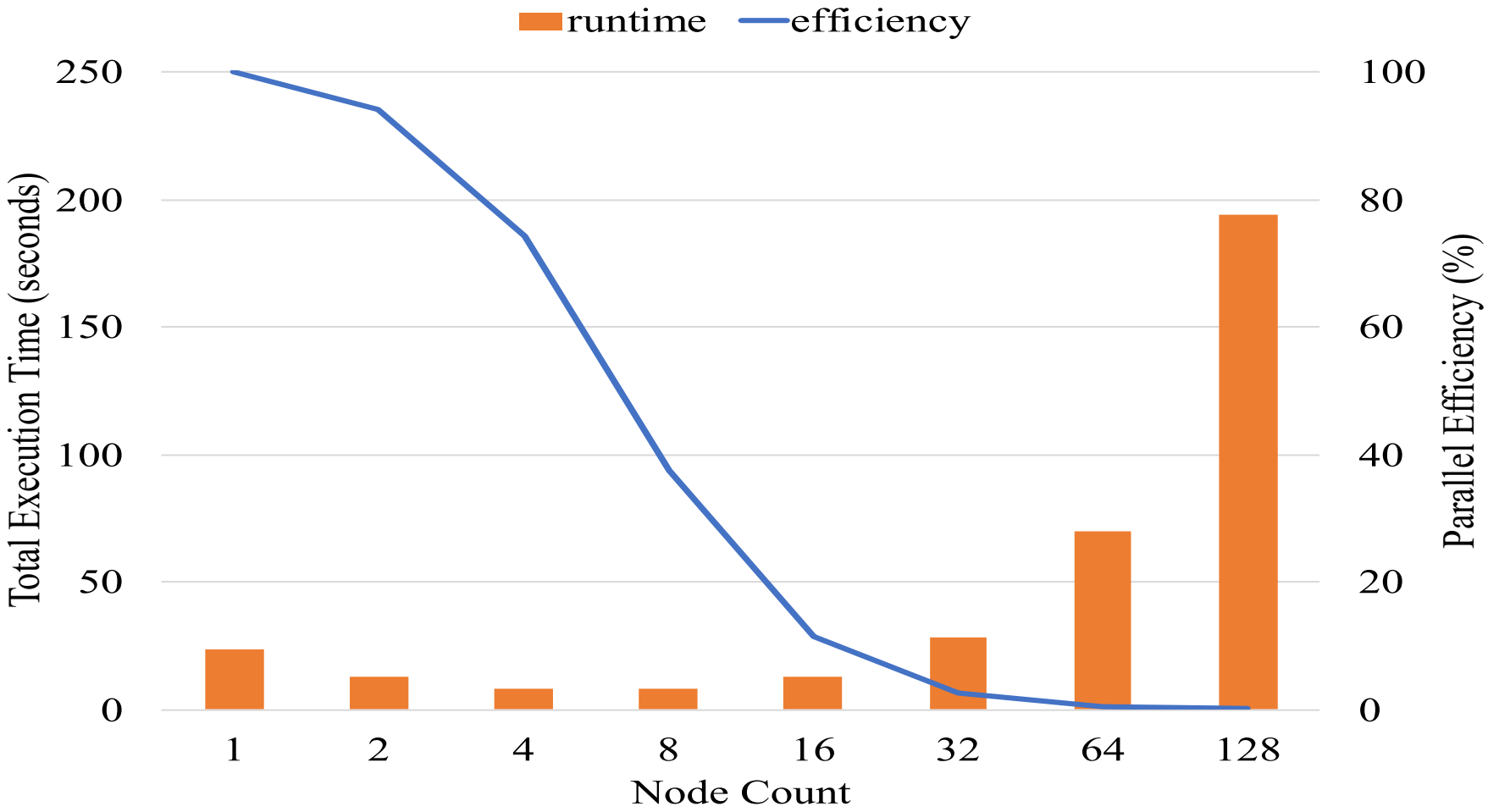
Evaluating Methodology

- Application workload:
  - matrix-matrix multiplication: direct SHMEM port.
  - Jacobi solver: highly optimized.
- Target platform:
  - A Supercomputer with 4,602 compute nodes.
  - 6 NVIDIA V100 GPUs and 2 IBM Power9 CPUs per node.
- Evaluation criteria:
  - Usability: Complexity of writing a NVSHMEM code.
  - Functionality: Robustness of NVSHMEM itself.
  - Scalability: Performance improvement with more GPUs.

Case Study: Matrix Multiplication

- Code unoptimized and porting straightforward.
- Bug inherited from SHMEM code identified.
- Optimization non-trivial.
  - CUDA programming is non-trivial:
    - <<<1,1>>> is easy, but not <<<M,N>>>. Also corner cases.
  - NVSHMEM adds complexity:
    - Per-GPU data size different, but same symmetric memory size.
    - A rich set of extended API, but lacking examples.
- Limited strong scaling.
  - Improved for larger problem sizes.
- Performance portability not guaranteed.

