POP2_AR_065: HemeLB_GPU on JUWELS performance assessment report

Brian Wylie, Jülich Supercomputing Centre (JSC)

b.wylie@fz-juelich.de, July 2020

EU H2020 Center of Excellence (CoE)

1 December 2018 – 30 November 2021

Grant Agreement No 824080

Background

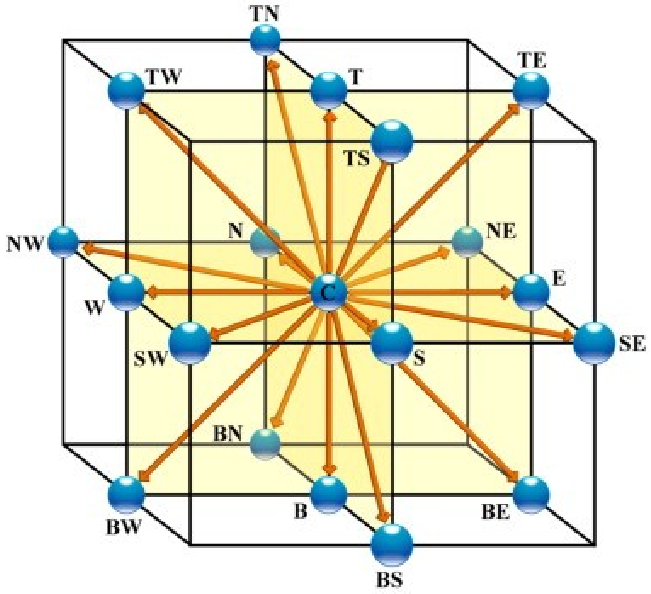
- Applicant: Ioannis Zacharoudiou, UC London, UK (developer) [HPC CoE CompBioMed]
- Code: **HemeLB_GPU** (3D macroscopic blood flow simulation) C++ & MPI & CUDA
- Platform: **JUWELS** (@ JSC)
  - 56 dual 20-core Intel Xeon Platinum 6148 ‘Skylake’ compute nodes each accelerated with 4 Nvidia V100 ‘Volta’ GPUs
  - GCC/8.3.0 & CUDA/10.1.105 & ParaStationMPI/5.4

Model of human male arteries,
used to simulate blood pressure
during one heart cycle

- Testcase: 1.78 GiB CBM2019_Arteries_patched geometry
  - 66,401,494 lattice sites; 1+38 iolets; simulation of 2,000 time-steps (of 100μs)
- Scale: up to 128+1 processes (32 compute nodes, each with 4 MPI ranks)
- Scalasca/Score-P summary and trace measurements
  - using selective instrumentation filter

HemeLB lattice-Boltzmann method for haemodynamic simulation

- Fluid particles tracked at macroscopic level on a lattice grid
  - discrete set of permissible velocities
  - only nearest-neighbour interactions
- Comparable accuracy to conventional continuum CFD
  - relatively straightforward implementation of complex boundary conditions
  - exhibit superior parallel performance
    - less communication between computational subdomains



www.2020science.net/software/hemelb.html

www.hemelb.org

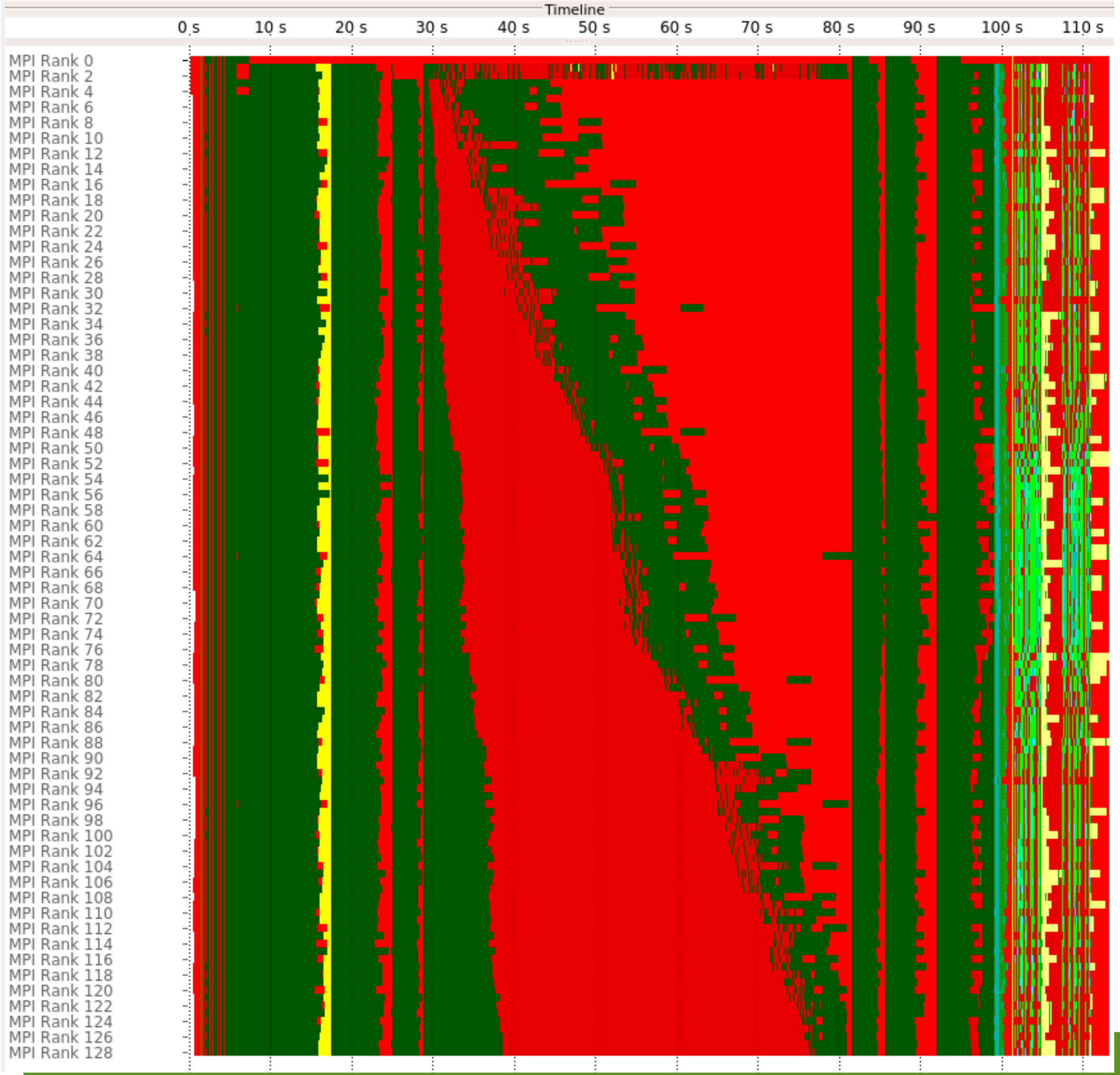
Details

- JUWELS (GPU nodes): 46 (+10 reserved for development)
  - 192 GiB compute nodes; Dual EDR-Infiniband (Connect-X4)
  - IBM Spectrum Scale (GPFS) parallel filesystem; CentOS Linux 7.8.2003
- Execution configuration:
  - gres=gpu:4, ntasks-per-node=4 (plus one extra for monitor on first node)
  - all GPUs used, but only 10% of CPU cores
- HemeLB: GPU development version (using BasicDecomposition scheme)
  - configured to offload kernels from each MPI process to associated GPU
  - MPI File writing of intermediate properties state every 1000 steps
  - 2 variants (reordered ‘a’ & original ‘b’) order of actions within each timestep

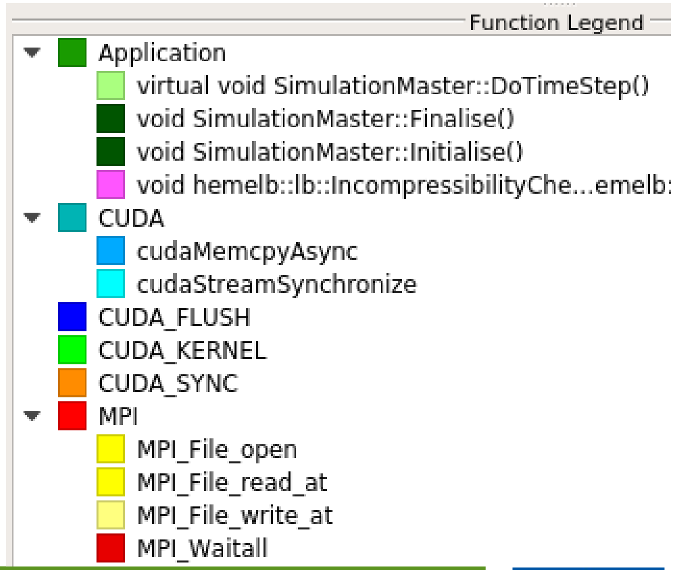
Method

- Score-P/6.0 (GCC+ParaStationMPI) instrumenter used when building code
  - CMake: CUDACXX=scorep-nvcc (no instrumentation of CXX)
    - workaround replace “-dc” with “--relocatable-device=true -c”
  - set SCOREP_WRAPPER_INSTRUMENTER_FLAGS=
    “--mpp=mpi --thread=none --cuda --instrument-filter=hemelb.filt”
- Scalasca/2.5 runtime measurement configuration
  - SCAN_TRACE_ANALYZER=none
    SCOREP_MPI_ENABLE_GROUPS='coll','env','io','p2p','rma','topo','xnonblock' # 'cg'
    SCOREP_CUDA_ENABLE=runtime,memcpy,kernel,sync,flushatexit
    SCOREP_CUDA_BUFFER=10M
    SCOREP_TOTAL_MEMORY=64M
- Profiles explored with CUBE/4.4.4, execution traces examined with Vampir/9.8.0

HemeLB_GPU execution timeline (v1.20a with file writing)



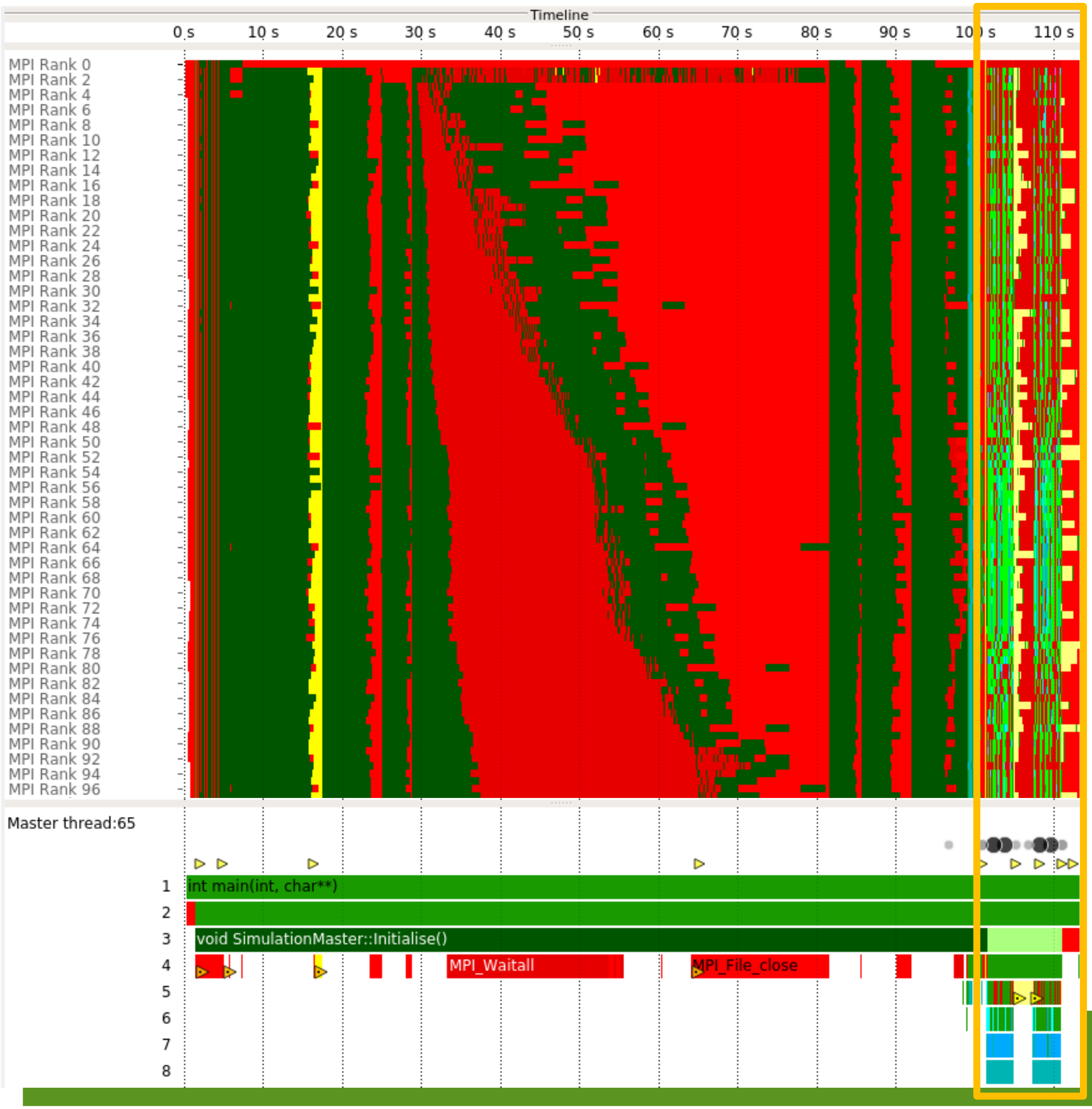
- 32 nodes: 129 MPI processes driving 128 GPUs
  - monitor rank 0 not participating in simulation
  - reader ranks 1 &2 distribute simulation geometry, then participate in simulation



Function Legend

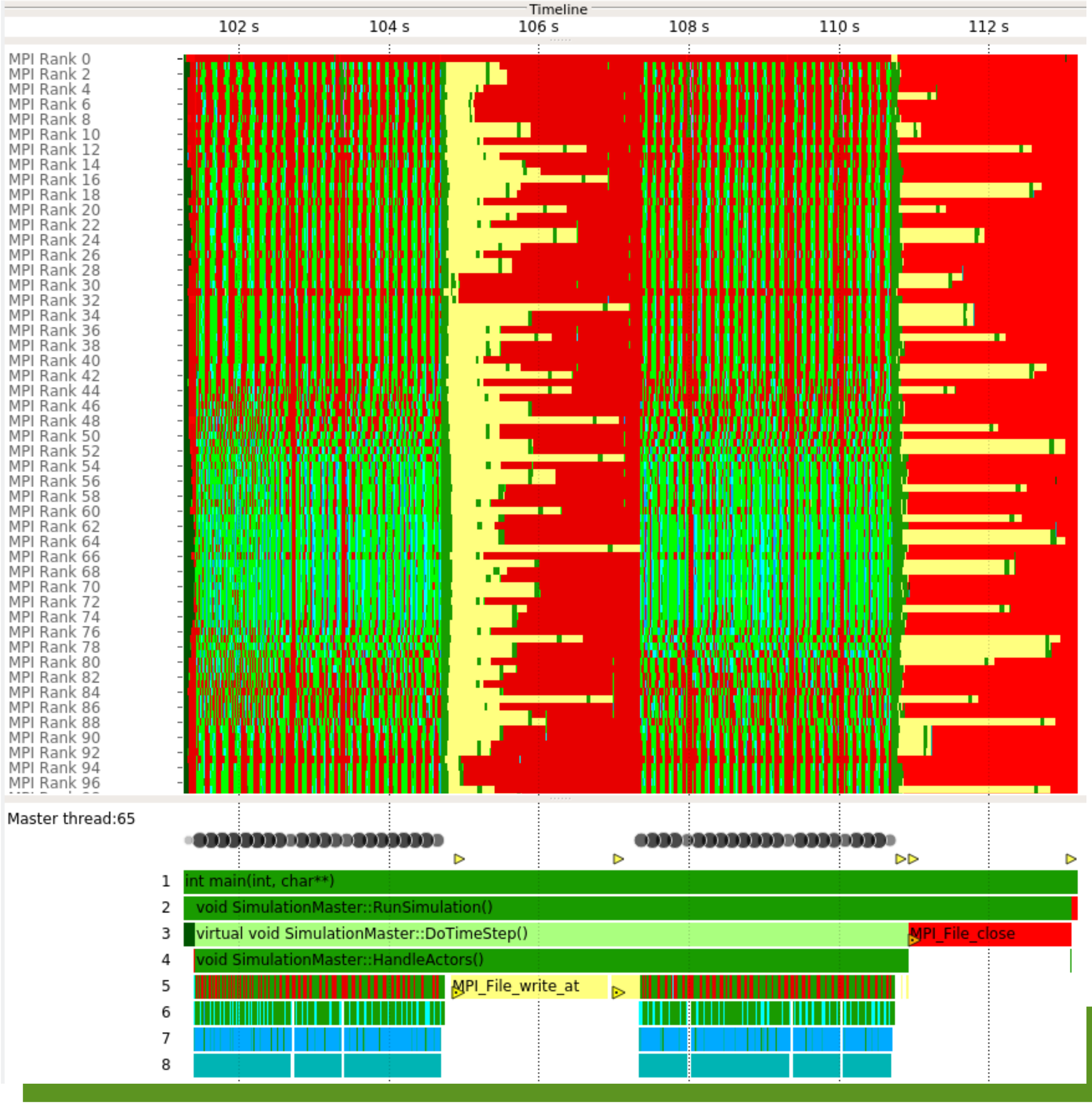
- 🟩 Application
  - 🟩 virtual void SimulationMaster::DoTimeStep()
  - 🟩 void SimulationMaster::Finalise()
  - 🟩 void SimulationMaster::Initialise()
  - 🟪 void hemelb::lb::IncompressibilityChe...emelb:
- 🟦 CUDA
  - 🟦 cudaMemcpyAsync
  - 🟦 cudaStreamSynchronize
- 🟦 CUDA_FLUSH
- 🟩 CUDA_KERNEL
- 🟧 CUDA_SYNC
- 🟥 MPI
  - 🟨 MPI_File_open
  - 🟨 MPI_File_read_at
  - 🟨 MPI_File_write_at
  - 🟥 MPI_Waitall

