Acceleration of HPC applications on hybrid CPU-GPU systems: When can Multi-Process Service (MPS) help?

GTC 2018
March 28, 2018

Olga Pearce (Lawrence Livermore National Laboratory) http://people.llnl.gov/olga
Max Katz (NVIDIA), Leopold Grinberg (IBM)

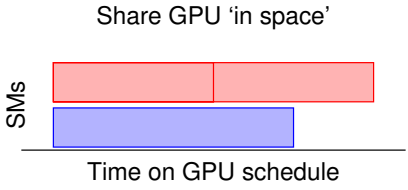
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Multi-Process Service (MPS)

Allows kernels launched from different (MPI) processes to be processed concurrently on the same GPU

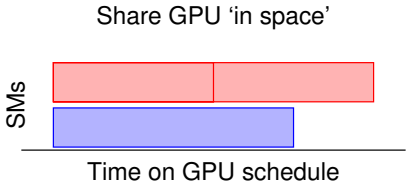
- Utilize inactive SMs when the work is small

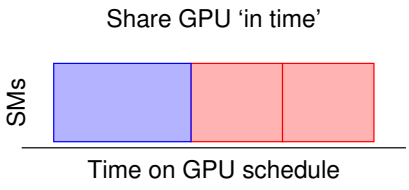


Multi-Process Service (MPS)

Allows kernels launched from different (MPI) processes to be processed concurrently on the same GPU

- Utilize inactive SMs when the work is small
- Processes take turns if every SM is occupied





Sierra system architecture finalized and currently under deployment at LLNL

Compute System

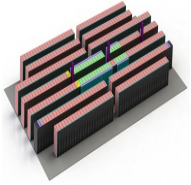
4,320 nodes

1.29 PB Memory

240 Compute Racks

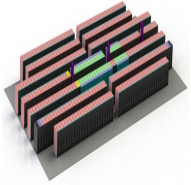
125 PFLOPS

≈ 12 MW



Sierra system architecture finalized and currently under deployment at LLNL

Compute System
4,320 nodes
1.29 PB Memory
240 Compute Racks
125 PFLOPS
≈ 12 MW

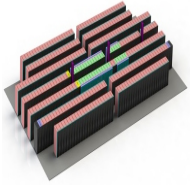


Compute Node
2 IBM POWER9 CPUs
4 NVIDIA Volta GPUs
256 GiB DDR4
NVMe-compatible PCIe 1.6 TB SSD
16 GiB Globally addressable HBM2
associated with each GPU
Coherent Shared Memory

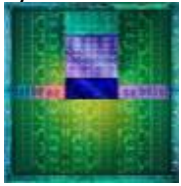
Sierra system architecture finalized and currently under deployment at LLNL

Compute System
4,320 nodes
1.29 PB Memory
240 Compute Racks
125 PFLOPS
≈ 12 MW

Compute Node
2 IBM POWER9 CPUs
4 NVIDIA Volta GPUs
256 GiB DDR4
NVMe-compatible PCIe 1.6 TB SSD
16 GiB Globally addressable HBM2
associated with each GPU
Coherent Shared Memory

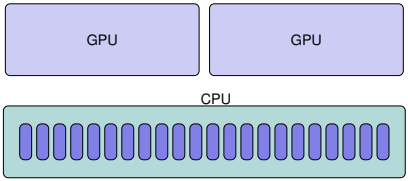


5% FLOPS



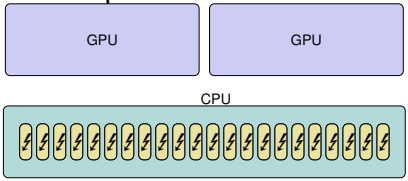
95% FLOPS

Ways to utilize a node of Sierra (showing one socket)



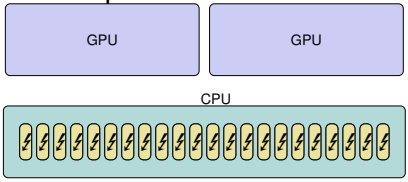
Ways to utilize a node of Sierra (showing one socket)

