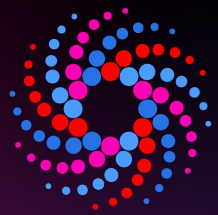


SC22
Dallas, TX | hpc accelerates.

Extending OpenMP and OpenSHMEM for Efficient Heterogeneous Computing

Wenbin Lu, Shilei Tian, Tony Curtis, and Barbara Chapman

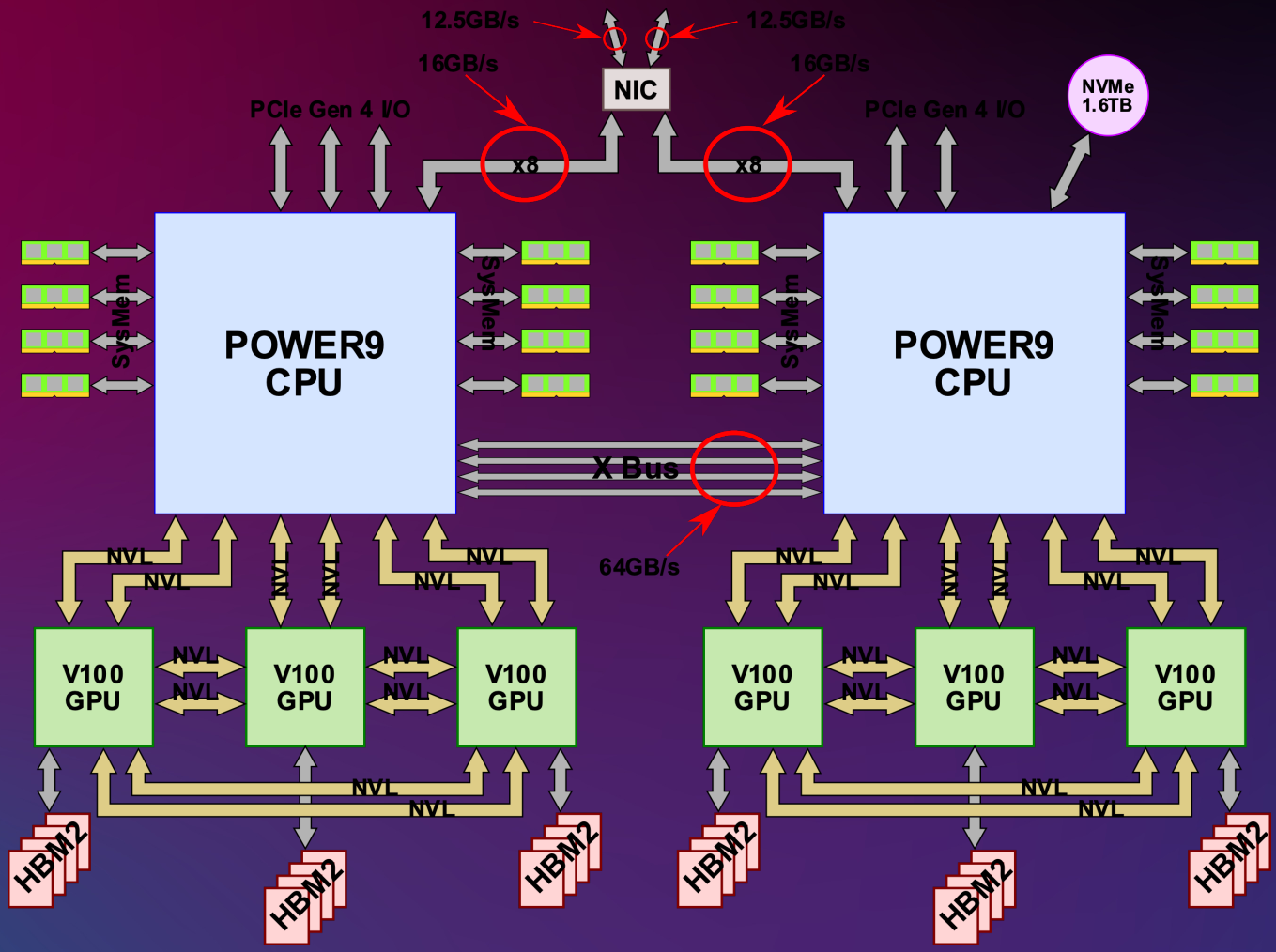
Institute for Advanced Computational Science