HemeLB_GPU execution timeline: Focus of Analysis (FOA) *DoTimeStep*



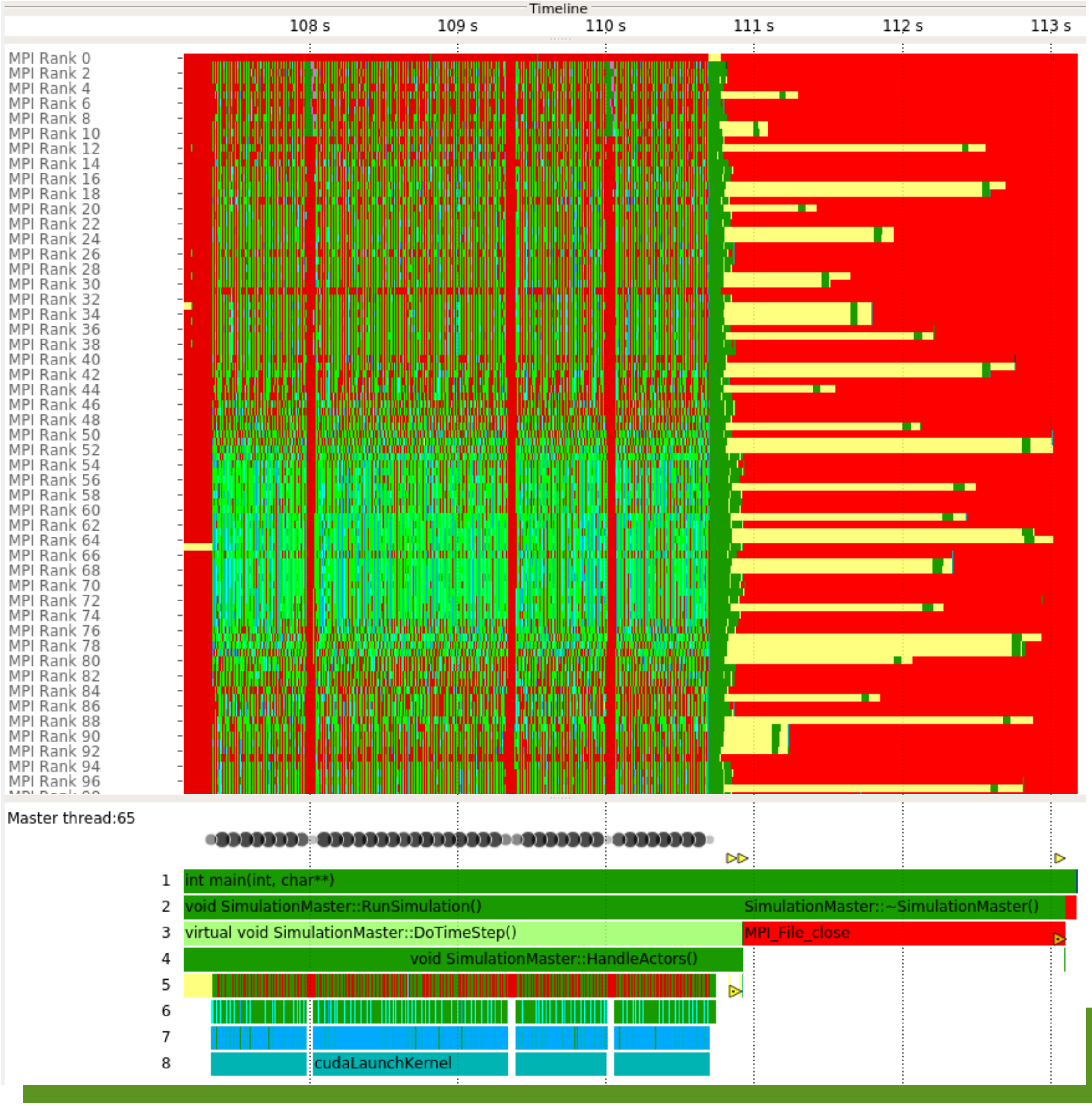
- 32 nodes: 129 MPI processes driving 128 GPUs
  - monitor rank 0 not participating in simulation
  - reader ranks 1 &2 distribute simulation geometry, then participate in simulation
- SimulationMaster class constructor/destructor methods initialisation and finalisation
- FOA is RunSimulation *DoTimeStep* routine
  - 2000 simulation steps: approx. 101-113 seconds
  - CUDA kernel offloads to dedicated GPU

HemeLB_GPU execution timeline: FOA zoom



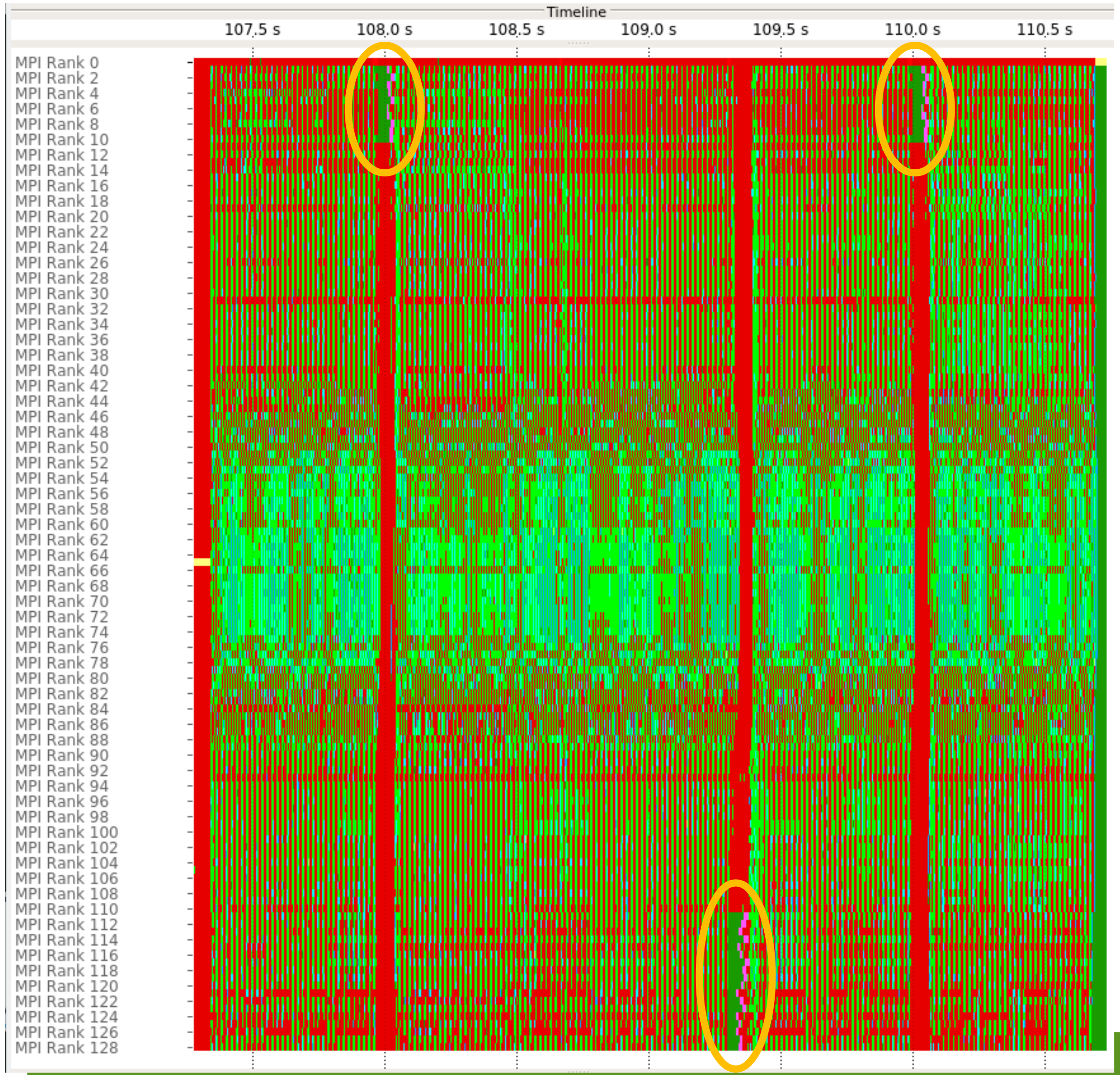
- 32 nodes: 129 MPI processes driving 128 GPUs
  - monitor rank 0 not participating in simulation
  - reader ranks 1 &2 distribute simulation geometry, then participate in simulation
- SimulationMaster class constructor/destructor methods initialisation and finalisation
- FOA is RunSimulation *DoTimeStep* routine
  - 2000 simulation steps: approx. 101-113 seconds
  - CUDA kernel offloads to dedicated GPU
  - property file writing after each 1000 steps
    - MPI_File_write_at by each process

HemeLB_GPU execution timeline: FOA zoom (1000 steps incl. writing)



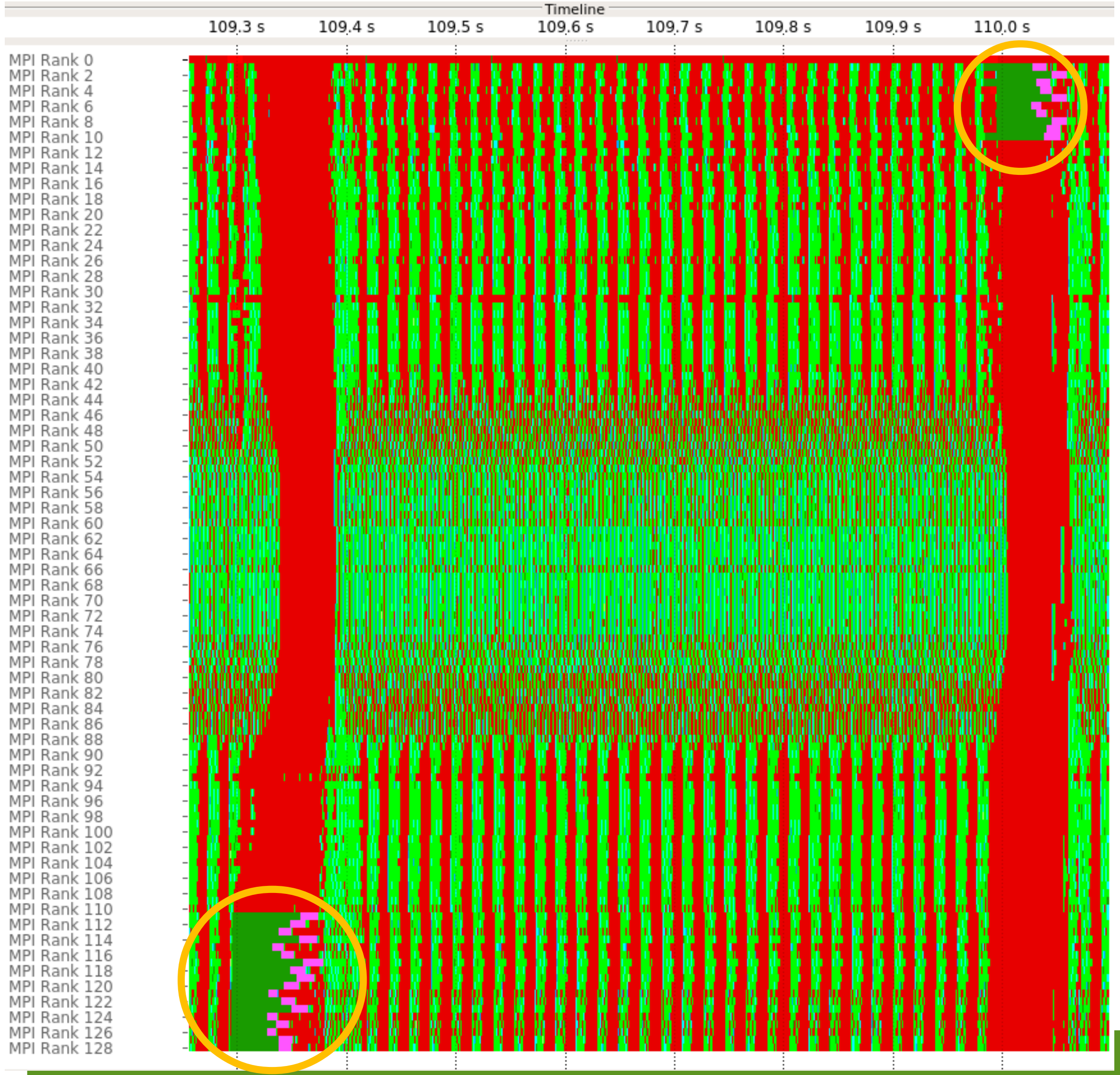
- 32 nodes: 129 MPI processes driving 128 GPUs
  - monitor rank 0 not participating in simulation
  - reader ranks 1 &2 distribute simulation geometry, then participate in simulation
- SimulationMaster class constructor/destructor methods initialisation and finalisation
- FOA is RunSimulation *DoTimeStep* routine
  - 2000 simulation steps: approx. 101-113 seconds
  - CUDA kernel offloads to dedicated GPU
  - property file writing after each 1000 steps
    - MPI_File_write_at by each process, very imbalanced
    - amount written varies 27-84MB

HemeLB_GPU execution timeline: FOA zoom (1000 steps)