- MPI process/core

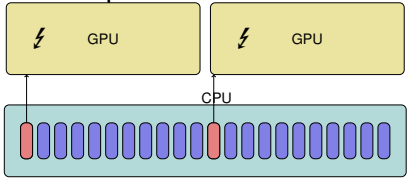


Ways to utilize a node of Sierra (showing one socket)

- MPI process/core

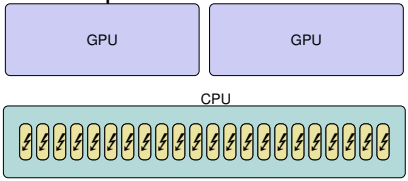


- MPI process/GPU

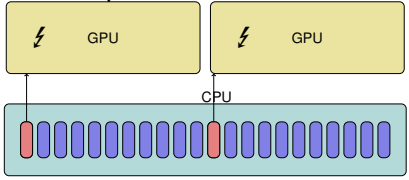


Ways to utilize a node of Sierra (showing one socket)

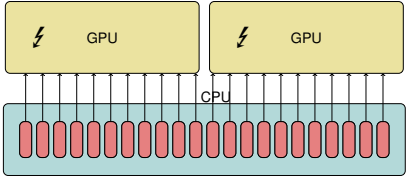
- MPI process/core



- MPI process/GPU

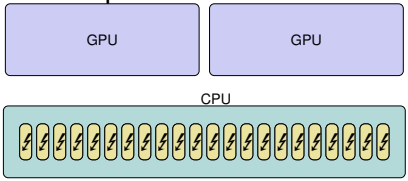


- MPI process/core, MPS for GPU

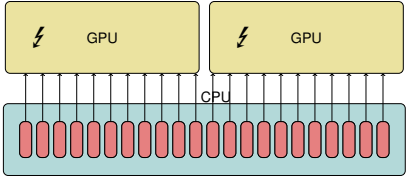


Ways to utilize a node of Sierra (showing one socket)

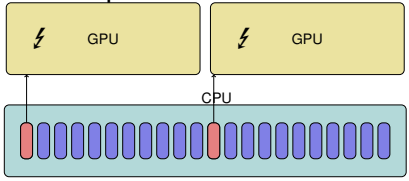
- MPI process/core

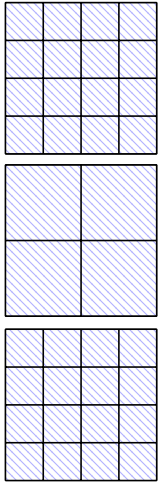


- MPI process/core, MPS for GPU



- MPI process/GPU





Parallel performance of multiphysics simulations

Decide how to run each phase of the multiphysics simulation

- On GPU vs. on CPU
- How many MPI processes (one per CPU core or one per GPU?)
- If some phases use one MPI process per CPU core, can we use MPS for the accelerated phases?

Parallel performance of multiphysics simulations

Decide how to run each phase of the multiphysics simulation

- On GPU vs. on CPU
- How many MPI processes (one per CPU core or one per GPU?)
- If some phases use one MPI process per CPU core, can we use MPS for the accelerated phases?

Outline:

- Tools used for measurement
- Multiphysics application
- How the application is accelerated
- Results: MPI process/GPU vs. 4 MPI processes/GPU + MPS
  - Impact on kernel performance
  - Impact on communication

Tool: Caliper [SC'16]

https://github.com/LLNL/Caliper

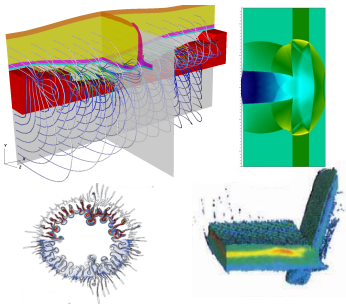
- Performance analysis toolbox, leverages existing tools
- Developed at LLNL
- Caliper team is responsive to our needs

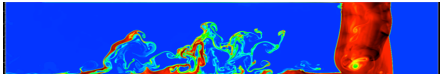
1. Annotate: begin/end API similar to timers libraries
   - Annotation of libraries (e.g., SAMRAI, hypre) combined seamlessly
2. Collect: Runtime parameters to instruct Caliper to measure:
   - Measure MPI function calls
   - Linux perf_event sampling (Libpfm)
   - Measure CUDA driver/runtime calls (using CUPTI)
3. Analyze
   - Using JSON output format