Limited Strong Scaling

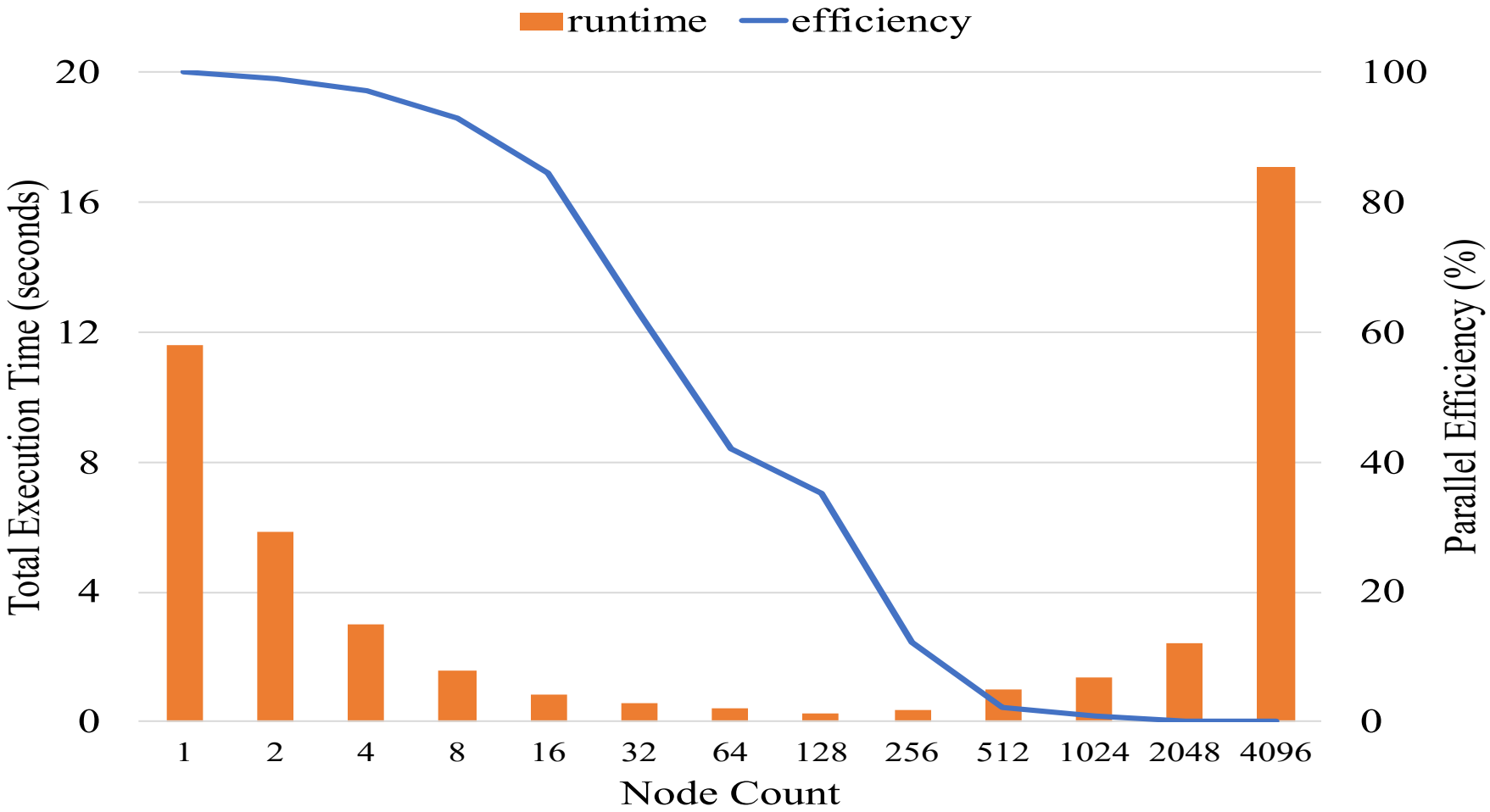


Code: matrix multiplication. Overall problem size: 1024 × 1024

Case Study: Jacobi Solver

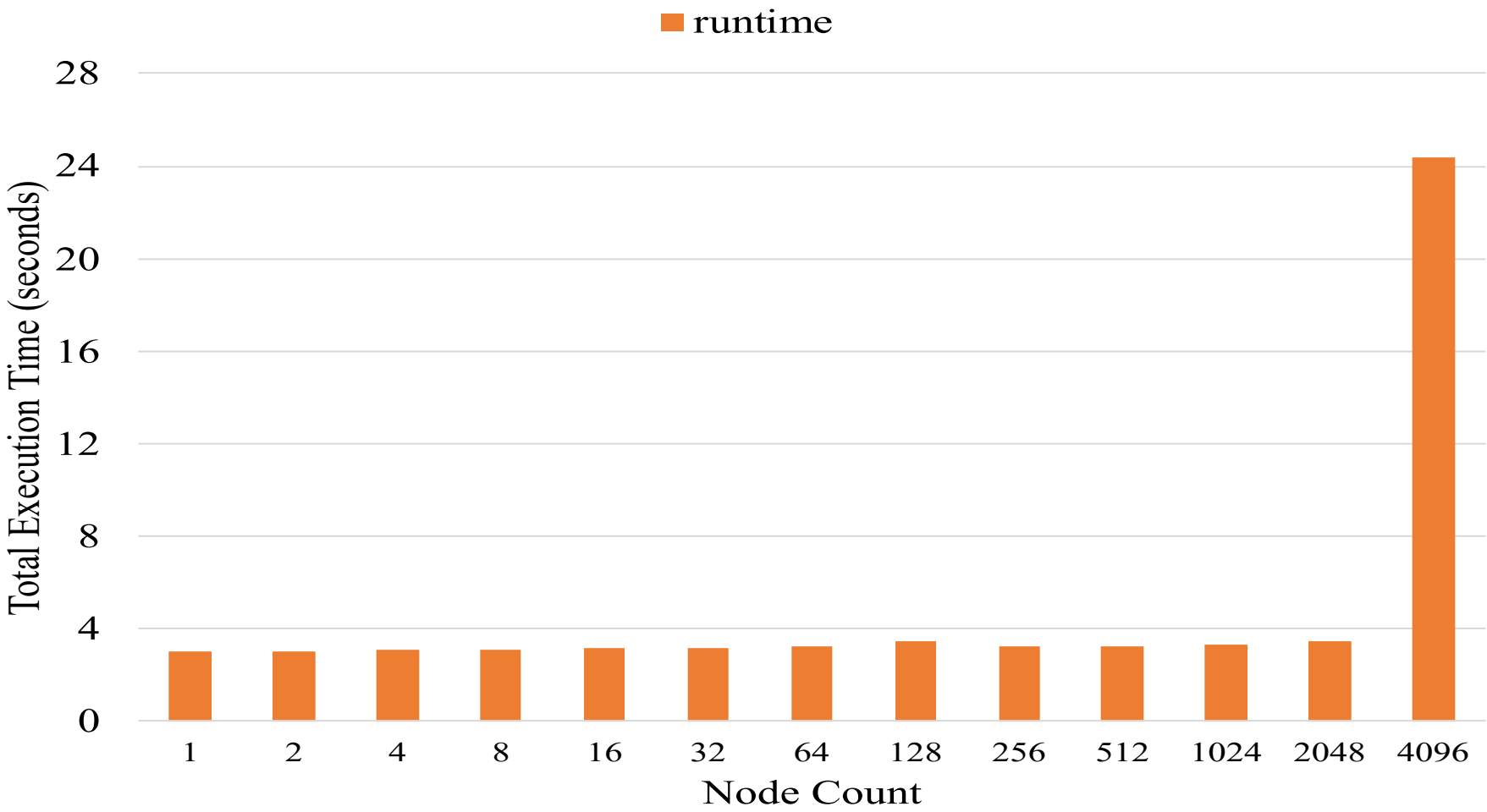
- Code highly optimized and a bit complex.
- The tested library is buggy.
  - Code crashes with no warnings at 24K GPUs.
  - Code hangs running a less-optimized version.
- Not all issues are from the library.
  - Error in configuring CUDA kernel to run.
  - Logic flaw uncaught until we modified the code.
- Limited strong scaling.
  - But near-optimal weak scaling.
- Performance better than CUDA+MPI.
  - But unexplained anomalies at the largest scale.

Limited Strong Scaling



Code: Jacobi solver. Overall problem size: 32768 × 32768

Near-Optimal Weak Scaling



Code; Jacobi solver. Per-node problem size: 32768 × 8192

Conclusions

- Writing a correct NVSHMEM code can be non-trivial.
  - Requires good knowledge of CUDA and SHMEM.
  - Code may hang; Bugs may manifest only at largest scale.
- Writing an efficient NVSHMEM code can be non-trivial.
  - Direct porting does not guarantee performance portability.
  - Direct use of API does not guarantee best performance.
  - Requires expertise in CUDA and NVSHMEM extended API.
- NVSHMEM has potentials and will improve over time.
  - We have helped improving its functionality & performance.

Acknowledgements

This work was supported by the United States Department of Defense (DoD) and used resources of the Oak Ridge Leadership Computing Facility at Oak Ridge National Laboratory.