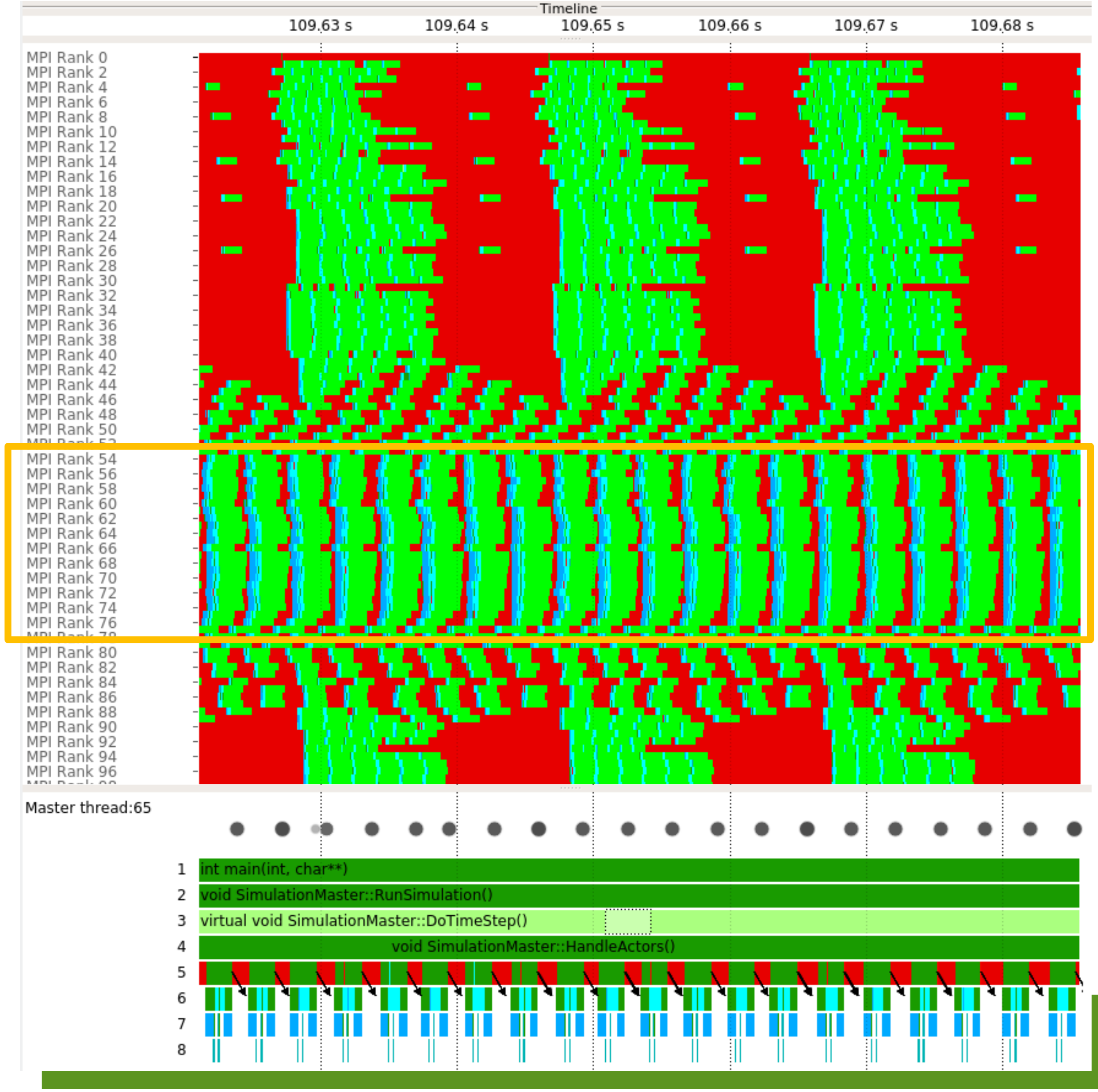
- 32 nodes: 129 MPI processes driving 128 GPUs
  - monitor rank 0 not participating in simulation
  - reader ranks 1 &2 distribute simulation geometry, then participate in simulation
- SimulationMaster class constructor/destructor methods initialisation and finalisation
- FOA is RunSimulation *DoTimeStep* routine
  - 2000 simulation steps: approx. 101-113 seconds
  - CUDA kernel offloads to dedicated GPU
  - property file writing after each 1000 steps
  - *IncompressibilityChecker* each 200 steps (mostly)
    - only lowest 10 simulation ranks
    - **or** highest 14 simulation ranks (32 nodes)

HemeLB_GPU execution timeline: FOA zoom (250 steps)



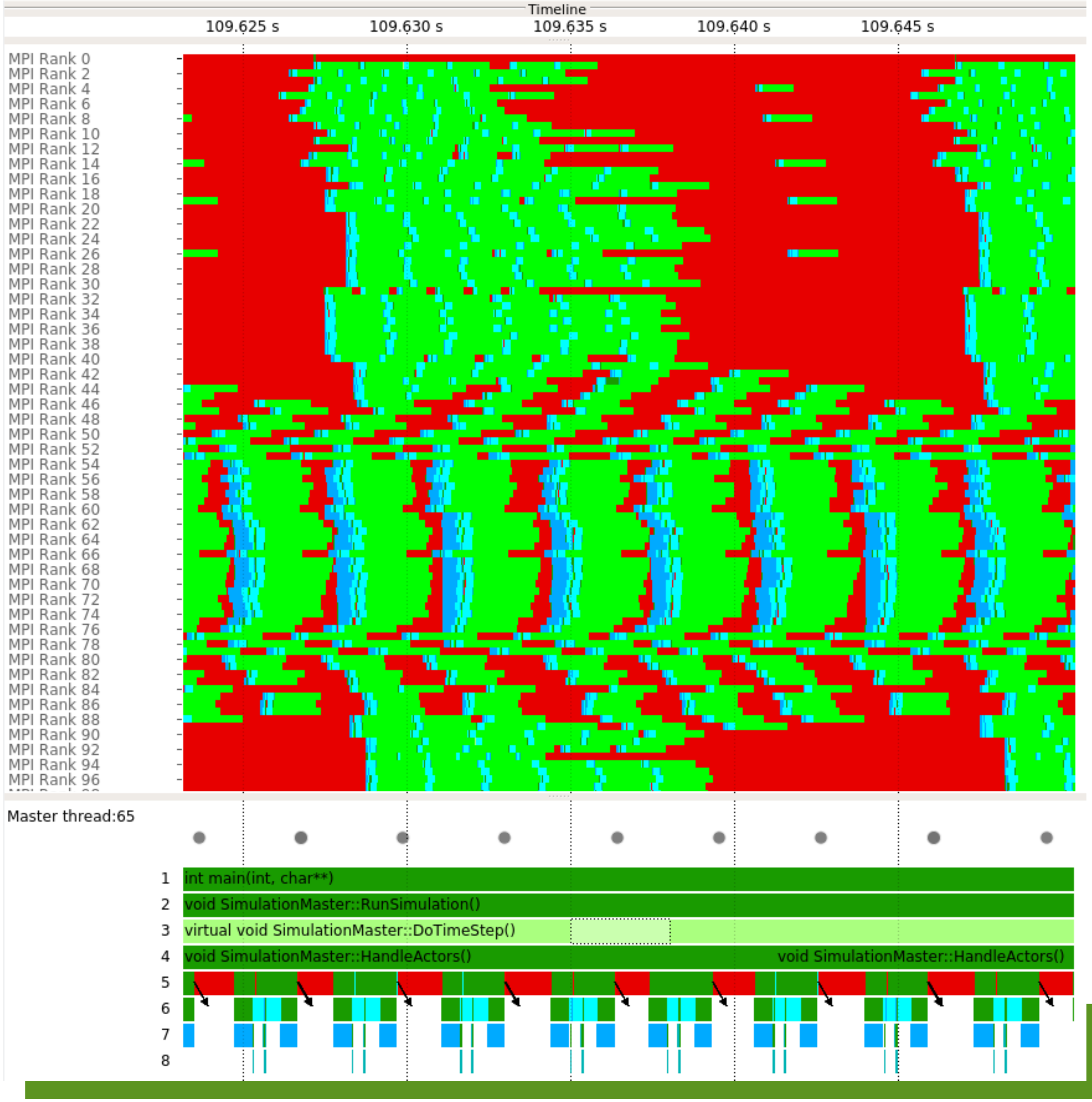
- 32 nodes: 129 MPI processes driving 128 GPUs
  - monitor rank 0 not participating in simulation
  - reader ranks 1 &2 distribute simulation geometry, then participate in simulation
- SimulationMaster class constructor/destructor methods initialisation and finalisation
- FOA is RunSimulation *DoTimeStep* routine
  - 2000 simulation steps: approx. 101-113 seconds
  - CUDA kernel offloads to dedicated GPU
  - property file writing after each 1000 steps
  - *IncompressibilityChecker* each 200 steps (mostly)
    - only lowest 10 simulation ranks
    - **or** highest 14 simulation ranks (32 nodes)

HemeLB_GPU execution timeline: FOA zoom (20 steps)



- 32 nodes: 129 MPI processes driving 128 GPUs
  - monitor rank 0 not participating in simulation
  - reader ranks 1 &2 distribute simulation geometry, then participate in simulation
- SimulationMaster class constructor/destructor methods initialisation and finalisation
- FOA is RunSimulation *DoTimeStep* routine
  - 2000 simulation steps: approx. 101-113 seconds
  - CUDA kernel offloads to dedicated GPU
  - property file writing after each 1000 steps
  - *IncompressibilityChecker* each 200 steps (mostly)
  - interior ranks (54-76) have "uniform" steps
    - others blocked every 6 steps, waiting for communication

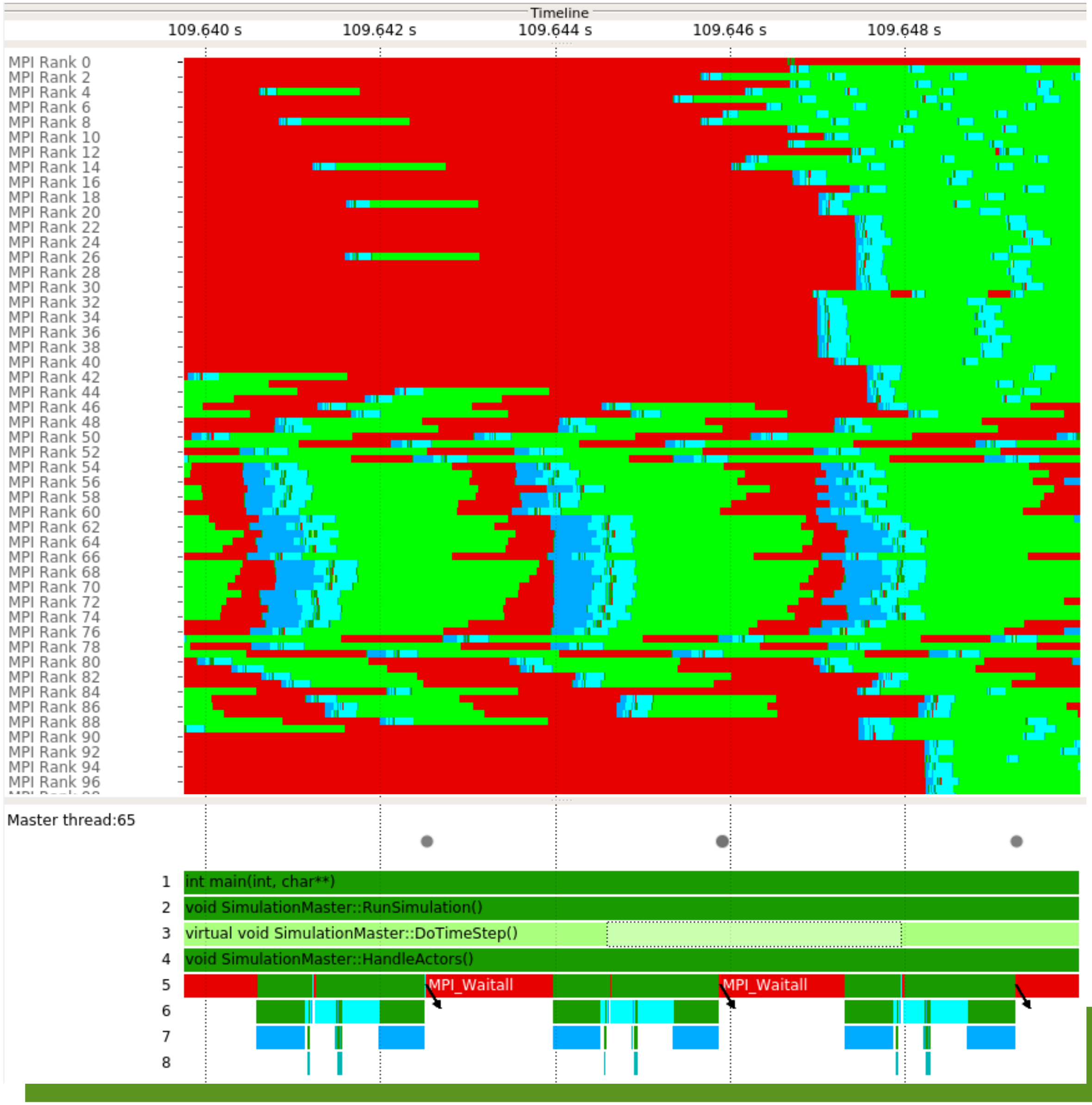
HemeLB_GPU execution timeline: FOA zoom (8 steps)



1 int main(int, char**)
2 void SimulationMaster::RunSimulation()
3 virtual void SimulationMaster::DoTimeStep()
4 void SimulationMaster::HandleActors() void SimulationMaster::HandleActors()

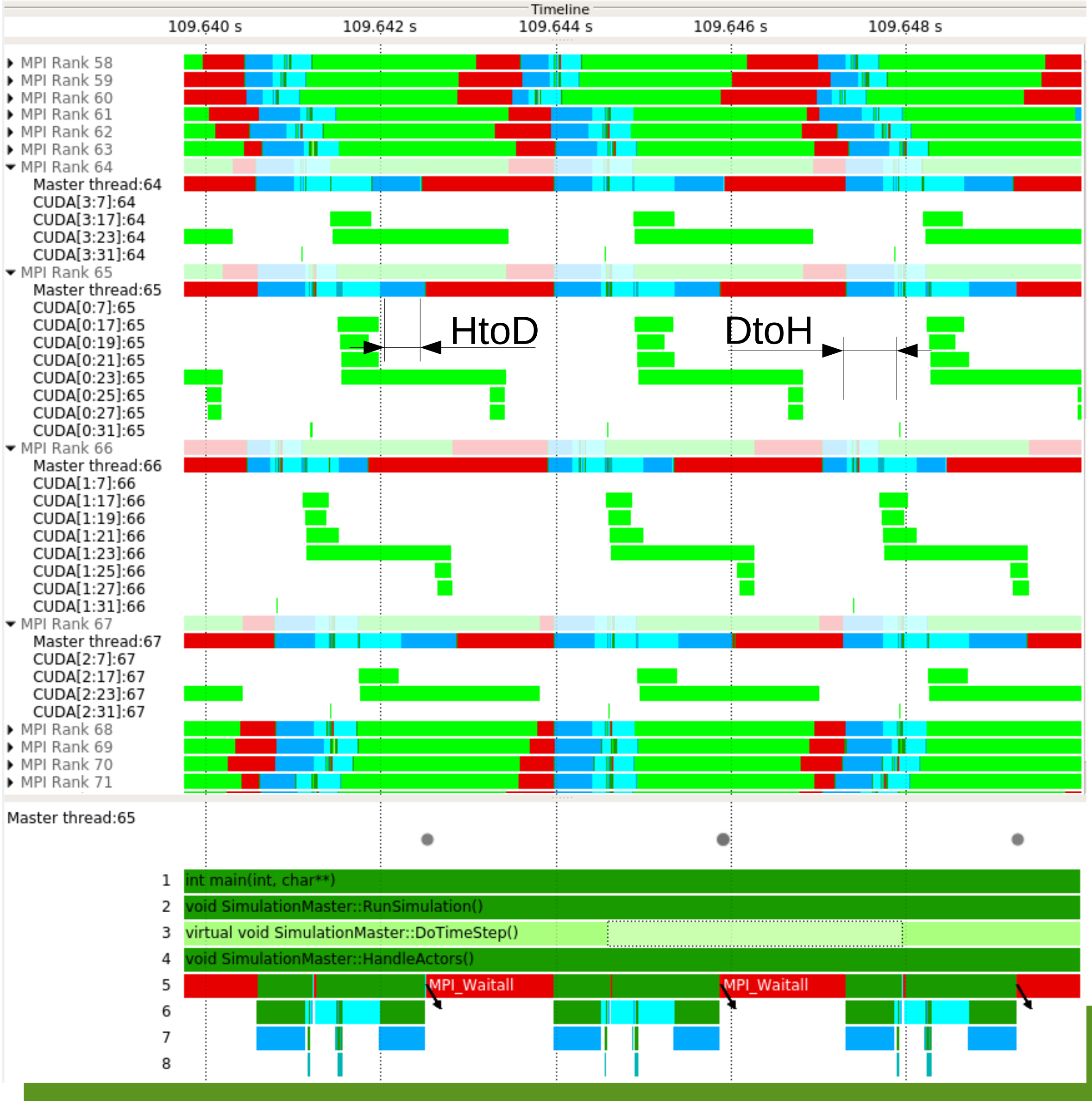
- 32 nodes: 129 MPI processes driving 128 GPUs
  - monitor rank 0 not participating in simulation
  - reader ranks 1 &2 distribute simulation geometry, then participate in simulation
- SimulationMaster class constructor/destructor methods initialisation and finalisation
- FOA is RunSimulation *DoTimeStep* routine
  - 2000 simulation steps: approx. 101-113 seconds
  - CUDA kernel offloads to dedicated GPU
  - property file writing after each 1000 steps
  - *IncompressibilityChecker* each 200 steps (mostly)
    - interior ranks (54-76) have “uniform” steps
    - others blocked every 6 steps, waiting for communication

HemeLB_GPU execution timeline: FOA zoom (3 steps)