Application: ARES is a massively parallel, multi-dimensional, multi-physics code at LLNL

Physics Capabilities:

- ALE-AMR Hydrodynamics
- High-order Eulerian Hydrodynamics
- Elastic-Plastic flow
- 3T plasma physics
- High-Explosive modeling
- Diffusion, S_N Radiation
- Particulate flow
- Laser ray-tracing
- Magnetohydrodynamics (MHD)
- Dynamic mixing
- Non-LTE opacities





Applications:

- Inertial Confinement Fusion (ICF)
- Pulsed power
- National Ignition Facility debris
- High-Explosive experiments

ARES

- 800k lines of C/C++ with MPI
- 22 years old, used daily on our current supercomputers
- Single code base effectively utilizes all HPC platforms

ARES uses RAJA

https://github.com/LLNL/RAJA

- 800k lines of C/C++ with MPI
- 22 years old, used daily on our current supercomputers
- Single code base effectively utilizes all HPC platforms
- Use RAJA as an abstraction layer for on-node parallelization
  - RAJA is a collection of C++ software abstractions
  - Separation of concerns

C-style for-loop:

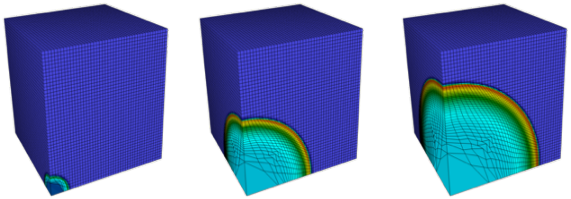
```
1: double* x; double* y;
2: double a;
3: for( int i = begin;
4:         i < end; ++i ) {
5:     y[i] += a * x[i];
6: }
```

RAJA-style loop:

```
1: double* x; double* y;
2: double a;
3: RAJA::forall<exec_policy>
4:         (begin, end, [=] (int i) {
5:     y[i] += a * x[i];
6: });
```

- Use different RAJA backends (CUDA, OpenMP)

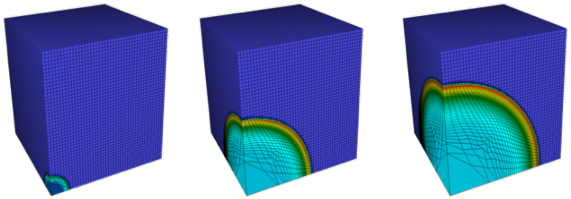
Results



3D Sedov blastwave problem

- Hydrodynamics calculation
- $\approx$ 80 kernels

Results



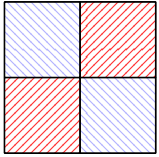
3D Sedov blastwave problem

- Hydrodynamics calculation
- $\approx$ 80 kernels

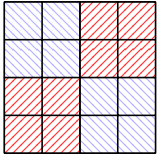
- Pre-SIERRA machine (rzmanta) - Minsky nodes:
  - 2x Power8+ CPUs (20 cores)
  - 4x NVIDIA P100 (Pascal) GPUs with 16GB memory each
  - NVLINK 1.0
  - * Some results generated with pre-release versions of compilers; improvements in performance expected in future releases
- All results shown use 4 Minsky nodes (16 GPUs)

