Preliminary Orchestration Runtime Testing

- Executed on toy problem
- Simple initial conditions for “energy” and “density”
- Two separate & independent actions to be applied to data
  - Apply Laplacian to energy
  - Apply Laplacian to density
- Tests also included a single action containing fused kernels
- Solved on a uniform grid with
  - 16 x 16 blocks (default FLASH setting)
  - 256 blocks total
  - 1 MPI Process with 7 cores/hardware threads off + 1 GPU*
  - Run with different block totals to verify correct scaling

* The runtime implementation should work with more processes, but it hasn’t been tested yet

Timing Test Information

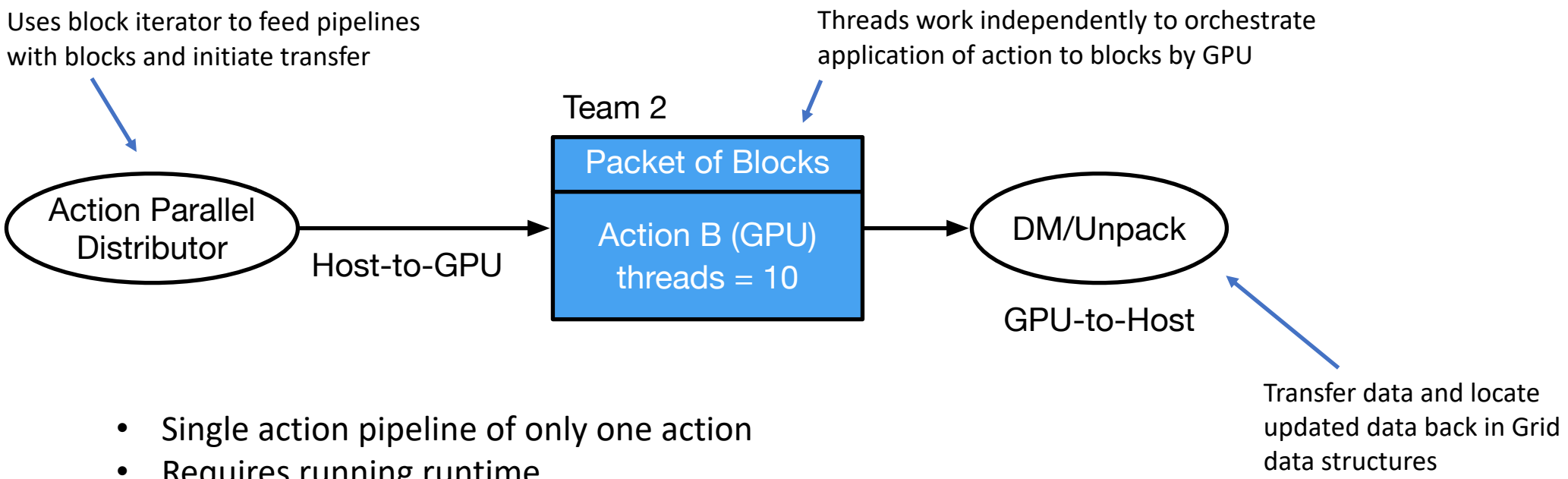
Summit

- PGI 19.9 / CUDA 10.1.243
- spectrum-mpi/10.3.1.2-20200121
- 1 MPI Process with 7 cores/hardware threads off + 1 GPU
- No optimization by compiler
- 28 October 2020 / Commit 8166903b
- Job ID 447085