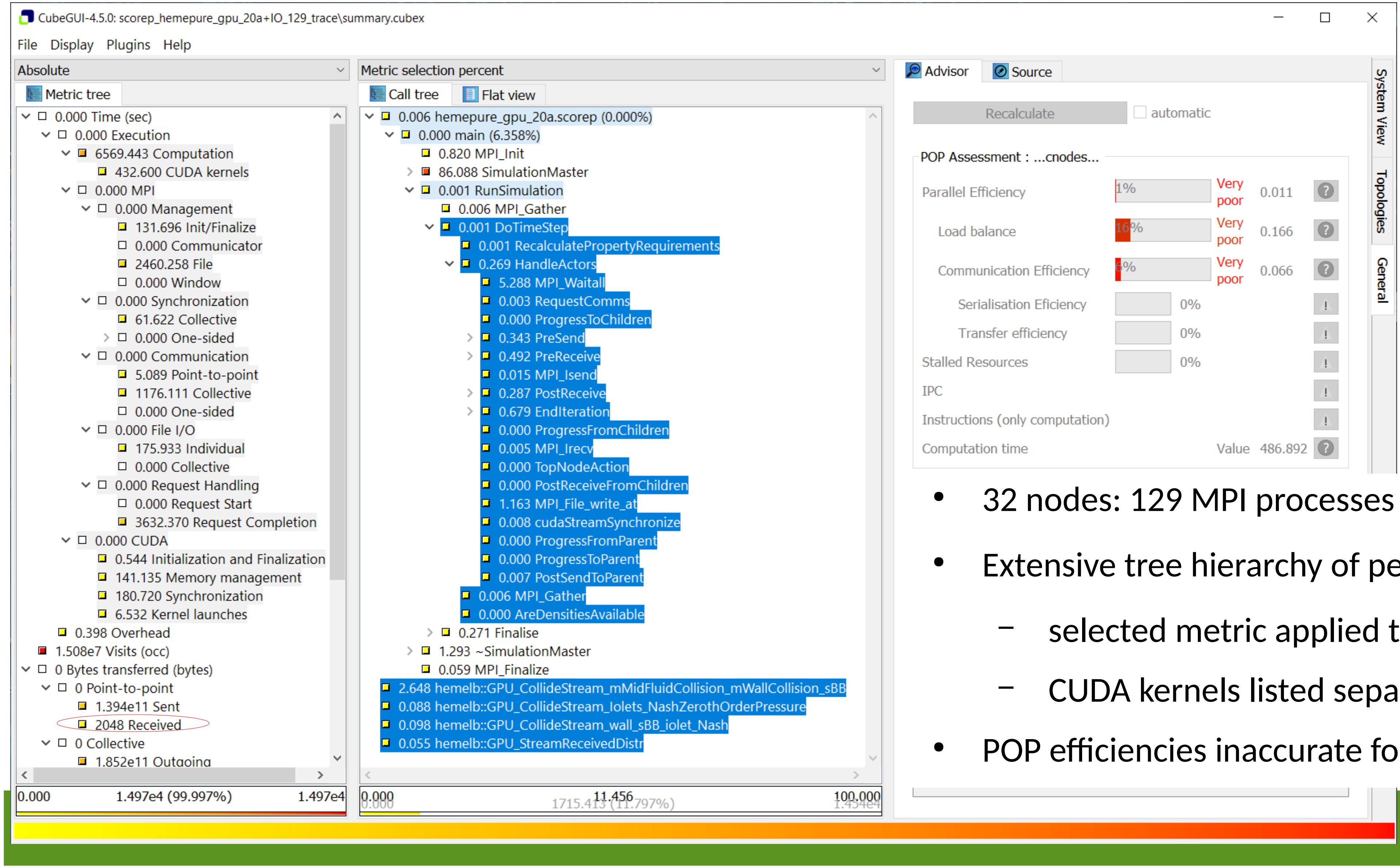
- 32 nodes: 129 MPI processes driving 128 GPUs
  - monitor rank 0 not participating in simulation
  - reader ranks 1 &2 distribute simulation geometry, then participate in simulation
- SimulationMaster class constructor/destructor methods initialisation and finalisation
- FOA is RunSimulation *DoTimeStep* routine
  - 2000 simulation steps: approx. 101-113 seconds
  - CUDA kernel offloads to dedicated GPU
  - property file writing after each 1000 steps
  - *IncompressibilityChecker* each 200 steps (mostly)
    - interior ranks (54-76) have “uniform” steps
    - others blocked every 6 steps, waiting for communication

HemeLB_GPU execution timeline: FOA zoom (3 steps) showing CUDA streams



- 32 nodes: 129 MPI processes driving 128 GPUs
  - monitor rank 0 not participating in simulation
  - reader ranks 1 &2 distribute simulation geometry, then participate in simulation
- SimulationMaster class constructor/destructor methods initialisation and finalisation
- FOA is RunSimulation *DoTimeStep* routine
  - 2000 simulation steps: approx. 101-113 seconds
  - CUDA kernel offloads to dedicated GPU
    - some ranks/GPUs have additional kernels (to process iolets) executed concurrently
    - critical PostReceive *cuMemcpyDtoHAsync* in Read_DistrFunctions_CPU_to_GPU_totalSharedFs
      - no longer overlapped with kernels

HemeLB_GPU profile



CubeGUI-4.5.0: scorep_hemepure_gpu_20a+IO_129_trace\summary.cubex

- 0.000 Time (sec)
  - 0.000 Execution
    - 6569.443 Computation
      - 432.600 CUDA kernels
    - 0.000 MPI
      - 0.000 Management
        - 131.696 Init/Finalize
        - 0.000 Communicator
        - 2460.258 File
        - 0.000 Window
      - 0.000 Synchronization
        - 61.622 Collective
        - 0.000 One-sided
      - 0.000 Communication
        - 5.089 Point-to-point
        - 1176.111 Collective
        - 0.000 One-sided
      - 0.000 File I/O
        - 175.933 Individual
        - 0.000 Collective
      - 0.000 Request Handling
        - 0.000 Request Start
        - 3632.370 Request Completion
    - 0.000 CUDA
      - 0.544 Initialization and Finalization
      - 141.135 Memory management
      - 180.720 Synchronization
      - 6.532 Kernel launches
  - 0.398 Overhead
- 1.508e7 Visits (occ)
- 0 Bytes transferred (bytes)
  - 0 Point-to-point
    - 1.394e11 Sent
    - 2048 Received
  - 0 Collective
    - 1.852e11 Outgoing

0.000 1.497e4 (99.997%) 1.497e4

- 0.006 hemepure_gpu_20a.scorep (0.000%)
  - 0.000 main (6.358%)
    - 0.820 MPI_Init
    - 86.088 SimulationMaster
    - 0.001 RunSimulation
      - 0.006 MPI_Gather
      - 0.001 DoTimeStep
        - 0.001 RecalculatePropertyRequirements
        - 0.269 HandleActors
          - 5.288 MPI_Waitall
          - 0.003 RequestComms
          - 0.000 ProgressToChildren
          - 0.343 PreSend
          - 0.492 PreReceive
          - 0.015 MPI_Isend
          - 0.287 PostReceive
          - 0.679 EndIteration
          - 0.000 ProgressFromChildren
          - 0.005 MPI_Irecv
          - 0.000 TopNodeAction
          - 0.000 PostReceiveFromChildren
          - 1.163 MPI_File_write_at
          - 0.008 cudaStreamSynchronize
          - 0.000 ProgressFromParent
          - 0.000 ProgressToParent
          - 0.007 PostSendToParent
        - 0.006 MPI_Gather
        - 0.000 AreDensitiesAvailable
      - 0.271 Finalize
    - 1.293 ~SimulationMaster
    - 0.059 MPI_Finalize
- 2.648 hemelb::GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB
- 0.088 hemelb::GPU_CollideStream_Iolets_NashZerothOrderPressure
- 0.098 hemelb::GPU_CollideStream_wall_sBB_iolet_Nash
- 0.055 hemelb::GPU_StreamReceivedDistr

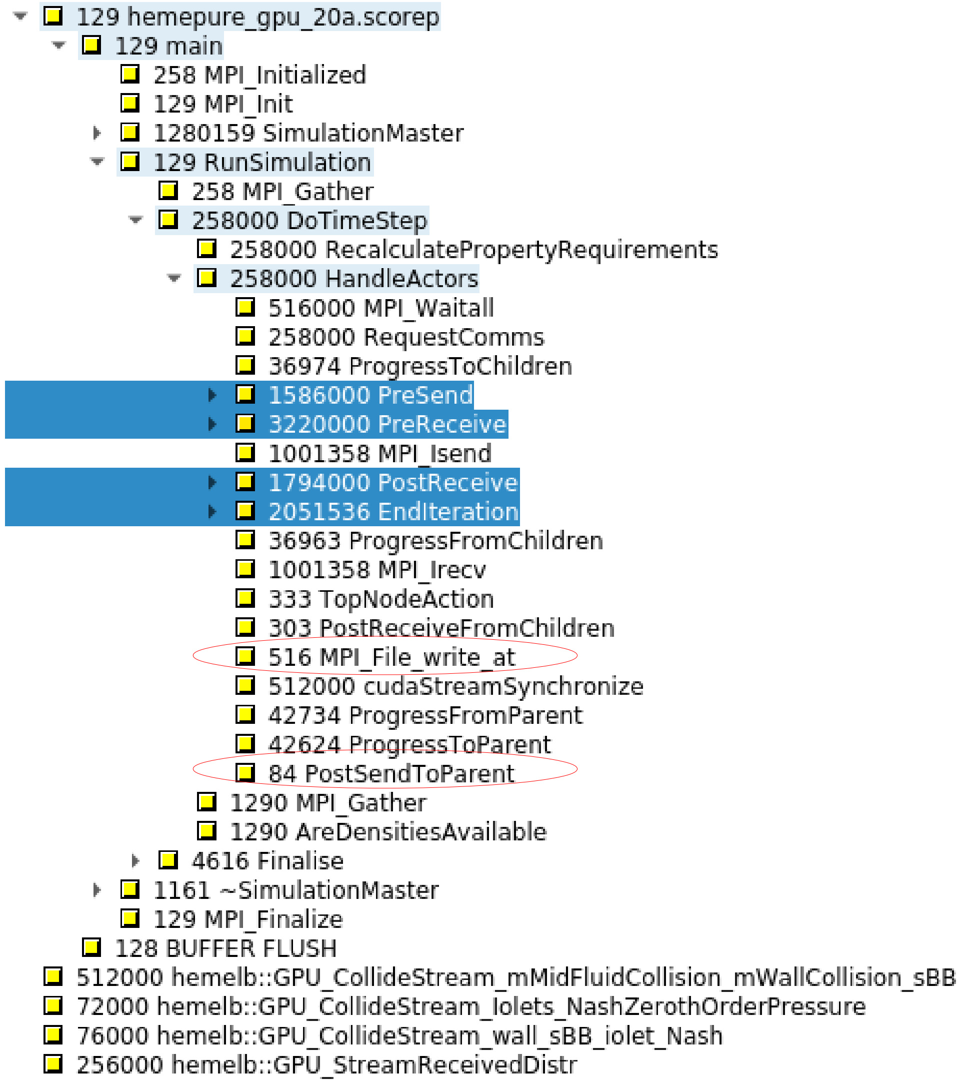
0.000 11.456 100.000
0.000 1715.413 (11.797%) 1.454e4

POP Assessment : ...cnodes...

Parallel Efficiency	1%	Very poor	0.011	?
Load balance	16%	Very poor	0.166	?
Communication Efficiency	6%	Very poor	0.066	?
Serialisation Eficiency	0%			!
Transfer efficiency	0%			!
Stalled Resources	0%			!
IPC				!
Instructions (only computation)				!
Computation time		Value	486.892	?

- 32 nodes: 129 MPI processes driving 128 GPUs
- Extensive tree hierarchy of performance metrics
  - selected metric applied to routines in call-tree
  - CUDA kernels listed separately at bottom
- POP efficiencies inaccurate for CPU+GPU hybrid!

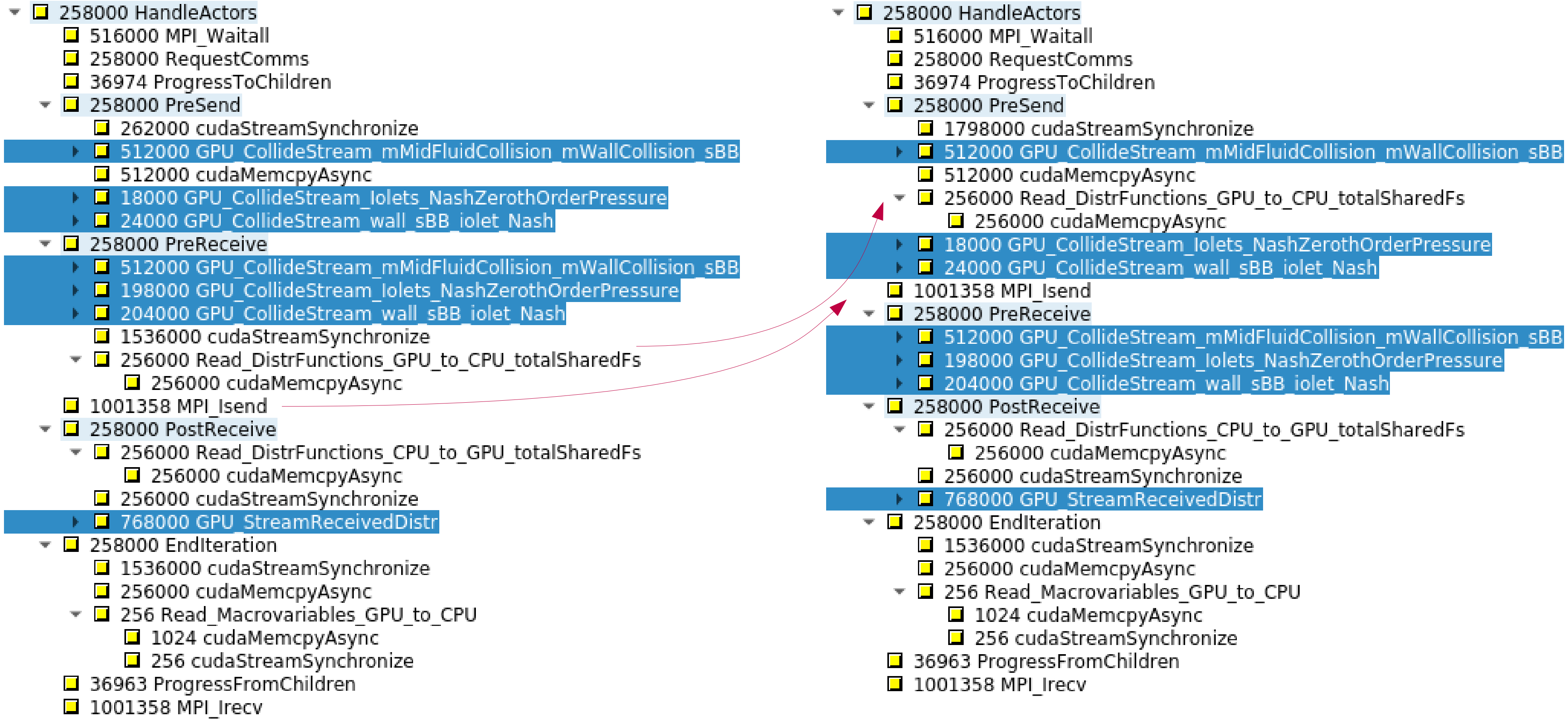
HemeLB_GPU code structure and Focus of Analysis (FOA) DoTimeStep



```
▾ ■ 129 hemepure_gpu_20a.scorep
  ▾ ■ 129 main
      ■ 258 MPI_Initialized
      ■ 129 MPI_Init
    ▸ ■ 1280159 SimulationMaster
    ▾ ■ 129 RunSimulation
        ■ 258 MPI_Gather
      ▾ ■ 258000 DoTimeStep
          ■ 258000 RecalculatePropertyRequirements
        ▾ ■ 258000 HandleActors
            ■ 516000 MPI_Waitall
            ■ 258000 RequestComms
            ■ 36974 ProgressToChildren
          ▸ ■ 1586000 PreSend
          ▸ ■ 3220000 PreReceive
            ■ 1001358 MPI_Isend
          ▸ ■ 1794000 PostReceive
          ▸ ■ 2051536 EndIteration
            ■ 36963 ProgressFromChildren
            ■ 1001358 MPI_Irecv
            ■ 333 TopNodeAction
            ■ 303 PostReceiveFromChildren
            ■ 516 MPI_File_write_at
            ■ 512000 cudaStreamSynchronize
            ■ 42734 ProgressFromParent
            ■ 42624 ProgressToParent
            ■ 84 PostSendToParent
          ■ 1290 MPI_Gather
          ■ 1290 AreDensitiesAvailable
      ▸ ■ 4616 Finalise
    ▸ ■ 1161 ~SimulationMaster
      ■ 129 MPI_Finalize
    ■ 128 BUFFER FLUSH
■ 512000 hemelb::GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB
■ 72000 hemelb::GPU_CollideStream_Iolets_NashZerothOrderPressure
■ 76000 hemelb::GPU_CollideStream_wall_sBB_iolet_Nash
■ 256000 hemelb::GPU_StreamReceivedDistr
```