Stony Brook University

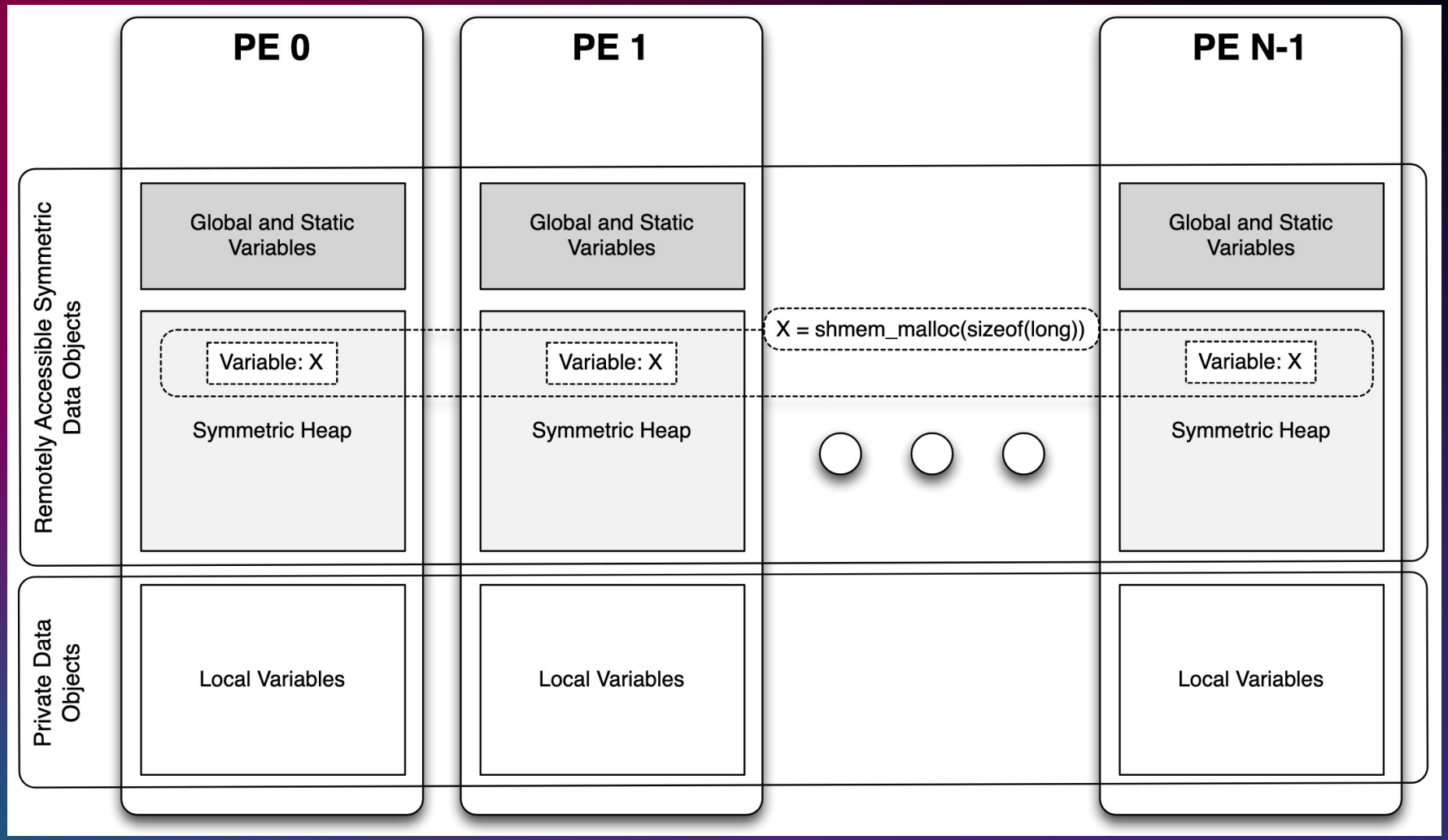
Parallel Applications Workshop, Alternatives To MPI+X