The runtime is using

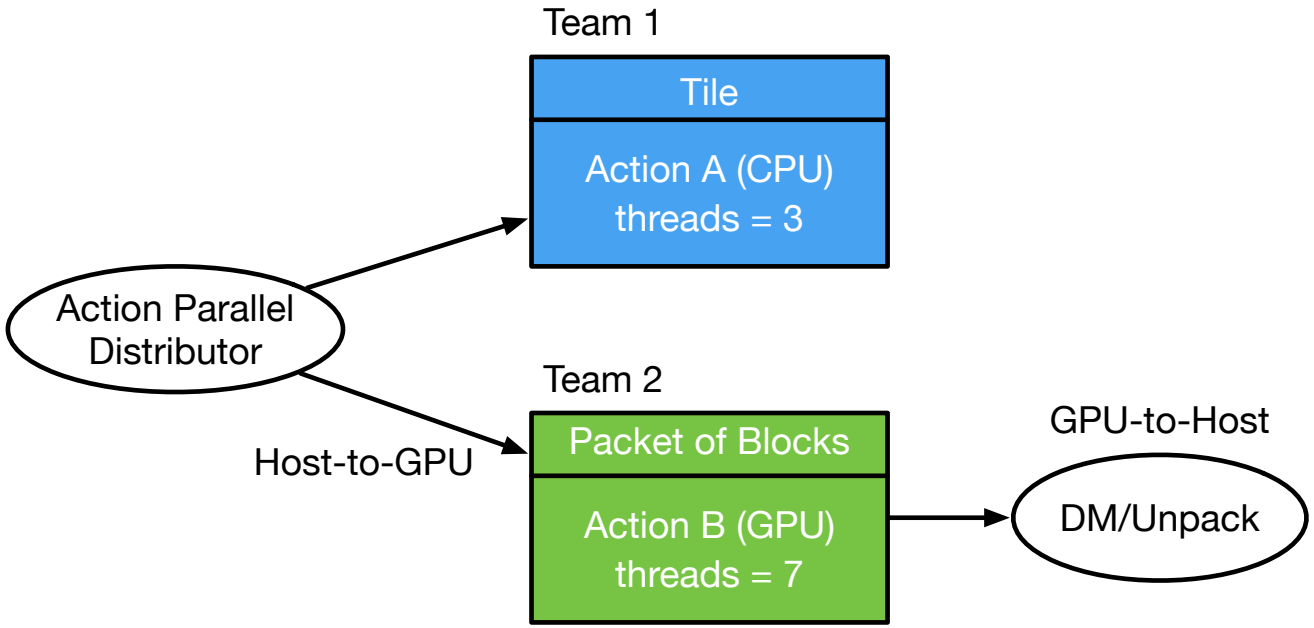
- AMReX-based (v20.08) Grid backend
- CUDA backend for memory allocation and data transfer
- Not using managed memory to potentially improve portability
- OpenACC offloading of Laplacian computation