- 32 nodes: 129 MPI processes driving 128 GPUs
  - v1.20a with file writing, 2000 time-steps
- SimulationMaster class constructor/destructor methods initialisation and finalisation
- FOA is RunSimulation *DoTimeStep* routine
  - loops through key actions
    - PreSend, PreReceive, Send, PostReceive, EndIteration
      - (a) Send after PreReceive
      - (b) Send before PreReceive
    - plus periodic MPI file writing by all processes
    - and occasional *IncompressibilityChecker* by subset
  - launches CUDA kernels on GPUs
    - execute asynchronously on specific CUDA streams
    - 4 distinct CUDA kernels

Simulation loop: 20a 20b



```
258000 HandleActors
    516000 MPI_Waitall
    258000 RequestComms
    36974 ProgressToChildren
    258000 PreSend
        262000 cudaStreamSynchronize
        512000 GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB
        512000 cudaMemcpyAsync
        18000 GPU_CollideStream_Iolets_NashZerothOrderPressure
        24000 GPU_CollideStream_wall_sBB_iolet_Nash
    258000 PreReceive
        512000 GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB
        198000 GPU_CollideStream_Iolets_NashZerothOrderPressure
        204000 GPU_CollideStream_wall_sBB_iolet_Nash
        1536000 cudaStreamSynchronize
        256000 Read_DistrFunctions_GPU_to_CPU_totalSharedFs
            256000 cudaMemcpyAsync
    1001358 MPI_Isend
    258000 PostReceive
        256000 Read_DistrFunctions_CPU_to_GPU_totalSharedFs
            256000 cudaMemcpyAsync
        256000 cudaStreamSynchronize
        768000 GPU_StreamReceivedDistr
    258000 EndIteration
        1536000 cudaStreamSynchronize
        256000 cudaMemcpyAsync
        256 Read_Macrovariables_GPU_to_CPU
            1024 cudaMemcpyAsync
            256 cudaStreamSynchronize
    36963 ProgressFromChildren
    1001358 MPI_Irecv
```

```
258000 HandleActors
    516000 MPI_Waitall
    258000 RequestComms
    36974 ProgressToChildren
    258000 PreSend
        1798000 cudaStreamSynchronize
        512000 GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB
        512000 cudaMemcpyAsync
        256000 Read_DistrFunctions_GPU_to_CPU_totalSharedFs
            256000 cudaMemcpyAsync
        18000 GPU_CollideStream_Iolets_NashZerothOrderPressure
        24000 GPU_CollideStream_wall_sBB_iolet_Nash
    1001358 MPI_Isend
    258000 PreReceive
        512000 GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB
        198000 GPU_CollideStream_Iolets_NashZerothOrderPressure
        204000 GPU_CollideStream_wall_sBB_iolet_Nash
    258000 PostReceive
        256000 Read_DistrFunctions_CPU_to_GPU_totalSharedFs
            256000 cudaMemcpyAsync
        256000 cudaStreamSynchronize
        768000 GPU_StreamReceivedDistr
    258000 EndIteration
        1536000 cudaStreamSynchronize
        256000 cudaMemcpyAsync
        256 Read_Macrovariables_GPU_to_CPU
            1024 cudaMemcpyAsync
            256 cudaStreamSynchronize
    36963 ProgressFromChildren
    1001358 MPI_Irecv
```

HemeLB_GPU Simulation CUDA kernels/streams

- PreSend
  – cudaStreamSynchronize [31]
  – [17] ***GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB***
  – 2x cudaMemcpyAsync / cuMemcpy**HtoD**Async_v2 + cudaStreamSynchronize [29,30]
  – [19,20] *GPU_CollideStream_Iolets_NashZerothOrderPressure*
  – [21,22] *GPU_CollideStream_wall_sBB_iolet_Nash*

local partition edge

- PreReceive
  – [23] ***GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB***
  – cudaStreamSynchronize [17,18,19,20,21,22]
  – Read_DistrFunctions_GPU_to_CPU_totalSharedFs / cuMemcpy**DtoH**Async_v2 [35]
  – cudaStreamSynchronize [35]
  – [25,26] *GPU_CollideStream_Iolets_NashZerothOrderPressure*
  – [27,28] *GPU_CollideStream_wall_sBB_iolet_Nash*

local partition interior

- PostReceive
  – Read_DistrFunctions_CPU_to_GPU_totalSharedFs / cuMemcpy**HtoD**Async_v2 [33]
  – cudaStreamSynchronize [33]
  – [31] *GPU_StreamReceivedDistr*

- EndIteration
  – cudaStreamSynchronize [23,24,25,26,27,28]
  – cudaMemcpyAsync / cuMemcpy**DtoD**Async_v2 [31]
  – Read_Macrovariables_GPU_to_CPU / cuMemcpy**DtoH**Async_v2 + cudaStreamSynchronize [34]

HemeLB_GPU CUDA kernels/streams

Stream 7: Memcpy HtoD [sync]

Stream 17: **PreSend** kernel1 hemelb::GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB

Stream 19: **PreSend** kernel2 hemelb::GPU_CollideStream_Iolets_NashZerothOrderPressure (inlet)

Stream 20: **PreSend** kernel2 hemelb::GPU_CollideStream_Iolets_NashZerothOrderPressure (outlet)

Stream 21: **PreSend** kernel3 hemelb::GPU_CollideStream_wall_sBB_iolet_Nash (inlet)

Stream 22: **PreSend** kernel3 hemelb::GPU_CollideStream_wall_sBB_iolet_Nash (outlet)

Stream 23: **PreReceive** kernel1 hemelb::GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB

Stream 25: **PreReceive** kernel2 hemelb::GPU_CollideStream_Iolets_NashZerothOrderPressure (inlet)

Stream 26: **PreReceive** kernel2 hemelb::GPU_CollideStream_Iolets_NashZerothOrderPressure (outlet)

Stream 27: **PreReceive** kernel3 hemelb::GPU_CollideStream_wall_sBB_iolet_Nash (inlet)

Stream 28: **PreReceive** kernel3 hemelb::GPU_CollideStream_wall_sBB_iolet_Nash (outlet)

Stream 29: ~~Memcpy HtoD [async] (stream_ghost_dens_inlet)~~

Stream 30: ~~Memcpy HtoD [async] (stream_ghost_dens_outlet)~~

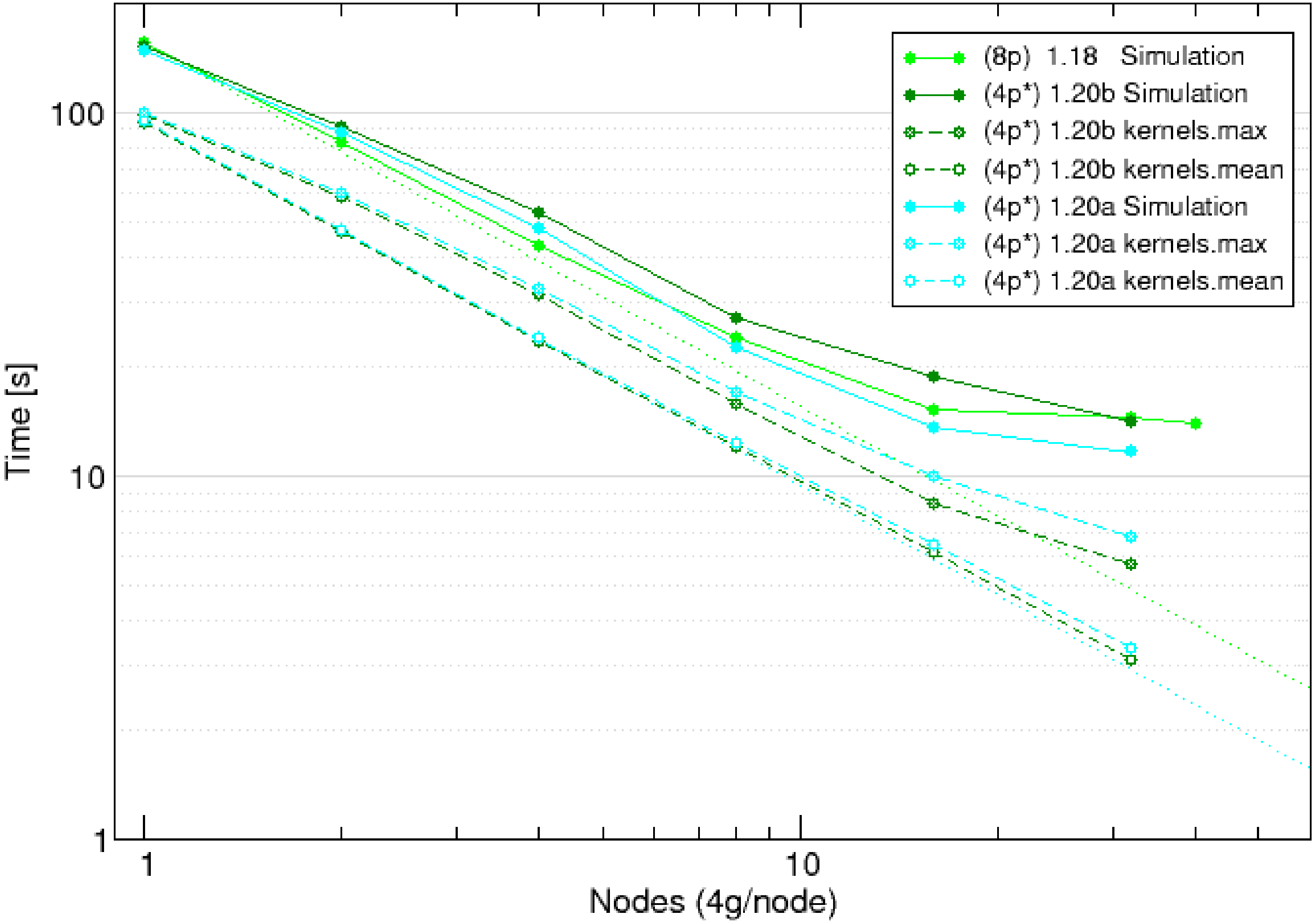
Stream 31: ~~Memcpy DtoD [async]~~ + **PostReceive** kernel4 hemelb::GPU_StreamReceivedDistr

Stream 33: ~~Memcpy HtoD [async] (stream_memCpy_CPU_GPU_domainEdge)~~

Stream 34: ~~Memcpy DtoH [async] (stream_Read_Data_GPU_Dens)~~

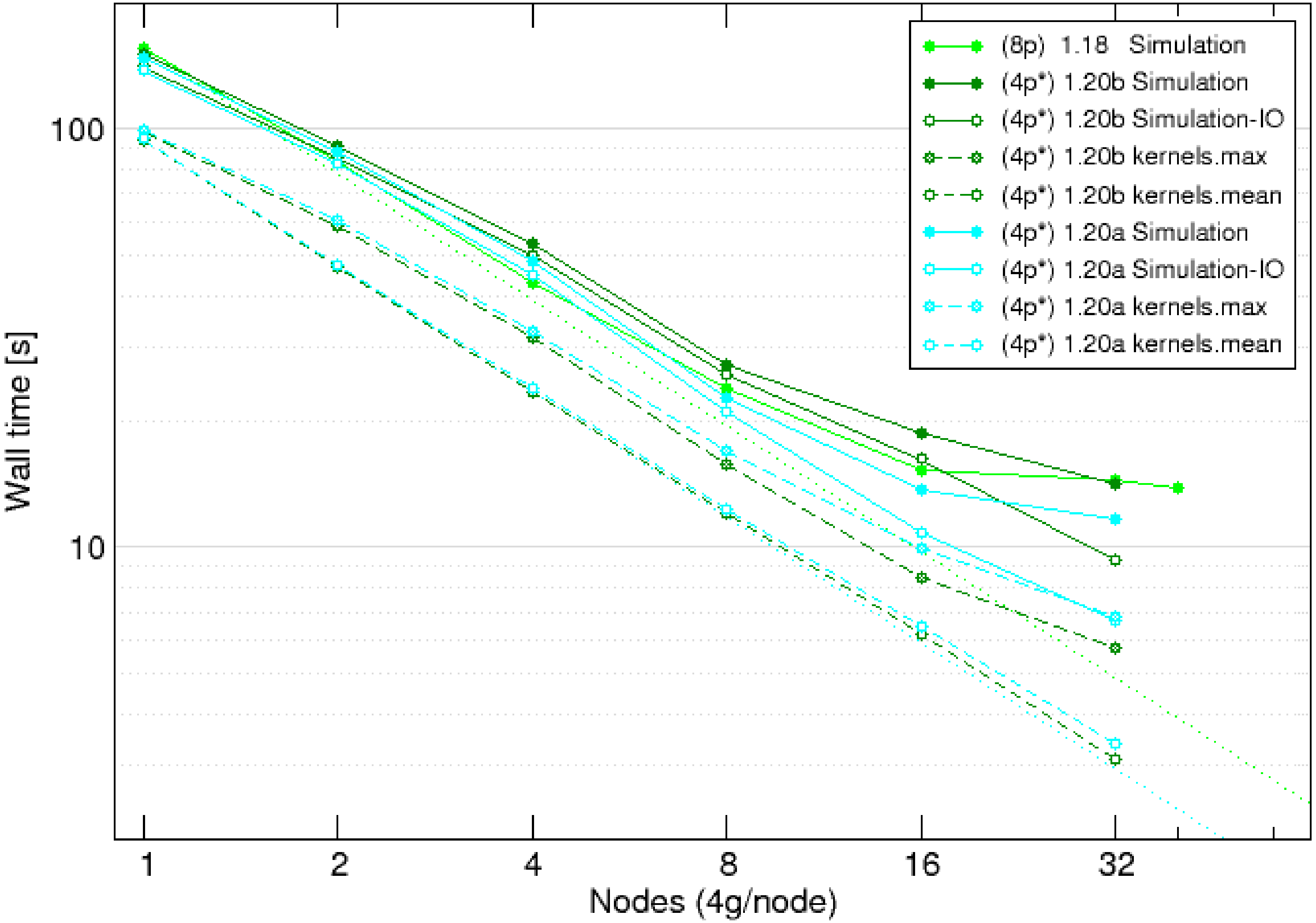
Stream 35: ~~Memcpy DtoH [async] (mNet_cuda_stream)~~

HemeLB_GPU Simulation strong scaling (with intermediate file writing)