Nov/14/2022

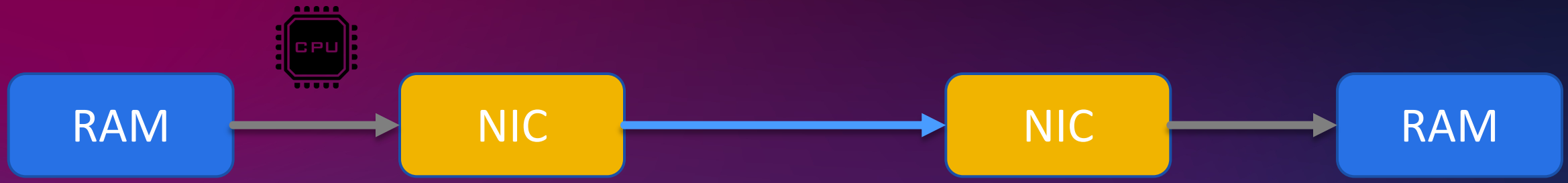
Heterogeneous Exascale Supercomputers



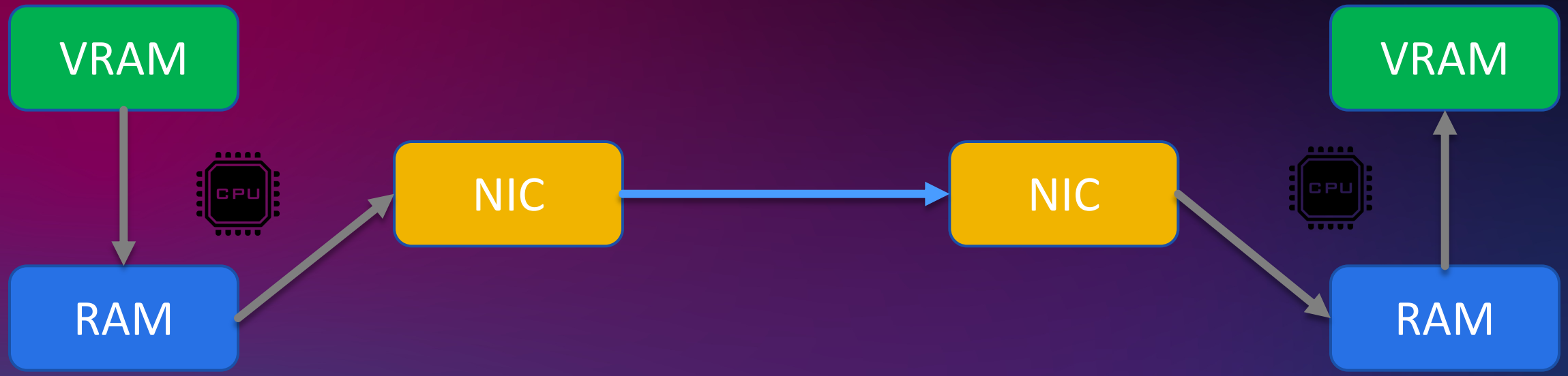
OpenSHMEM for Inter-Node Communication



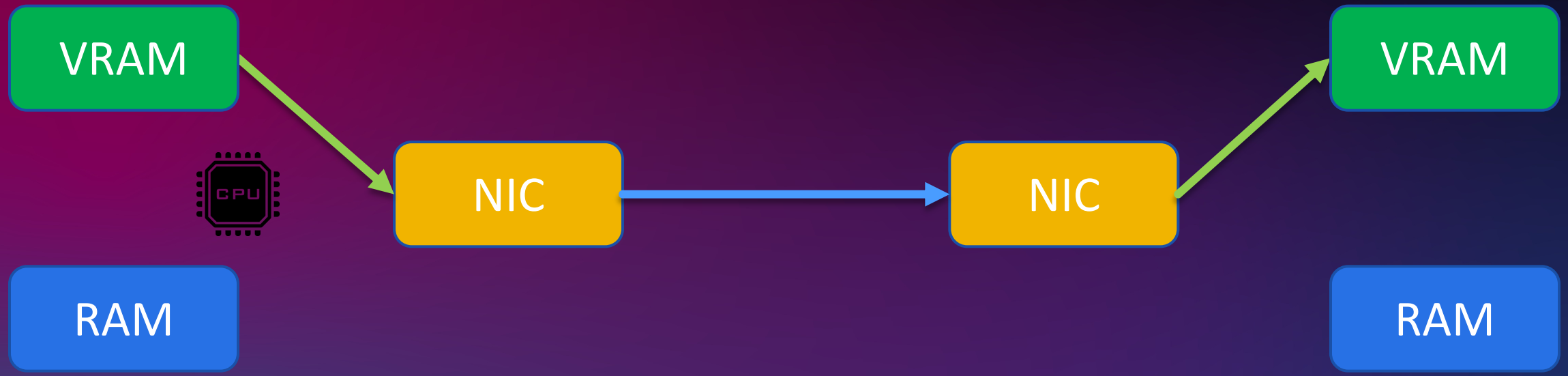
OpenSHMEM for Inter-Node Communication



OpenSHMEM for Inter-Node Communication (GPUs)



OpenSHMEM for Inter-Node Communication (GPUs)



OpenMP Device Offloading for GPU computation

OpenMP Device Offloading DAXPY Example

```
double* a = new double[N];
double* b = new double[N];

#pragma omp target data map(to: b[0:N]) map(tofrom: a[0:N])
{
    #pragma omp target teams distribute parallel for
    for (int i = 0; i < N; i++) {
        a[i] += 1.414 * b[i];
    }
}
```

Hybrid OpenSHMEM + OpenMP (PUT & Reduction)

```
#pragma omp target data map(tofrom:X[0:M]) map(from:U[0:N], V[0:N])
{
    // PUT Source PE
    #pragma omp target update from(X[0:M])
    shmem_float_put_signal(X, X, ...);

    // PUT Target PE
    shmem_signal_wait_until(...);
    #pragma omp target update to(X[0:M])

    // Vector Reduction
    #pragma omp target update from(V[0:N])
    shmem_float_sum_to_all(U, V, ...);
    #pragma omp target update to(U[0:N])
}
```

Hybrid OpenSHMEM + OpenMP (Read-Modify-Write)

```
// Read-Modify-Write Initiator
shmem_atomic_set(sig_g, ...);          // Send 'Ready to GET'
while (!sig_p) {};                     // Wait for 'Data PUT'
foo(...);                              // Modify
shmem_put_signal(sig_u, ...);          // Send 'Data Updated'

// Read-Modify-Write Polling Thread
terminate(sig_t);                      // Check for termination
check(sig_g, ...);                     // Check for 'Ready to GET'
#pragma omp target update from(...)    // Read from the GPU
shmem_put_signal(sig_p, ...);          // Send 'Data PUT'
while (!sig_u) {};                     // Wait for 'Data Updated'
#pragma omp target update to(...)      // Write to GPU
```

Poor Interoperability Leads to …

- Poor maintainability
  - Sophisticated signaling mechanisms for RMA operations
  - Breaks the one-sidedness of OpenSHMEM (why not just use MPI)
- Poor Performance
  - Extra memory copy operations
  - Dedicated polling thread / check the signal variables periodically

Idea: OpenSHMEM as an OpenMP Allocator

- At the core:
  - OpenMP provides automatic device memory management through its `map(…)` clause
  - OpenSHMEM manages the (device) symmetric heap through its C API

- Improve the interoperability between the two by using OpenSHMEM as OpenMP's allocator
  - This can be achieved through API + runtime extensions
  - For the users: write **less code**, attain **better performance**

OpenSHMEM Extension: Symmetric Partitions

```
// Defining symmetric partitions
// export SHMEM_SYMMETRIC_PARTITION<ID>=SIZE=<size>
//                                      [:PGSIZE=<pgsize>]
//                                      [:KIND=<kind>]
//
// Symmetric partition memory management routine
void* shmem_partition_malloc(size_t size, int id)
void* shmem_partition_free(void* ptr, int id)
```