Domain decomposition with and without MPS

MPI process/ GPU



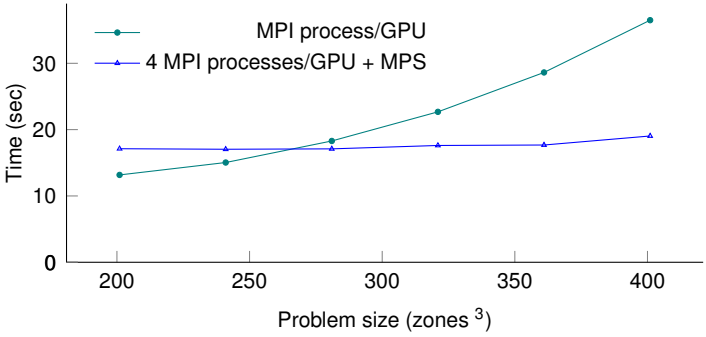
4 MPI processes/ GPU + MPS



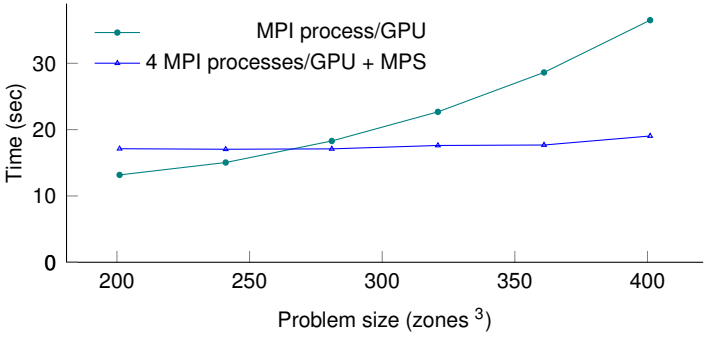
Differences:

- Computation: Work per MPI process
- Communication: Neighbors and surface to volume ratio

Overall runtime with and without MPS



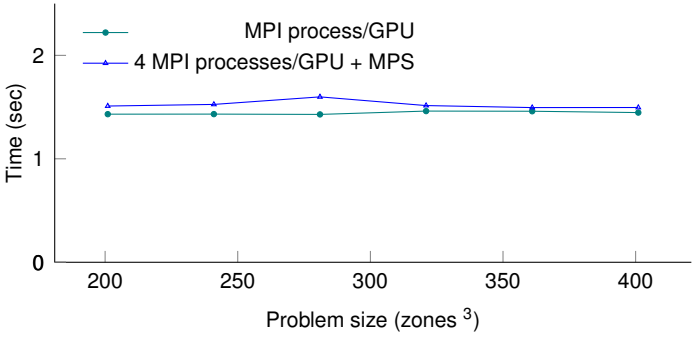
Overall runtime with and without MPS



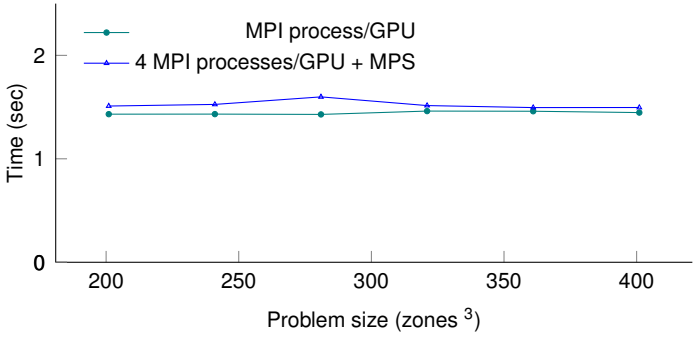
Differences:

- Computation
- Communication
- Memory

Computation time: Small kernels

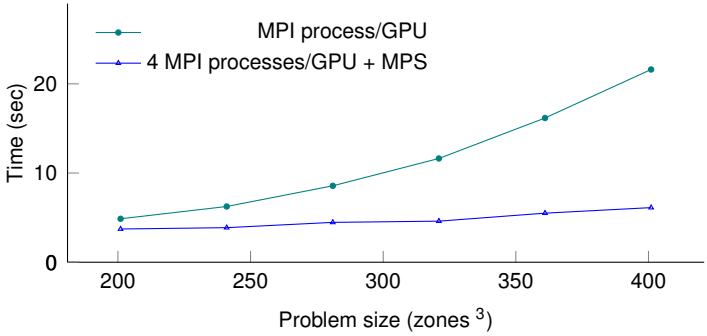


Computation time: Small kernels

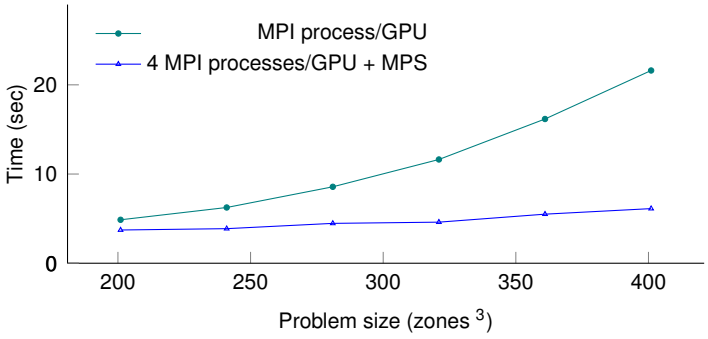


- Few zones, small amount of work per zone
- Dominated by kernel launch overhead
- MPS may be slightly slower

Computation time: Large kernels



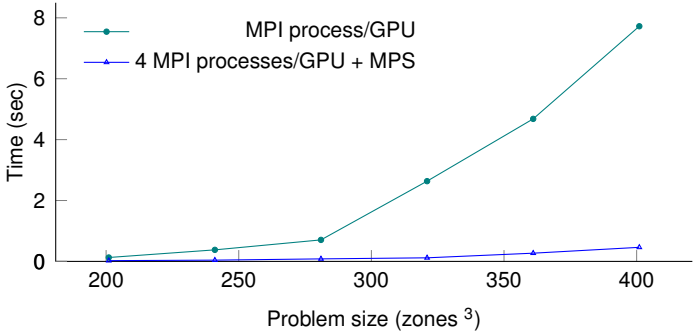
Computation time: Large kernels



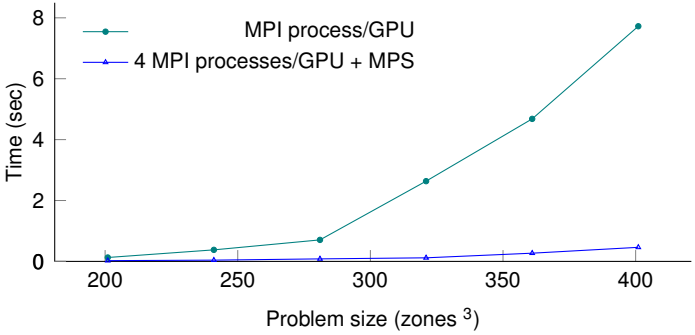
MPS is faster especially when problem size is large

- Utilizing GPU better?
  - GPU utilization?
  - GPU occupancy?
- Utilizing CPU better?
  - More parallelization?
  - Better utilization of CPU memory bandwidth?

Waiting on the GPU: cudaDeviceSynchronize



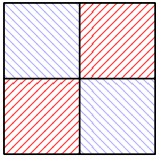
Waiting on the GPU: cudaDeviceSynchronize



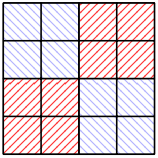
- Appear to be waiting on the GPU longer without MPS

Domain decomposition with and without MPS

MPI process/ GPU

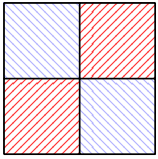


4 MPI processes/ GPU + MPS

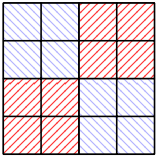


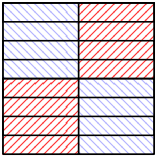
Domain decomposition with and without MPS

MPI process/ GPU



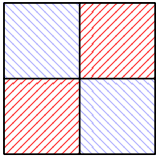
4 MPI processes/ GPU + MPS



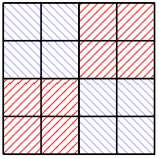


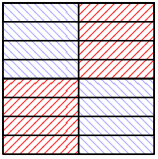
Domain decomposition with and without MPS