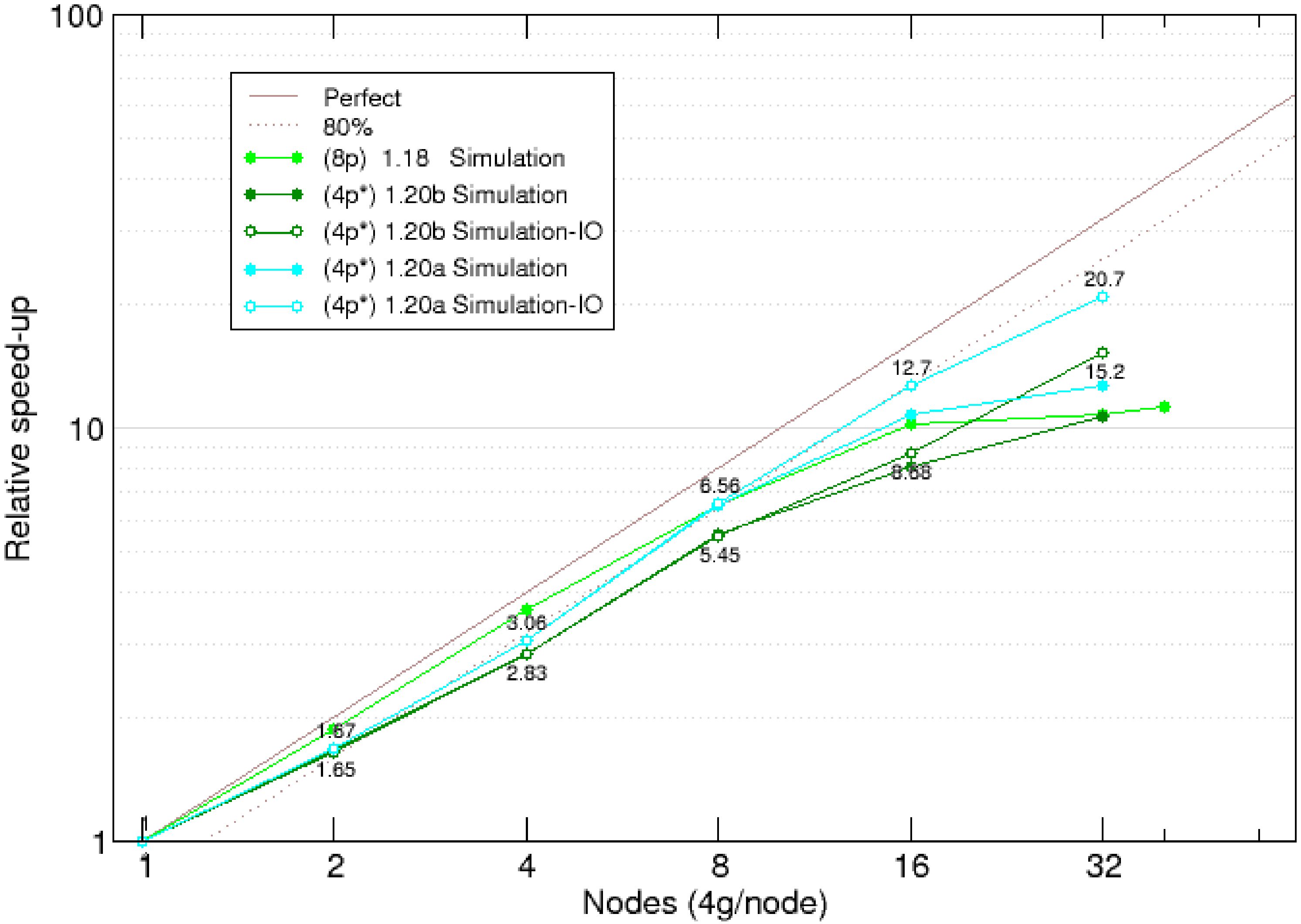
- Reference execution v1.18 with 8ppn
  - multiple processes offloading GPU kernels generally unproductive
- Comparison of v1.20a & v1.20b (4ppn)
  - v1.20a generally better
- CUDA kernels on GPUs
  - less than half of Simulation time (therefore GPUs mostly idle)
  - total kernel time scales very well (0.93 scaling efficiency)
  - load balance deteriorates (0.95 for 1 node, 0.50 for 32 nodes)
  - similar for both versions

HemeLB_GPU Simulation strong scaling (w/o file writing)



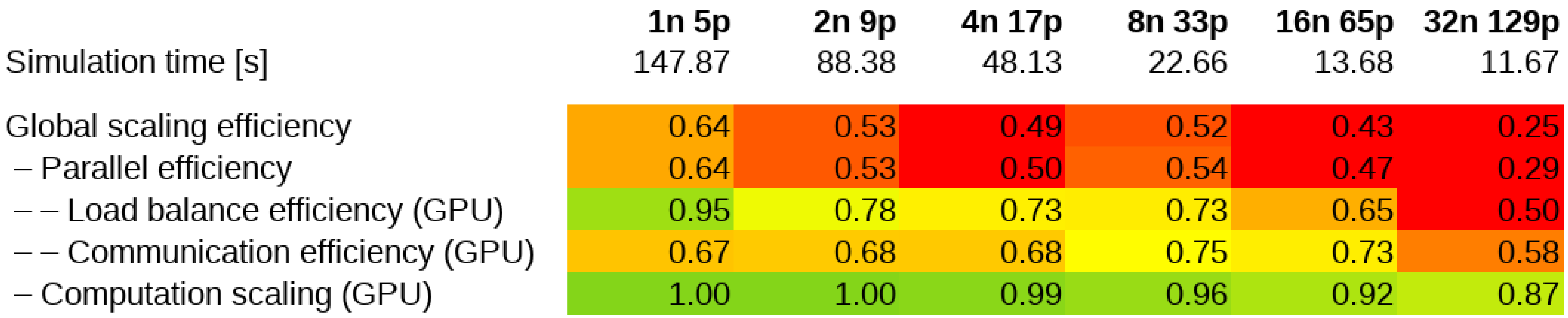
- File writing disabled during Simulation
  - previously every 1000 steps
- Scaling greatly improved
  - particularly beyond 8 nodes
- CUDA kernels on GPUs
  - essentially unaffected

HemeLB_GPU Simulation strong scaling (w/o file writing) speed-up

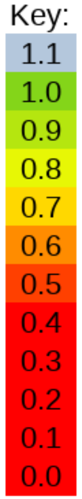


- File writing disabled during Simulation
  - previously every 1000 steps
- Speed-up improved
  - 21x for v1.20a on 32 nodes (65% scaling efficiency)
  - 80% scaling efficiency retained to 16 nodes
  - most of the loss of scaling occurs moving from shared-memory MPI within a single node to off-node communication via interconnect

HemeLB_GPU Simulation strong scaling efficiency (v1.20a with file writing)



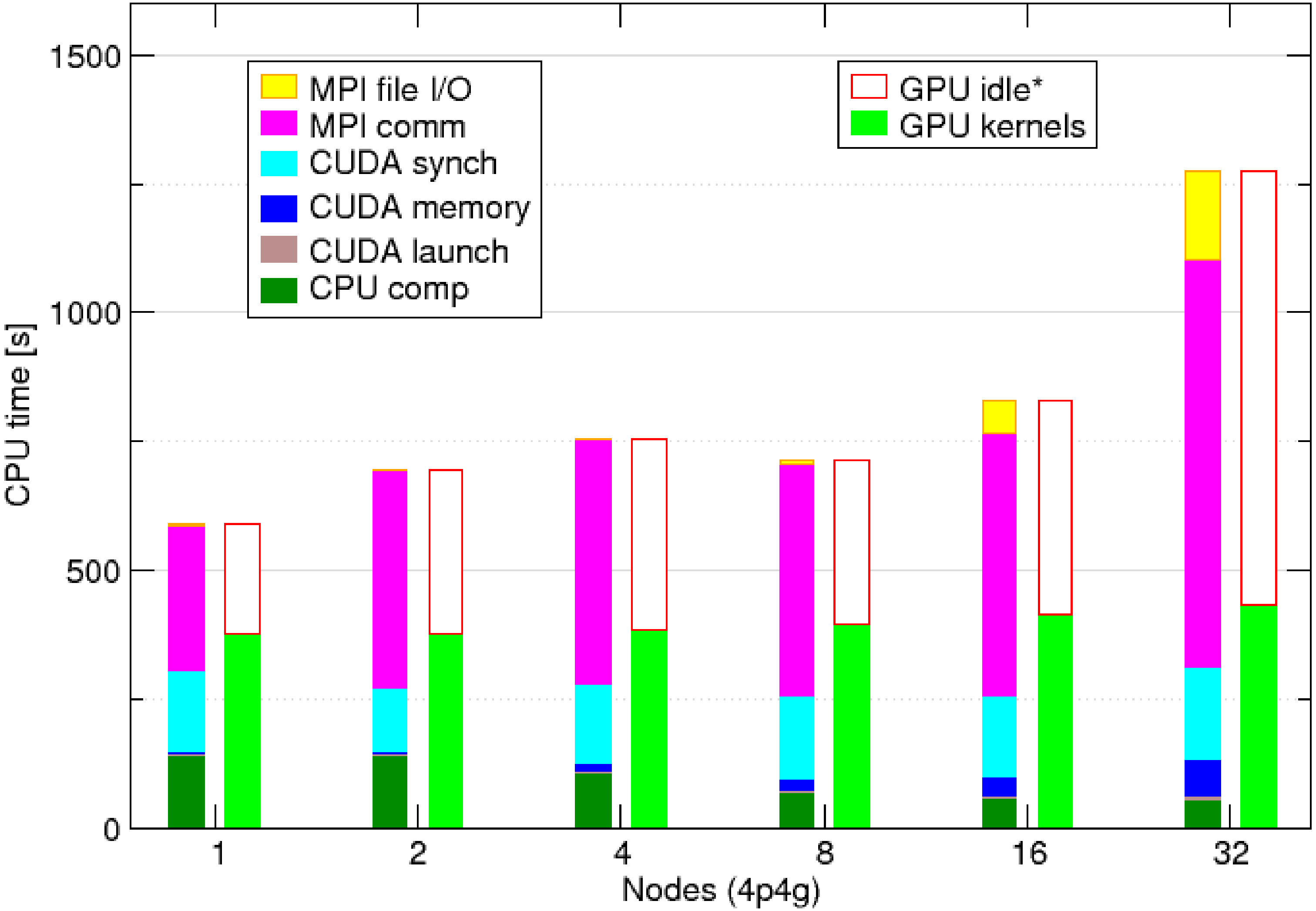
	1n 5p	2n 9p	4n 17p	8n 33p	16n 65p	32n 129p
Simulation time [s]	147.87	88.38	48.13	22.66	13.68	11.67
Global scaling efficiency	0.64	0.53	0.49	0.52	0.43	0.25
– Parallel efficiency	0.64	0.53	0.50	0.54	0.47	0.29
– – Load balance efficiency (GPU)	0.95	0.78	0.73	0.73	0.65	0.50
– – Communication efficiency (GPU)	0.67	0.68	0.68	0.75	0.73	0.58
– Computation scaling (GPU)	1.00	1.00	0.99	0.96	0.92	0.87



Key:
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

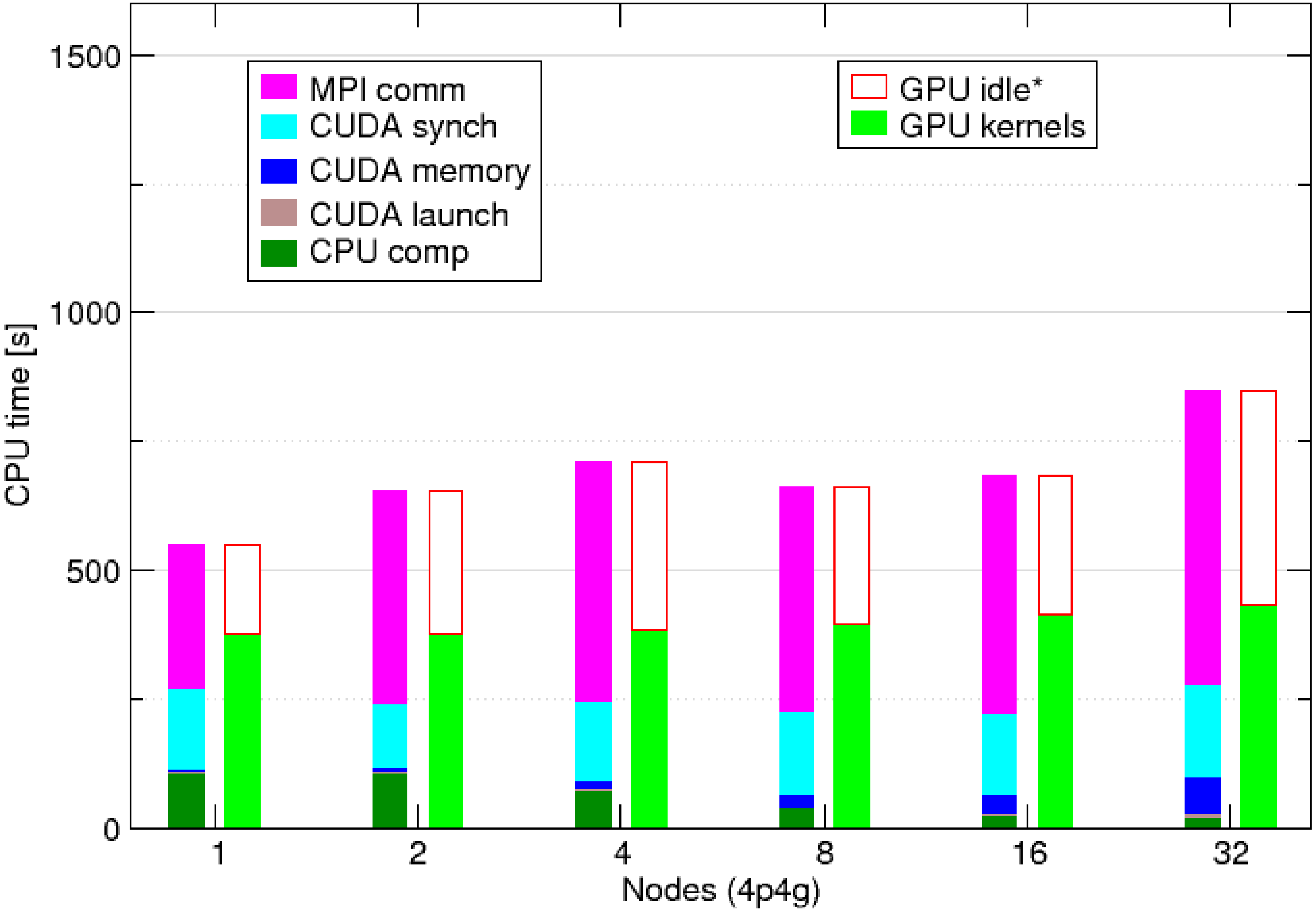
- Only considering GPUs (ignoring all CPU cores, 90% of which are completely unused)
- Parallel efficiency determined by load balance and communication (including file I/O)
- Single (quad-GPU) node already suffers significant communication inefficiency
  - but doesn’t degrade much as additional nodes are included
- Load balance of GPUs deteriorates progressively
- GPU computation scaling remains reasonably good

HemeLB_GPU Simulation time breakdown (v1.20a)



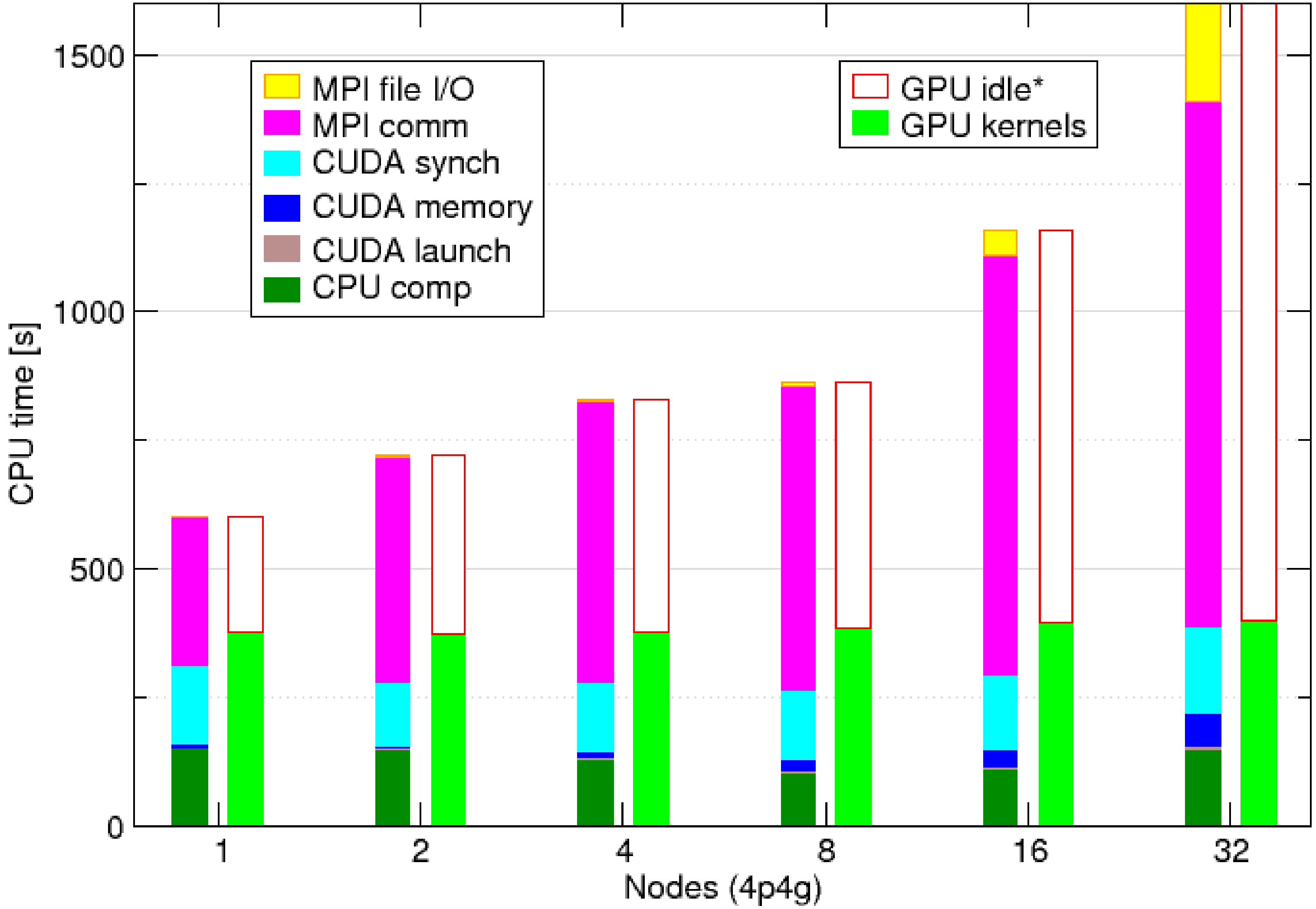
- CUDA kernels on GPUs
  - less than half of Simulation time (therefore GPUs mostly idle)
  - total kernel time scales very well (0.87 scaling efficiency)
- MPI processes on CPUs
  - computation time decreases
  - CUDA synchronization time fairly constant, but time for memory management increases somewhat
  - MPI communication time dominates, with much more time for file writing with 16 or more nodes