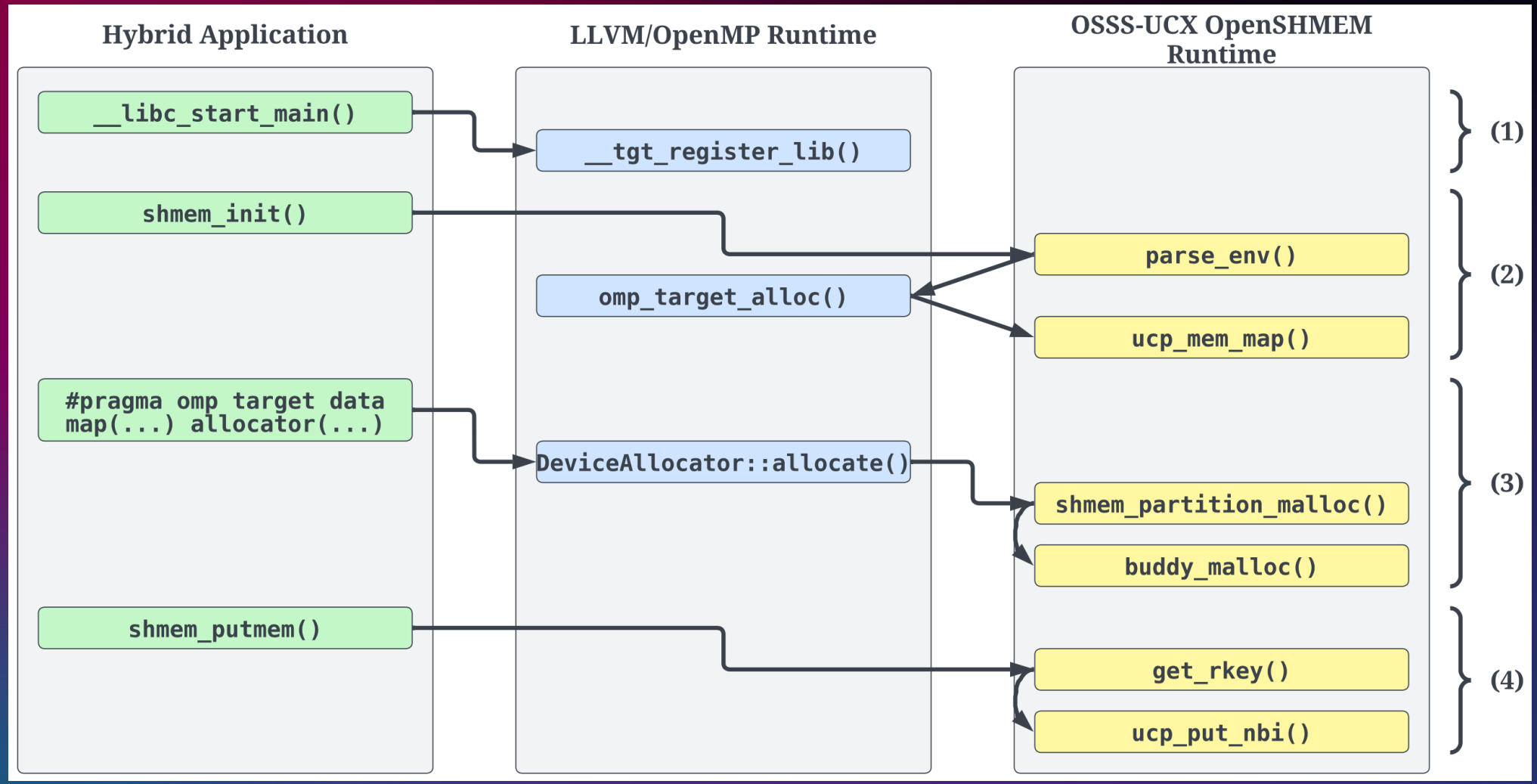
OpenSHMEM Extension: Symmetric Partitions

- Symmetric partitions are allocated during `shmem_init()`
  - Targeting an extended LLVM/OpenMP runtime (supports CUDA and ROCm)
  - `export SHMEM_SYMMETRIC_PARTITION0="SIZE=2G:KIND=libomp"`
- OpenSHMEM implementation is based on OSSS-UCX
  - UCX handles GPU buffer registration (also supports CUDA and ROCm)
  - Buddy allocator
  - No need for vendor-specific code!

OpenMP Extension: SHMEM Allocator

```
__tgt_set_shmem_allocator(shmem_partition_malloc, shmem_partition_free, ...);
omp_allocator_handle_t shmema = ...;

#pragma omp target data map(tofrom:X[0:M]) map(from:U[0:N], V[0:N]) \
                        uses_allocators(shmema) allocator(shmema:X, U, V)
{
    // RMA is enabled for X[], U[] and V[]
}
```