GPU-only Thread Team Configuration



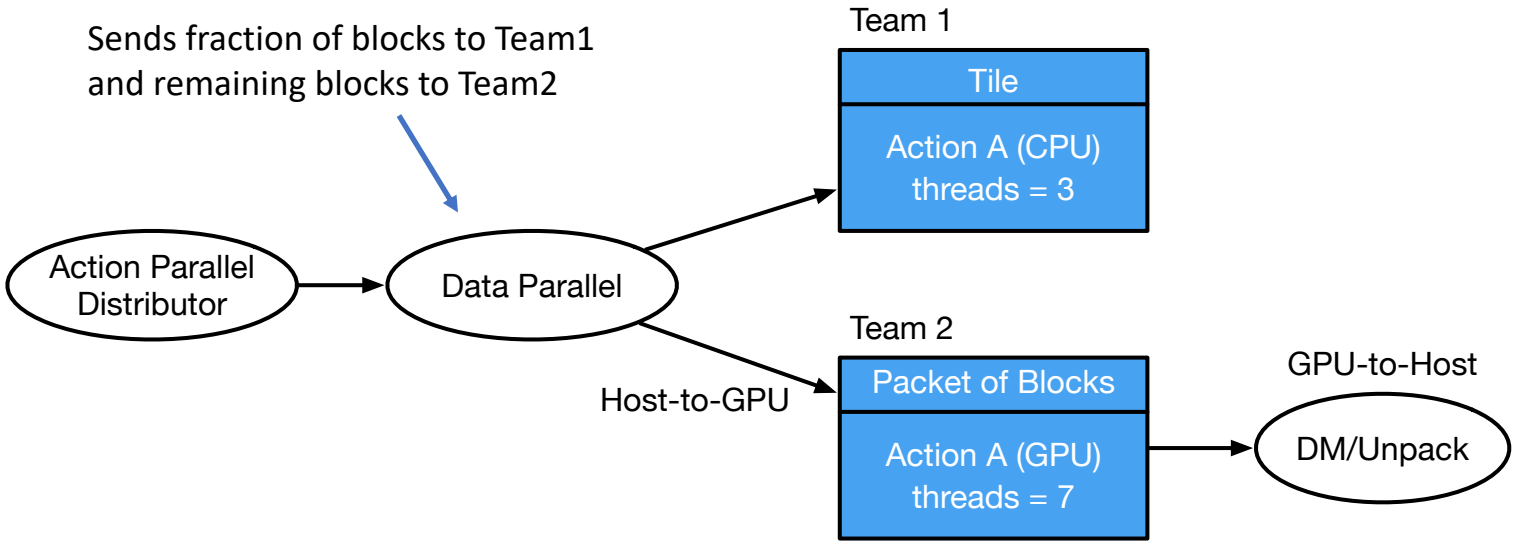
- Single action pipeline of only one action
- Requires running runtime
  - twice in the case of two separate actions
  - once for fused kernels action
- Action Parallel Distributor uses 1 host thread
- Number of active thread team host threads in {1, …, $N_{cores} - 1$}
- When distributor's thread waits, another thread in team can be activated
- Host-to-GPU is asynchronous, but GPU-to-Host is still synchronous

CPU-GPU Concurrent Configuration



- Two independent action pipeline of only one action each
- Each block is sent to each pipeline and execution of each action proceeds concurrently
- Active threads in distributor and two teams balanced so that there are always N_{cores} host threads in use

CPU-GPU Data Parallel Configuration



- Single action pipeline with a CPU and GPU version of the fused action
- Application of fused action proceeds concurrently on CPU and GPU
- Active threads in distributor and two teams balanced so that there are always N_{cores} host threads in use