MPI process/ GPU



4 MPI processes/ GPU + MPS

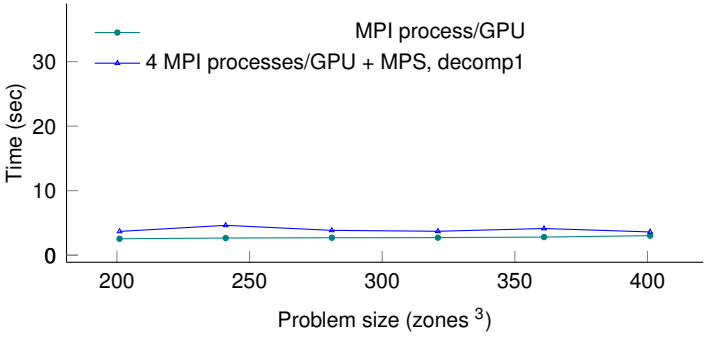




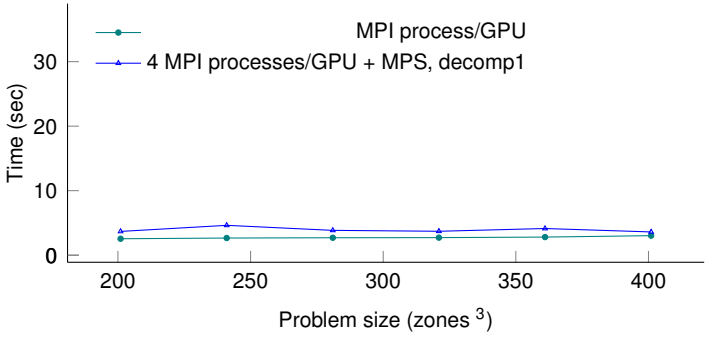
Differences in communication:

- Number of neighbors in halo exchange
- Surface to volume ratio
- Processor mapping
- Other

Communication time (MPI)

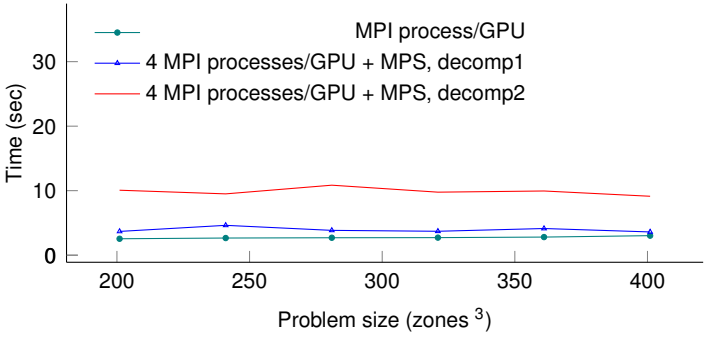


Communication time (MPI)



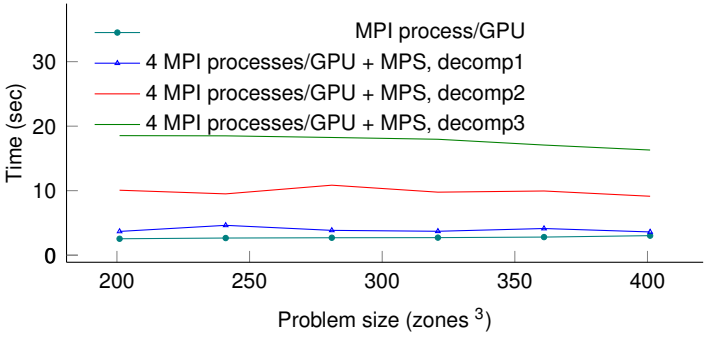
- More MPI processes = more communication

Communication time (MPI)



- More MPI processes = more communication

Communication time (MPI)



- More MPI processes = more communication
- Not all decompositions result in the same communication time

Conclusions

- MPS can be useful if non-accelerated portions of the code need all CPU cores
- MPS can help to utilize the GPU better
- However, using more MPI processes makes communication more expensive - many factors may have impact
- Caliper measures many aspects of performance, but there are more questions
  - Does using more CPU cores increase CPU memory bandwidth utilization?
  - How well am I utilizing/occupying the GPU?
  - What is the bottleneck now: the CPU or the GPU?
  - Other issues on new platforms

Thank you

- ARES team — Lawrence Livermore National Laboratory
- Caliper team — https://github.com/LLNL/Caliper
- RAJA team — https://github.com/LLNL/RAJA
- Steve Rennich, Max Katz — NVIDIA
- Leopold Grinberg — IBM