Hybrid Runtime Systems Workflow



Putting it All Together

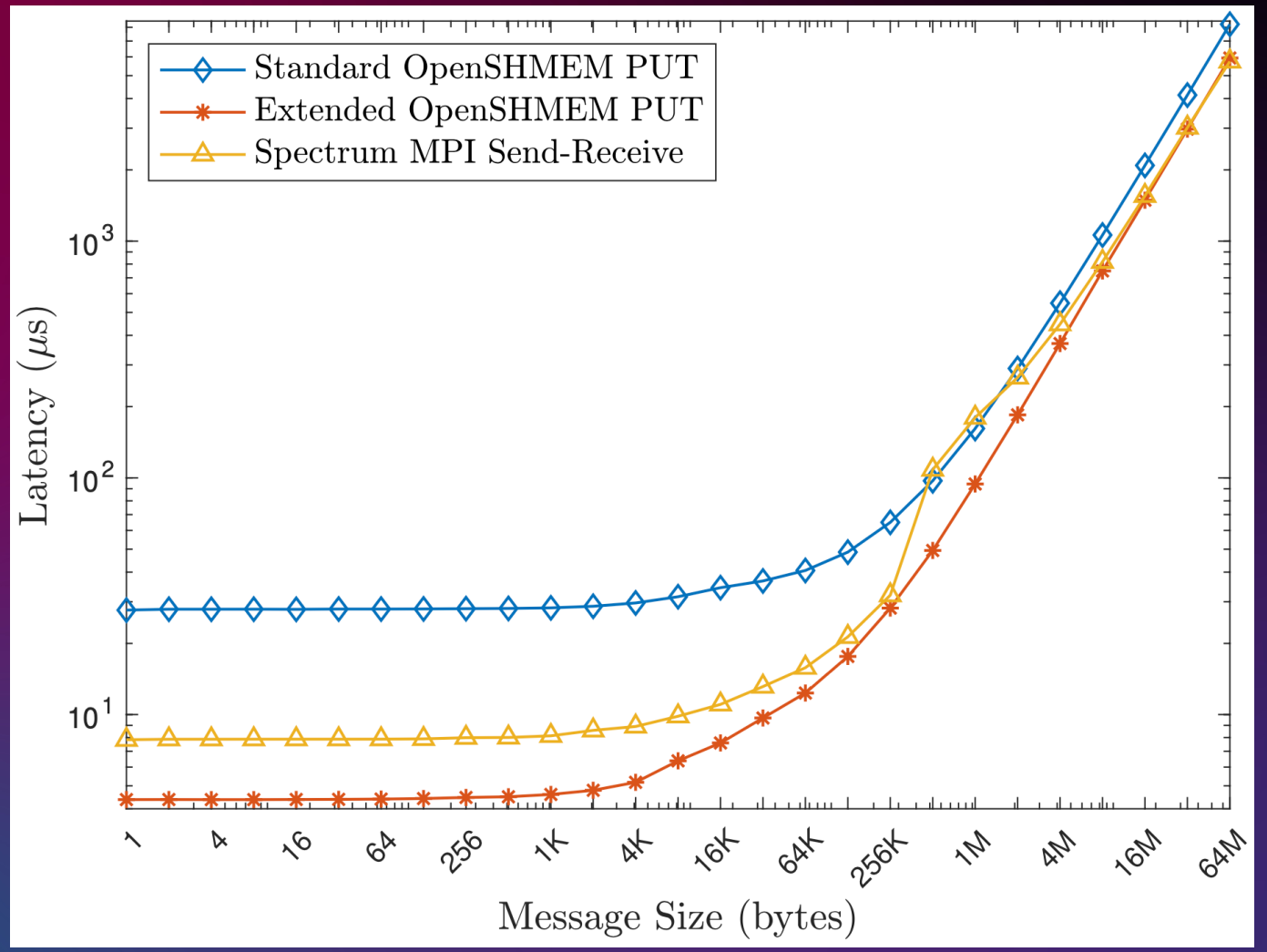
```
#pragma omp target data map(tofrom:X[0:M]) map(from:U[0:N], V[0:N]) \
                        uses_allocators(shmema) allocator(shmema:X, U, V)
{
    // PUT
    #pragma omp target data use_device_ptr(X)
    shmem_float_put(X, X, ...);

    // Vector Reduction
    #pragma omp target data use_device_ptr(U,V)
    shmem_float_sum_to_all(U, V, ...);

    // Read-Modify-Write
    #pragma omp target data use_device_ptr(table)
    {
        auto v = shmem_uint64_g(...);    // Read
        v = foo(v);                      // Modify
        shmem_uint64_p(...);             // Write
    }
}
```

Experiment Setup

- ORNL Summit
  - 2x21 IBM POWER9 cores, 2x3 NVIDIA V100 GPUs, Mellanox ConnectX-5 EDR 100Gb/s
  - RHEL 8.2, GCC 8.5.0, CUDA 11.0.3
  - MLNX OFED 4.9, GDRCopy 2.0
  - UCX 1.12, UCC 1.1 pre-release
  - LLVM 15 pre-release

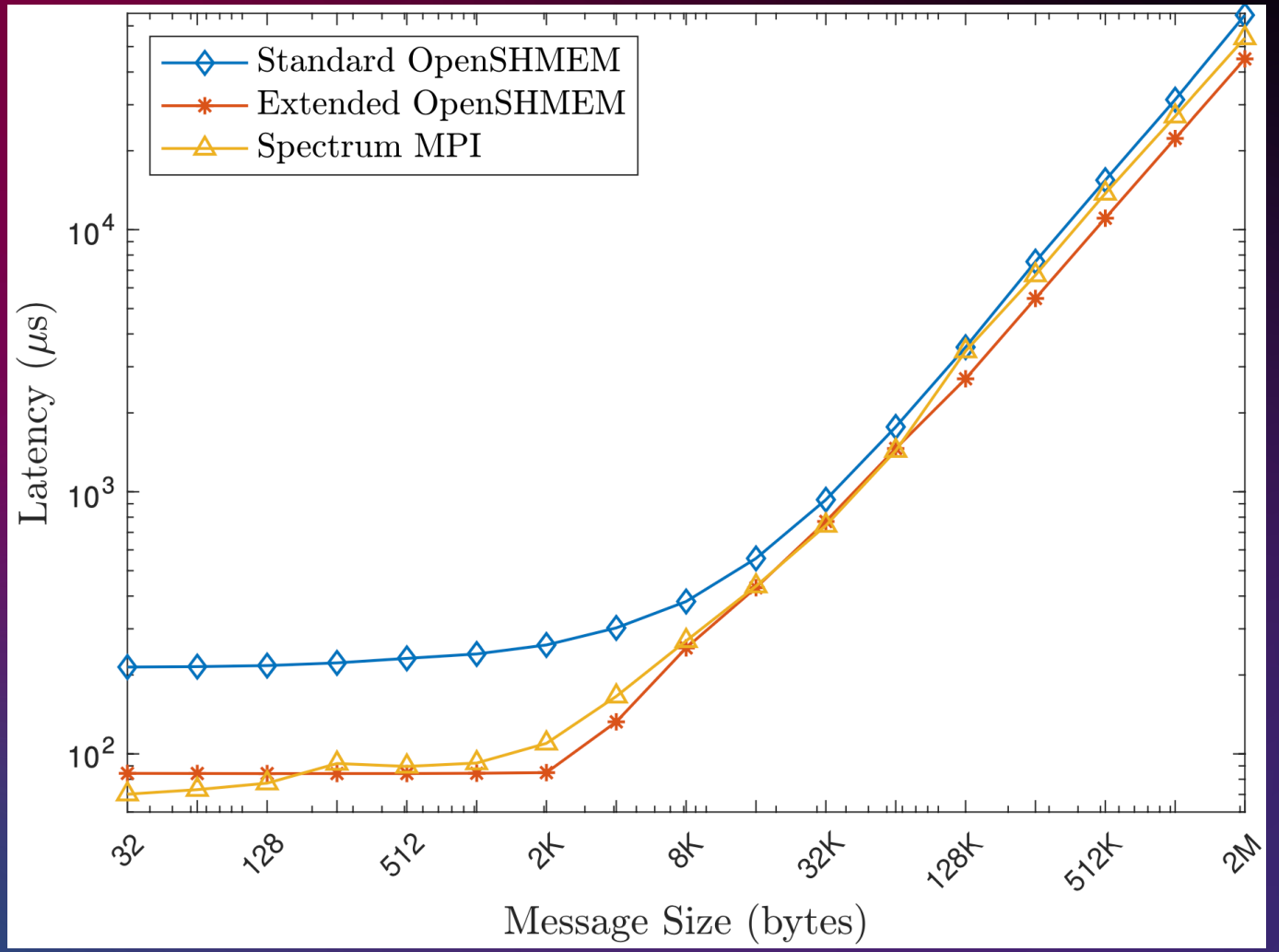
- MPI performance baseline
  - IBM Spectrum MPI 10.4.0 (PAMI backend)