HemeLB_GPU Simulation time breakdown (v1.20a w/o file writing)



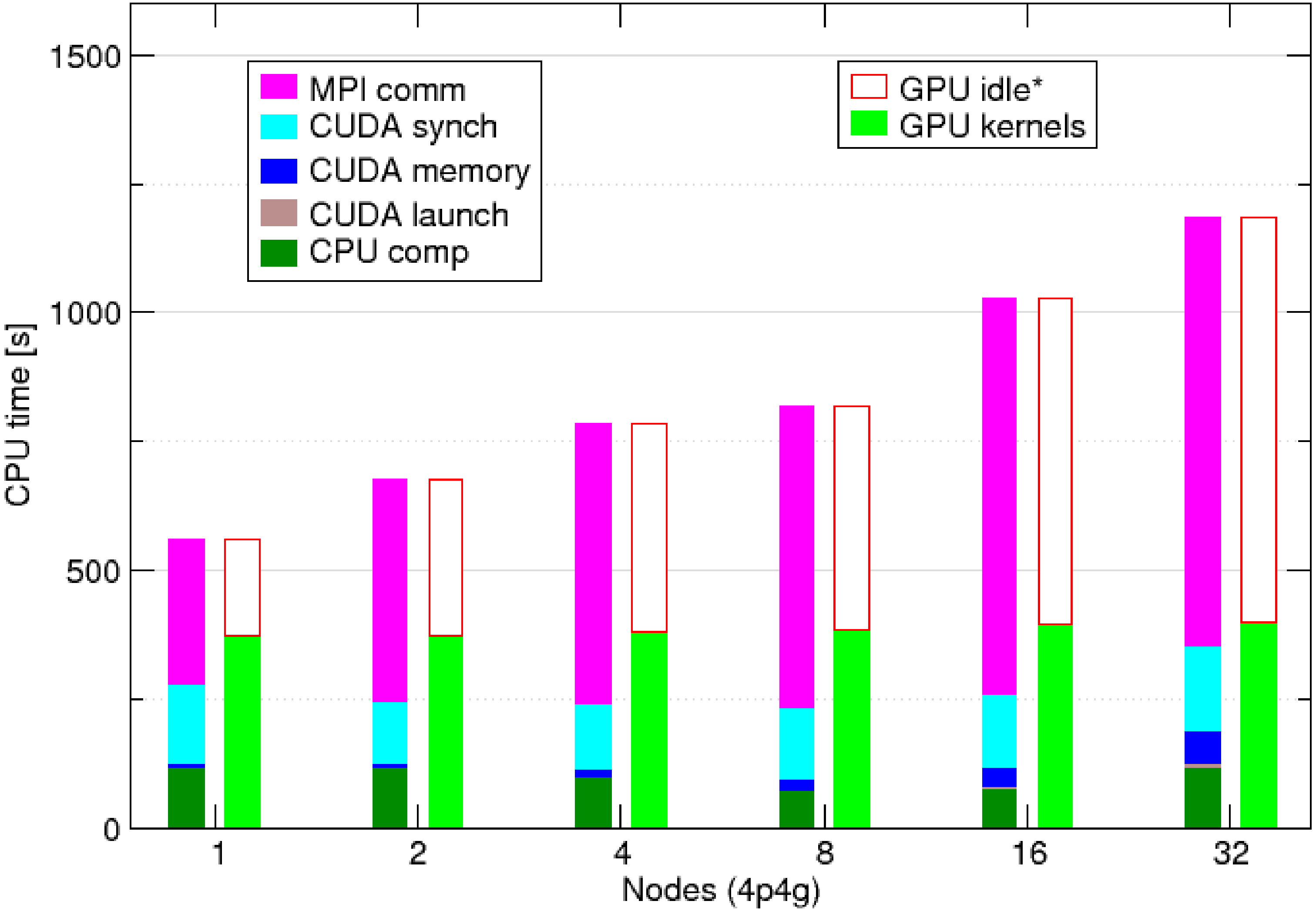
- Disabled Simulation file writing
  - no impact on GPU kernels, nor CUDA operations on CPU
  - reduced CPU computation & MPI communication (and no MPI file I/O)
- CUDA kernels on GPUs
  - reduced GPU idle time, but still significant (roughly half)
- MPI processes on CPUs
  - computation time decreases
  - MPI communication time dominates, at all scales but growing with scale

HemeLB_GPU Simulation time breakdown (v1.20b)



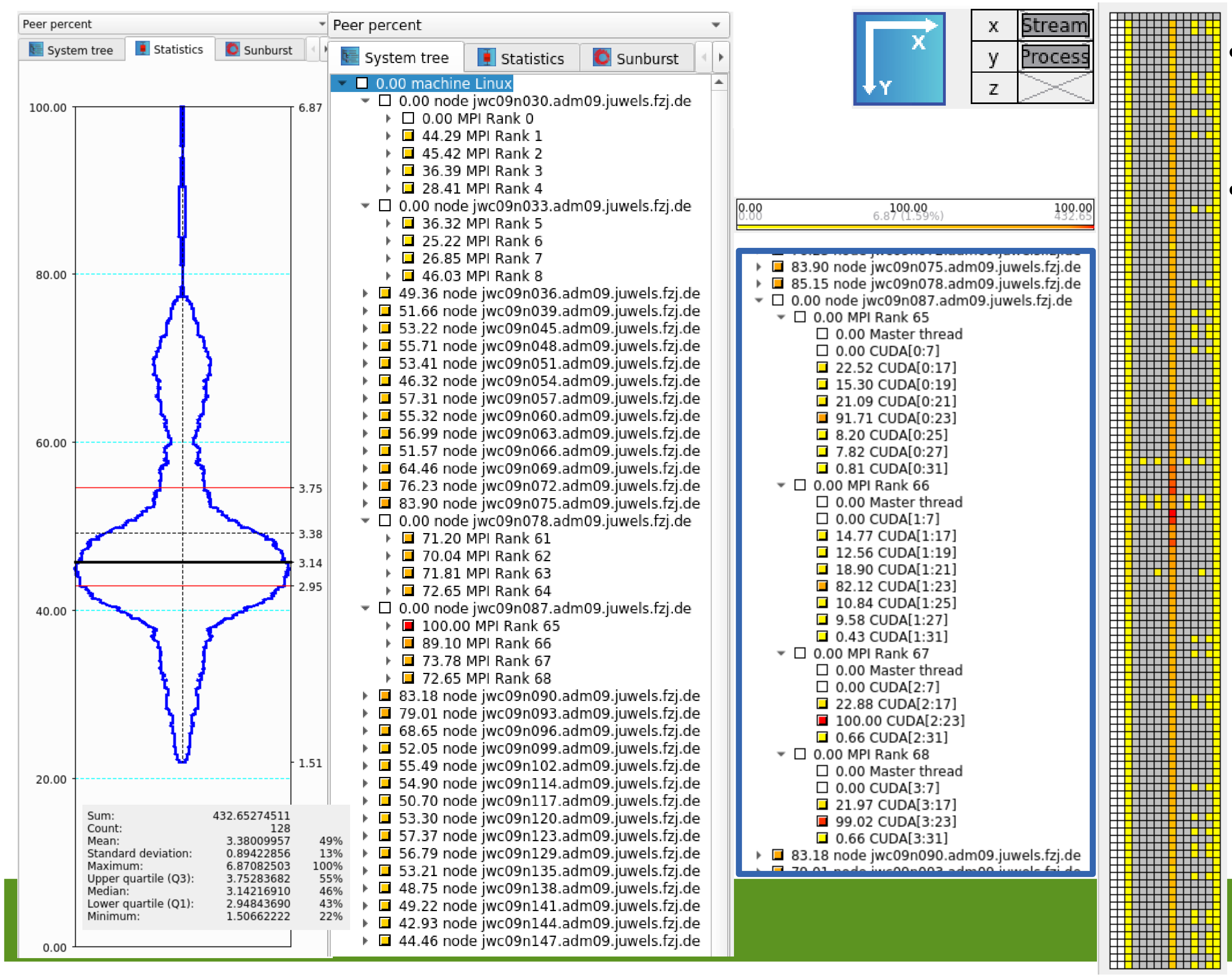
- CUDA kernels on GPUs
  - less than half of Simulation time (therefore GPUs mostly idle)
  - total kernel time scales very well (0.94 scaling efficiency)
- MPI processes on CPUs
  - computation time roughly constant
  - CUDA synchronization time fairly constant, but time for memory management increases somewhat
  - MPI communication time dominates, with much more time for file writing with 16 or more nodes

HemeLB_GPU Simulation time breakdown (v1.20b w/o file writing)



- Disabled Simulation file writing
  - no impact on GPU kernels, nor CUDA operations on CPU
  - reduced CPU computation & MPI communication (and no MPI file I/O)
- CUDA kernels on GPUs
  - reduced GPU idle time, but still significant (more than half)
- MPI processes on CPUs
  - computation time roughly constant
  - MPI communication time dominates, at all scales but growing with scale

HemeLB_GPU Simulation GPU time balance (v1.20a w/o file writing)



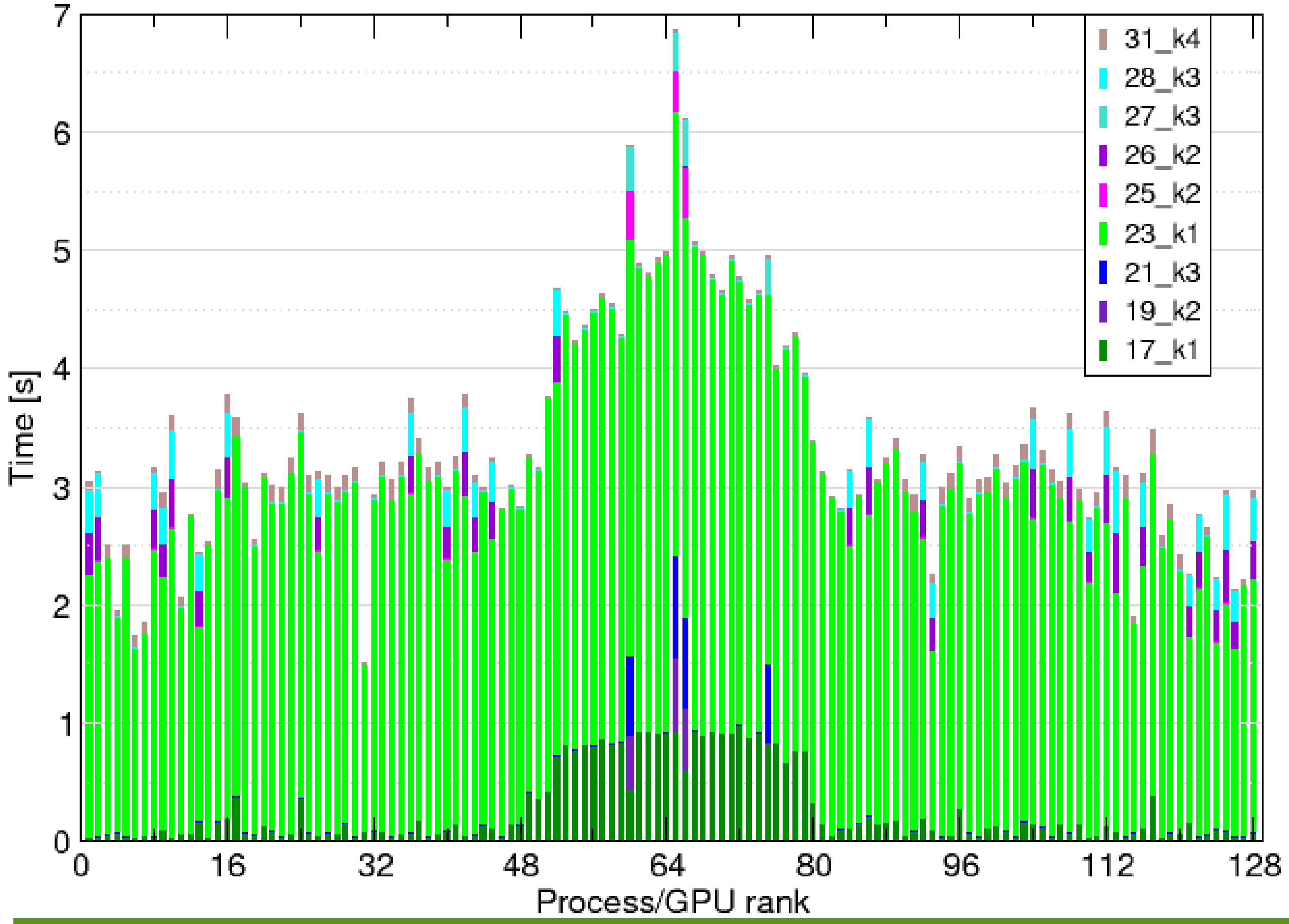
```
Sum:                        432.65274511
Count:                               128
Mean:                         3.38009957    49%
Standard deviation:           0.89422856    13%
Maximum:                      6.87082503   100%
Upper quartile (Q3):          3.75283682    55%
Median:                       3.14216910    46%
Lower quartile (Q1):          2.94843690    43%
Minimum:                      1.50662222    22%
```

```
▾ ☐ 0.00 machine Linux
  ▾ ☐ 0.00 node jwc09n030.adm09.juwels.fzj.de
    ▸ ☐ 0.00 MPI Rank 0
    ▸ 🟨 44.29 MPI Rank 1
    ▸ 🟨 45.42 MPI Rank 2
    ▸ 🟨 36.39 MPI Rank 3
    ▸ 🟨 28.41 MPI Rank 4
  ▾ ☐ 0.00 node jwc09n033.adm09.juwels.fzj.de
    ▸ 🟨 36.32 MPI Rank 5
    ▸ 🟨 25.22 MPI Rank 6
    ▸ 🟨 26.85 MPI Rank 7
    ▸ 🟨 46.03 MPI Rank 8
  ▸ 🟨 49.36 node jwc09n036.adm09.juwels.fzj.de
  ▸ 🟨 51.66 node jwc09n039.adm09.juwels.fzj.de
  ▸ 🟨 53.22 node jwc09n045.adm09.juwels.fzj.de
  ▸ 🟨 55.71 node jwc09n048.adm09.juwels.fzj.de
  ▸ 🟨 53.41 node jwc09n051.adm09.juwels.fzj.de
  ▸ 🟨 46.32 node jwc09n054.adm09.juwels.fzj.de
  ▸ 🟨 57.31 node jwc09n057.adm09.juwels.fzj.de
  ▸ 🟨 55.32 node jwc09n060.adm09.juwels.fzj.de
  ▸ 🟨 56.99 node jwc09n063.adm09.juwels.fzj.de
  ▸ 🟨 51.57 node jwc09n066.adm09.juwels.fzj.de
  ▸ 🟨 64.46 node jwc09n069.adm09.juwels.fzj.de
  ▸ 🟧 76.23 node jwc09n072.adm09.juwels.fzj.de
  ▸ 🟧 83.90 node jwc09n075.adm09.juwels.fzj.de
  ▾ ☐ 0.00 node jwc09n078.adm09.juwels.fzj.de
    ▸ 🟧 71.20 MPI Rank 61
    ▸ 🟧 70.04 MPI Rank 62
    ▸ 🟧 71.81 MPI Rank 63
    ▸ 🟧 72.65 MPI Rank 64
  ▾ ☐ 0.00 node jwc09n087.adm09.juwels.fzj.de
    ▸ 🟥 100.00 MPI Rank 65
    ▸ 🟧 89.10 MPI Rank 66
    ▸ 🟧 73.78 MPI Rank 67
    ▸ 🟧 72.65 MPI Rank 68
  ▸ 🟧 83.18 node jwc09n090.adm09.juwels.fzj.de
  ▸ 🟧 79.01 node jwc09n093.adm09.juwels.fzj.de
  ▸ 🟨 68.65 node jwc09n096.adm09.juwels.fzj.de
  ▸ 🟨 52.05 node jwc09n099.adm09.juwels.fzj.de
  ▸ 🟨 55.49 node jwc09n102.adm09.juwels.fzj.de
  ▸ 🟨 54.90 node jwc09n114.adm09.juwels.fzj.de
  ▸ 🟨 50.70 node jwc09n117.adm09.juwels.fzj.de
  ▸ 🟨 53.30 node jwc09n120.adm09.juwels.fzj.de
  ▸ 🟨 57.37 node jwc09n123.adm09.juwels.fzj.de
  ▸ 🟨 56.79 node jwc09n129.adm09.juwels.fzj.de
  ▸ 🟨 53.21 node jwc09n135.adm09.juwels.fzj.de
  ▸ 🟨 48.75 node jwc09n138.adm09.juwels.fzj.de
  ▸ 🟨 49.22 node jwc09n141.adm09.juwels.fzj.de
  ▸ 🟨 42.93 node jwc09n144.adm09.juwels.fzj.de
  ▸ 🟨 44.46 node jwc09n147.adm09.juwels.fzj.de
```

```
▸ 🟧 83.90 node jwc09n075.adm09.juwels.fzj.de
▸ 🟧 85.15 node jwc09n078.adm09.juwels.fzj.de
▾ ☐ 0.00 node jwc09n087.adm09.juwels.fzj.de
  ▾ ☐ 0.00 MPI Rank 65
      ☐ 0.00 Master thread
      ☐ 0.00 CUDA[0:7]
      🟨 22.52 CUDA[0:17]
      🟨 15.30 CUDA[0:19]
      🟨 21.09 CUDA[0:21]
      🟧 91.71 CUDA[0:23]
      🟨 8.20 CUDA[0:25]
      🟨 7.82 CUDA[0:27]
      🟨 0.81 CUDA[0:31]
  ▾ ☐ 0.00 MPI Rank 66
      ☐ 0.00 Master thread
      ☐ 0.00 CUDA[1:7]
      🟨 14.77 CUDA[1:17]
      🟨 12.56 CUDA[1:19]
      🟨 18.90 CUDA[1:21]
      🟧 82.12 CUDA[1:23]
      🟨 10.84 CUDA[1:25]
      🟨 9.58 CUDA[1:27]
      🟨 0.43 CUDA[1:31]
  ▾ ☐ 0.00 MPI Rank 67
      ☐ 0.00 Master thread
      ☐ 0.00 CUDA[2:7]
      🟨 22.88 CUDA[2:17]
      🟥 100.00 CUDA[2:23]
      🟨 0.66 CUDA[2:31]
  ▾ ☐ 0.00 MPI Rank 68
      ☐ 0.00 Master thread
      ☐ 0.00 CUDA[3:7]
      🟨 21.97 CUDA[3:17]
      🟥 99.02 CUDA[3:23]
      🟨 0.66 CUDA[3:31]
▸ 🟧 83.18 node jwc09n090.adm09.juwels.fzj.de
```