Preliminary Runtime Timing Results

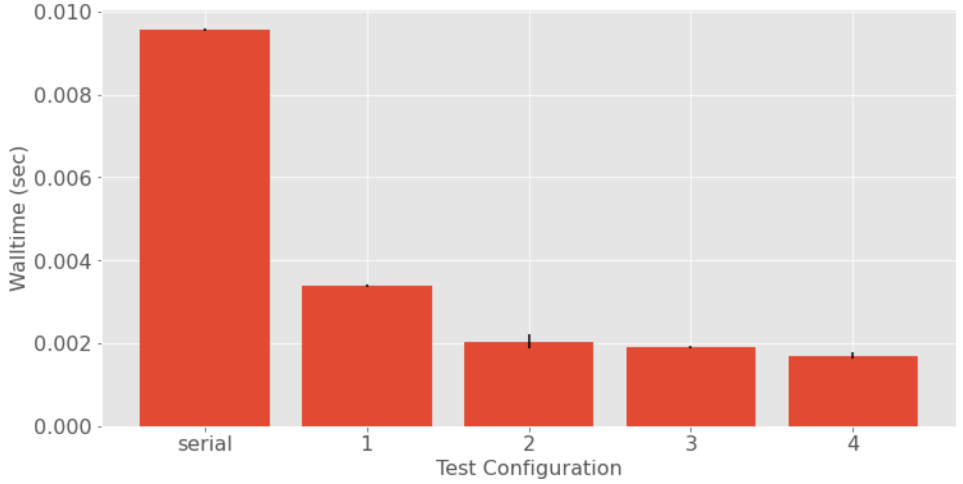
20 repeated trials and tiling not used

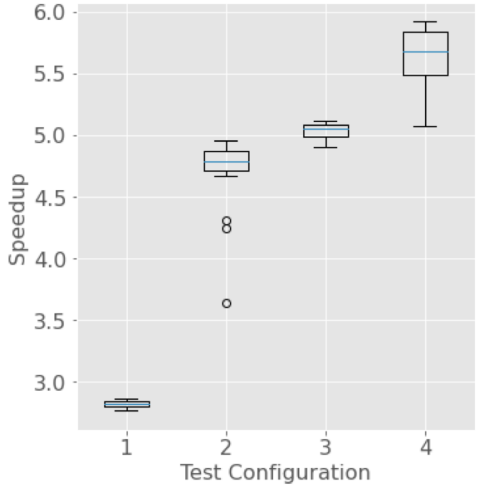
Approximate walltimes per block

- Single Laplacian executed on CPU = 18.7 μs (measured w/o profiling)
- Single Laplacian executed on GPU = 230 μs (profiled)
- Fused Laplacians executed on GPU = 417.5 μs (profiled)
- No manual tuning/optimization of code run on GPU

Test configurations (roughly optimized settings)

1. Each action run on the GPU in succession
   - 1+4 host threads and 110 blocks/packet
2. One action run on CPU and other run concurrently on GPU
   - 1+4 host threads for CPU
   - 2 host threads for GPU and 110 blocks/packet
3. Single action with fused kernels run on GPU
   - 1+1 host threads and 180 blocks/packet
   - Note that 5 threads are free to run a different action on CPU
4. Data parallel CPU/GPU with fused kernels
   - 1+4 host threads for CPU
   - 2 host threads for GPU and 60 blocks/packet
   - ~62% blocks sent to CPU and remainder to GPU