Inter-Node Point-to-Point PUT Latency



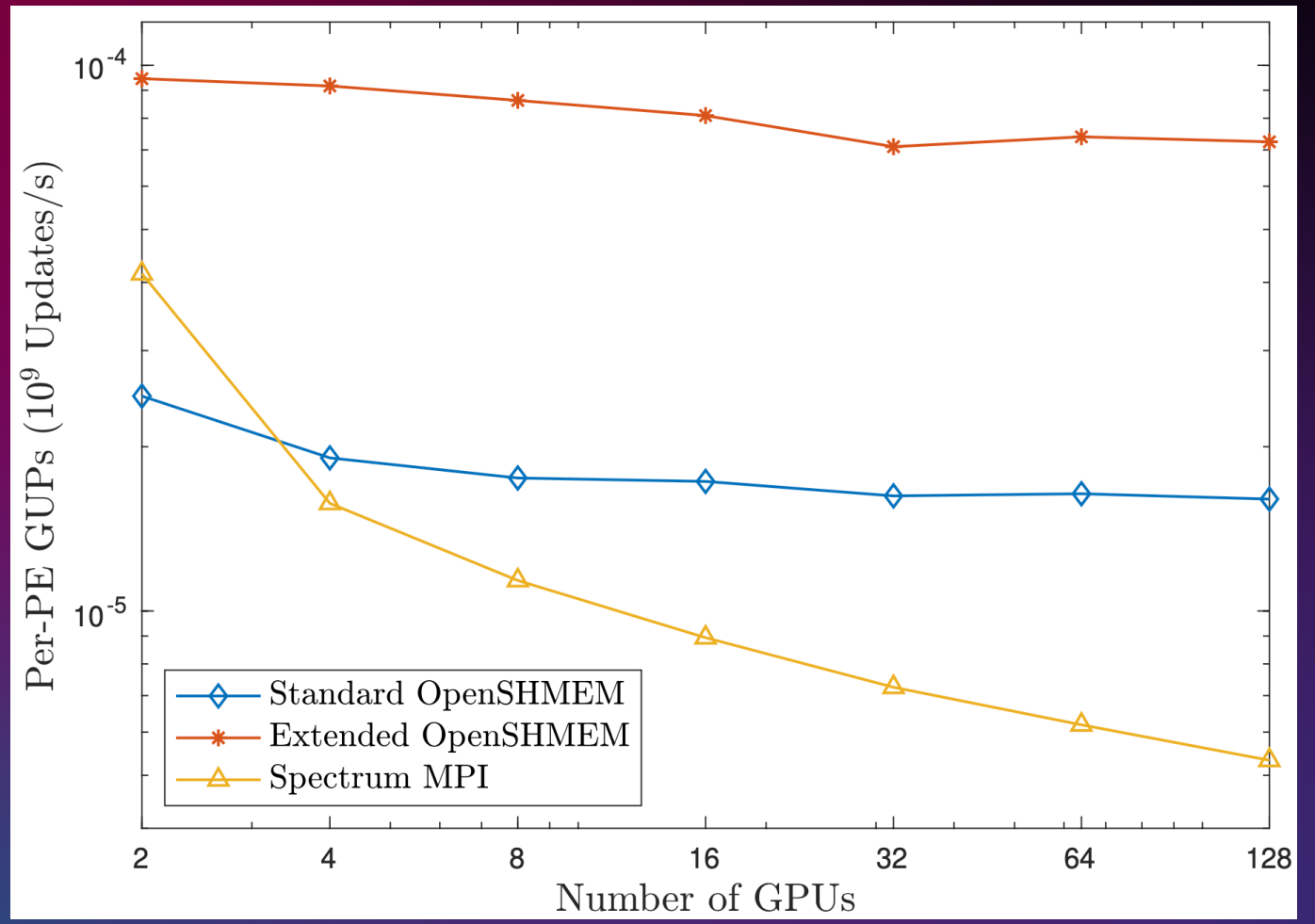
Lower is better

Inter-Node 48 GPU All-to-All Latency



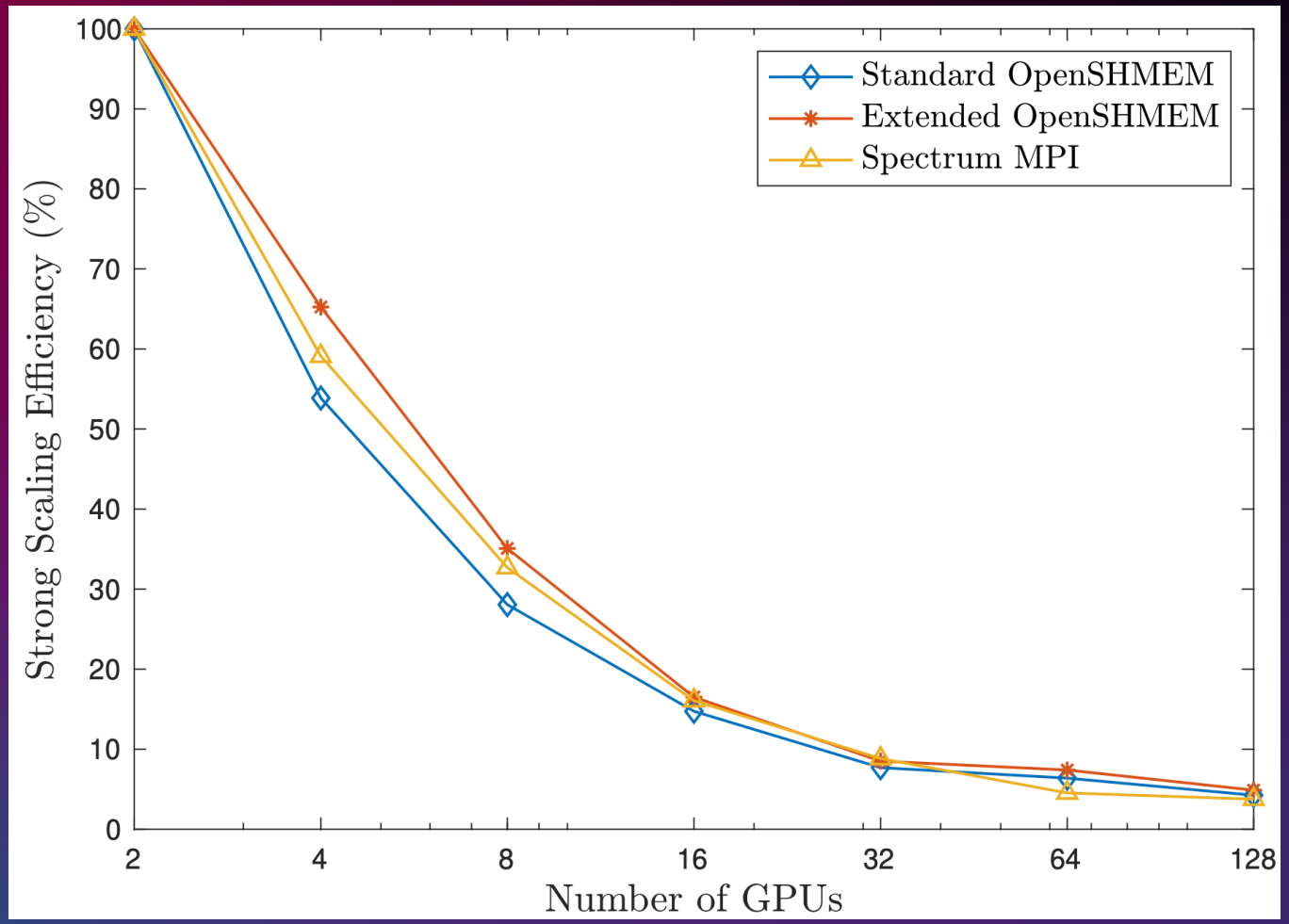
Lower is better

HPC Challenge Random Access (GUPs) Weak Scaling



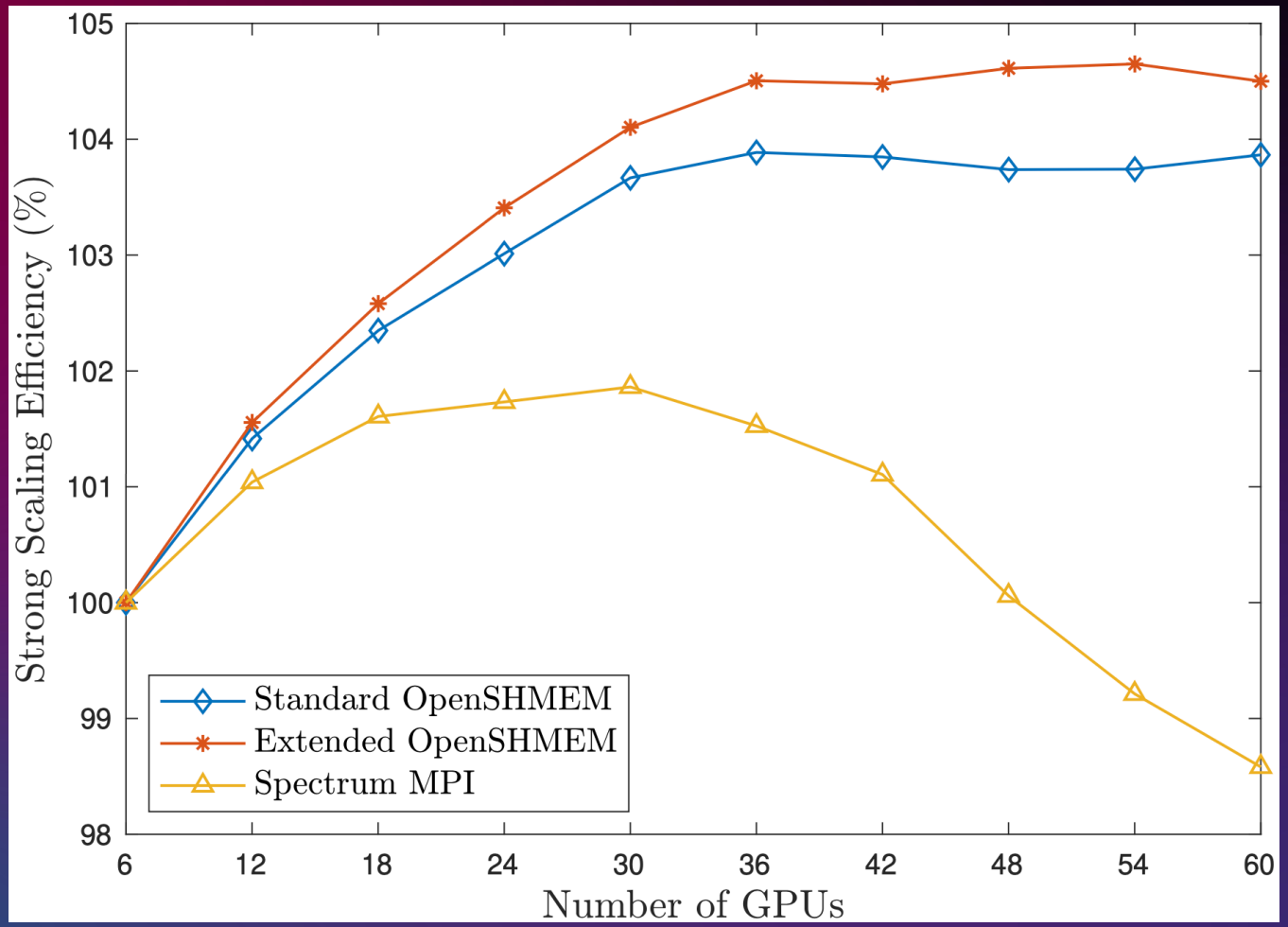
Higher is better

FFT Strong Scaling



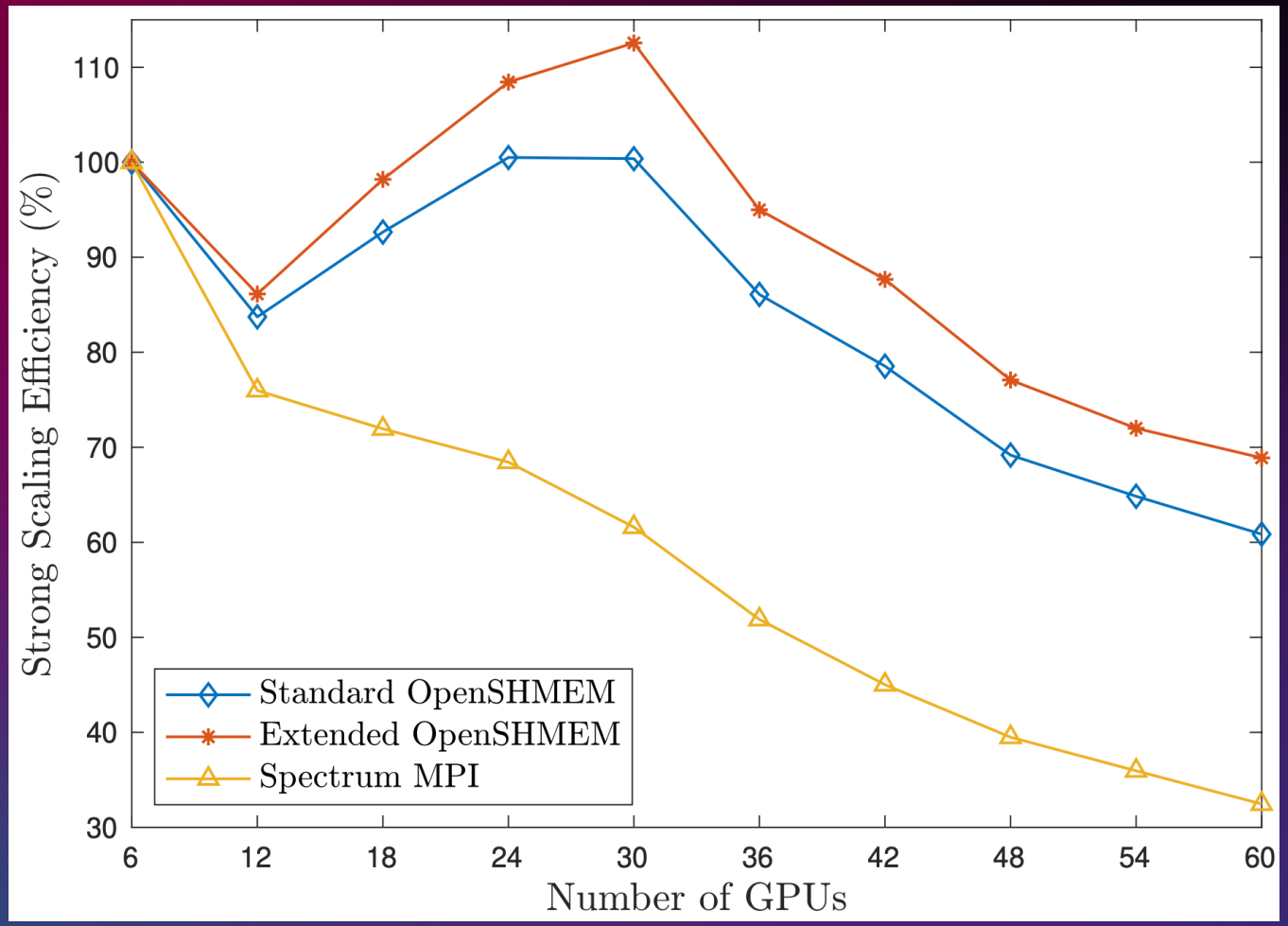
Higher is better

Cannon Matrix Multiplication Strong Scaling



Higher is better

3D Heat Equation Strong Scaling



Higher is better

Conclusion

- OpenSHMEM needs better GPU support for the exascale era
- OpenMP is excellent for programming GPUs, but doesn’t play well with PGAS
- Interoperability extensions to both OpenSHMEM and OpenMP turns them into a powerful pair
- Application developer can write fewer lines of code and get better performance
- Our experiments show great improvement over both vanilla OpenSHMEM + OpenMP and MPI + OpenMP

Future Work

- Move to teams-based OpenSHMEM extension
- Test on AMD & Intel GPUs
- Benchmark with real-world applications
- Compiler replace OpenSHMEM calls with NVSHMEM/ROC_SHMEM calls

Thank you!

iACS
INSTITUTE FOR ADVANCED COMPUTATIONAL SCIENCE

Stony Brook University

Exascal|lab
Exascale Programming Models Laboratory