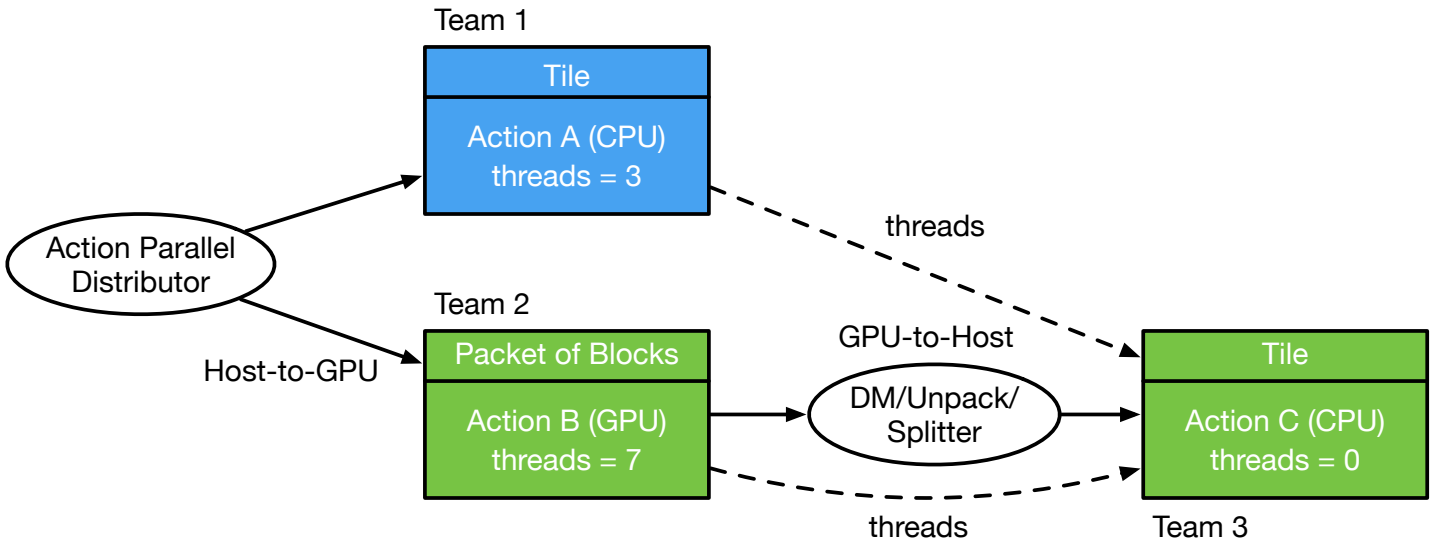




Areas of Improvement

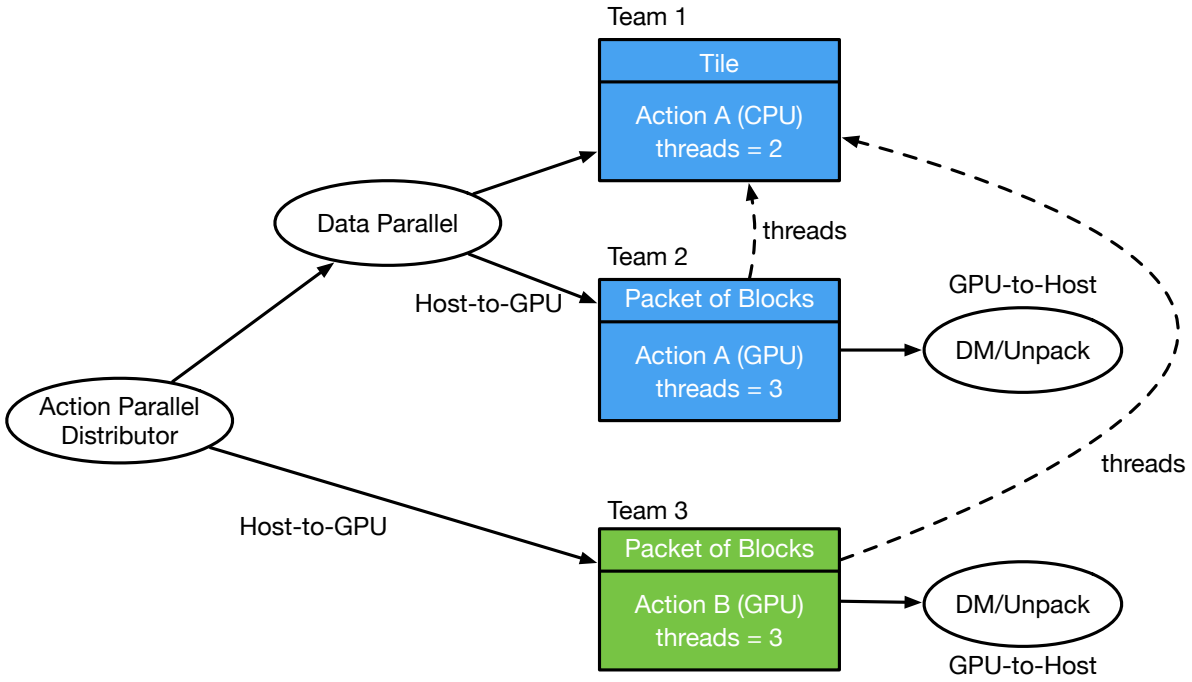
- Determine if sensible that GPU walltime/block is an order of magnitude more than CPU walltime/block
- Select configuration parameters with better optimization technique
- OpenACC + CUDA limited to 32 streams, which can slow down Host-to-GPU data transfers
- Make GPU-to-Host data transfers asynchronous
- Study C++ stdlib containers to determine which yield best performance
- Optimize memory use in data packets
- Allow for fusing of actions in addition to fusing of kernels
  - Launch more short-duration kernels per data packet transfer
- Improve shared resource management in thread teams only if this is serious limiting factor

Configurations Not Timed Yet



- Extend green action pipeline by addition of Action C
- Works if B/C independent or if C must be run after B
- Balancing of host thread resources
  - Dashed arrows indicate that a thread that goes Idle in one team can first activate a thread in another team

Configurations Not Timed Yet



- Two thread teams concurrently launching kernels on same GPU