- 32 nodes: 128 MPI processes/GPUs
  - disabled Simulation file writing
- Considerable variations by rank/GPU
  - mid-range ranks 49-80 take longer, partially due to PreSend 17_k1
  - PreReceive 23_k1 dominates, but PreSend 17_k1 also significant
  - rank 65 most heavily overloaded, along with ranks 66 & 60
    - apparently due to processing for iolets at partition edges (kernels k2 & k3)

HemeLB_GPU Simulation GPU time balance (v1.20a w/o file writing)



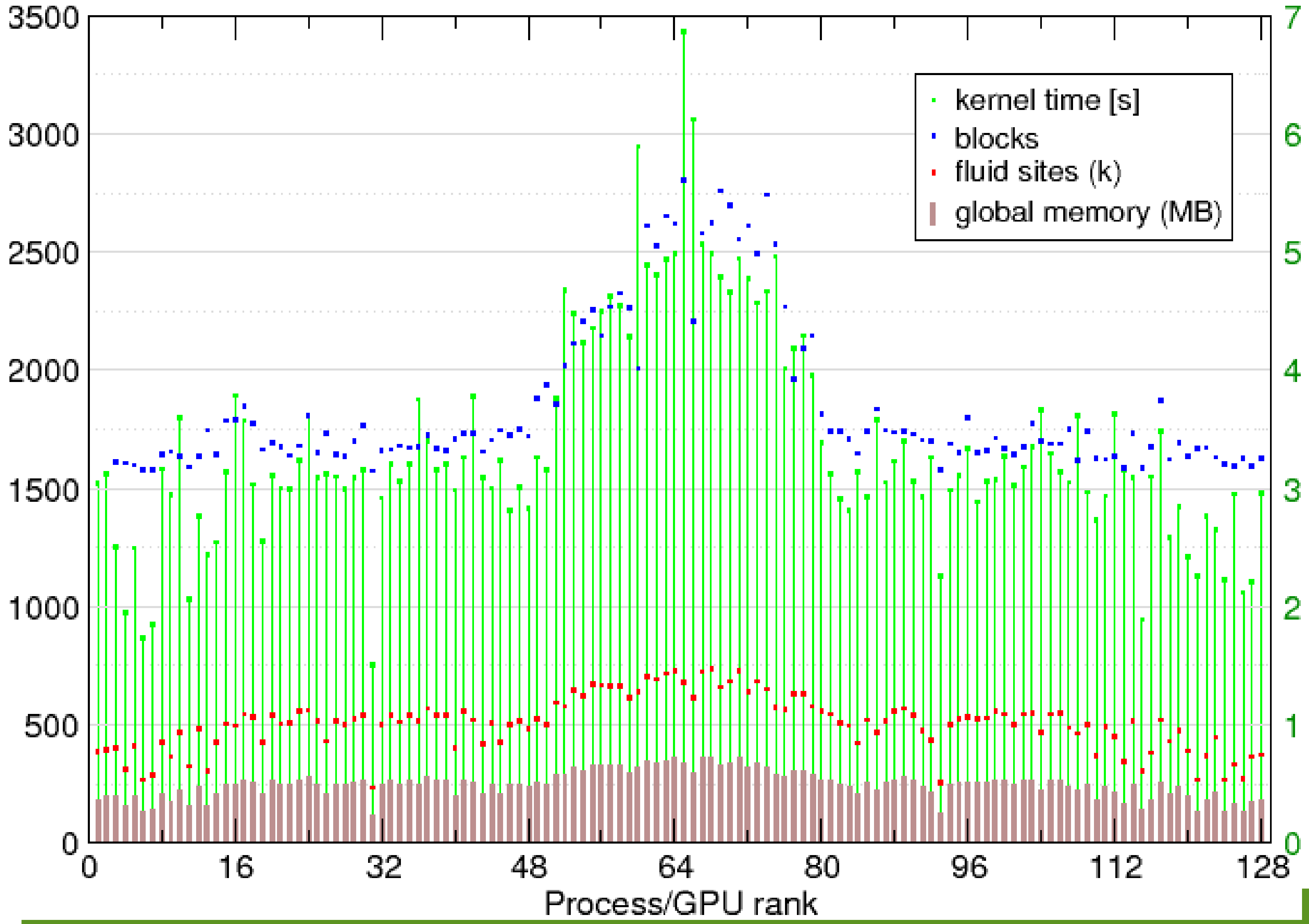
32 nodes: 128 MPI processes/GPUs

- disabled Simulation file writing

Considerable variations by rank/GPU

- mid-range ranks 52-76 take longer, partially due to PreSend 17_k1
- PreReceive 23_k1 dominates, but PreSend 17_k1 also significant
- rank 65 most heavily overloaded, along with ranks 66 & 60
  - apparently due to processing for iolets at partition edges (kernels k2 & k3)
- most GPUs idle half of the time!

HemeLB_GPU Simulation GPU time balance (v1.20a w/o file writing)



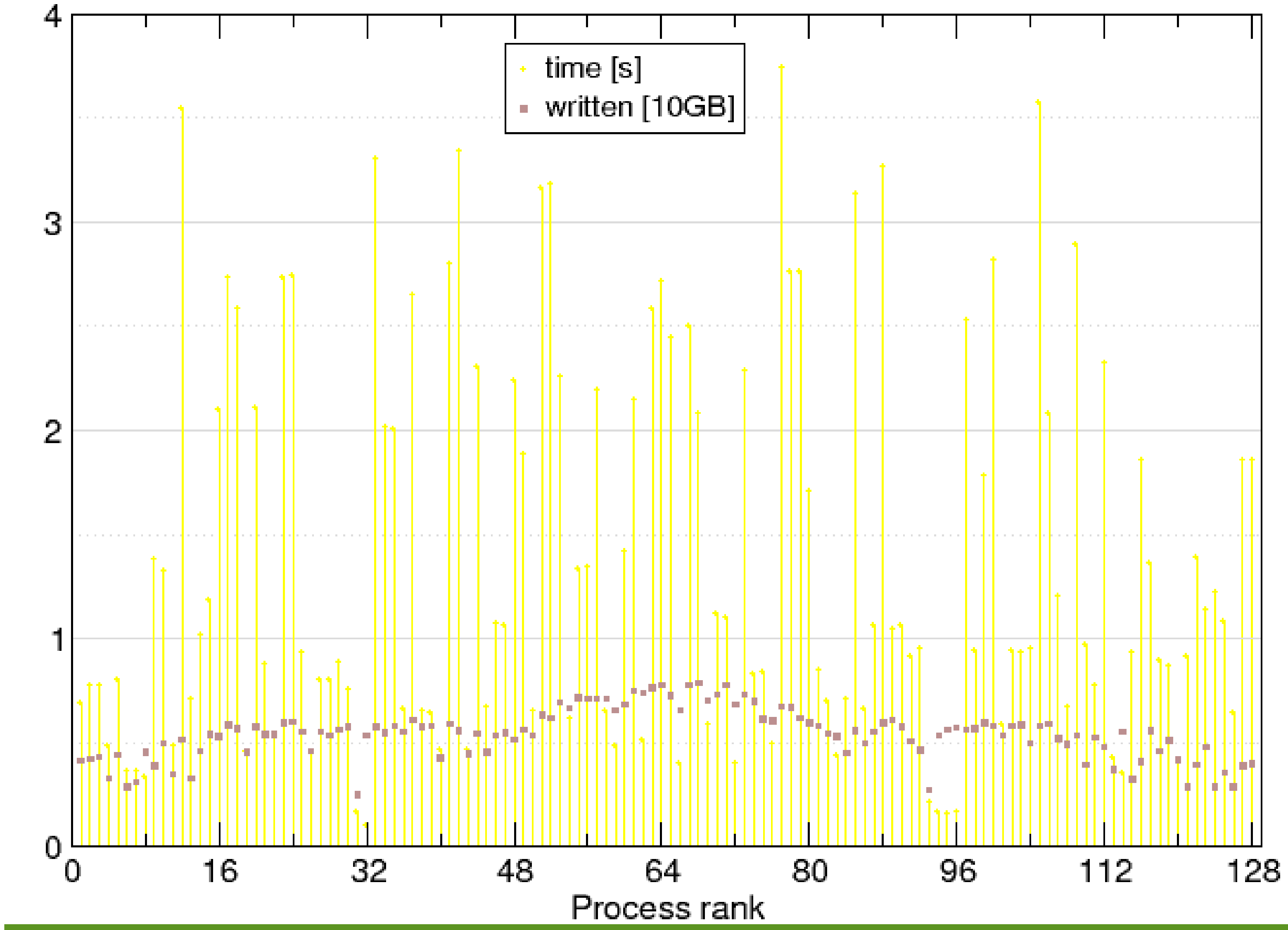
32 nodes: 128 MPI processes/GPUs

- disabled Simulation file writing

Considerable variations by rank

- GPU global memory correlates with number of fluid sites
- CUDA kernel processing time correlates with number of blocks?
- what about iolets?

HemeLB_GPU Simulation CPU file writing balance (v1.20a)



32 nodes: 128 MPI processes/GPUs

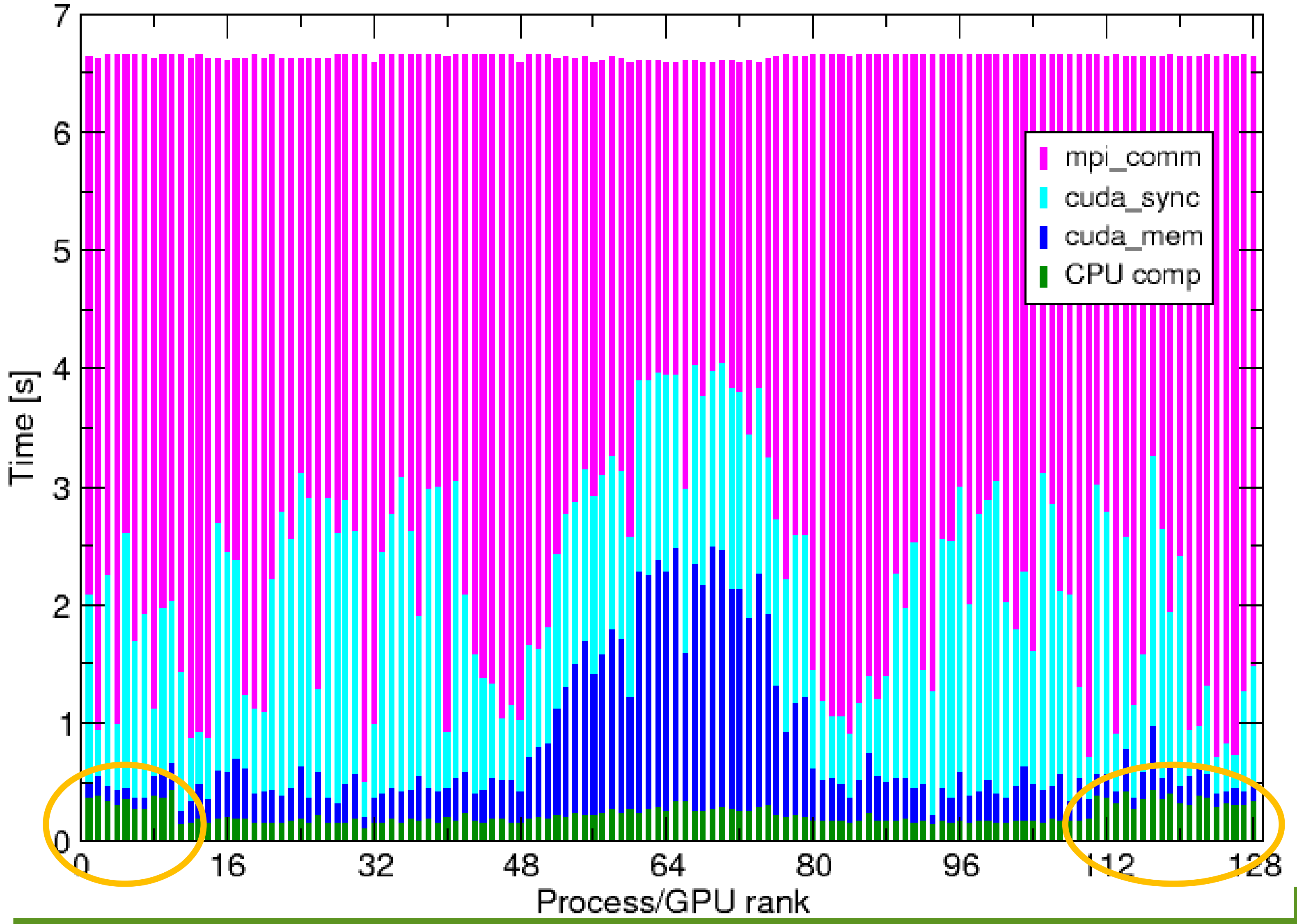
- 4 writes per rank: MPI_File_write_at
- 6.93 GiB written in total (to 2 files)
- 174s total writing time (0.01-3.75)
  - 4 procs: 3.55s (0.89-1.05)
  - 8 procs: 3.50s (0.44-0.71)
  - 16 procs: 3.76s (0.24-0.47)

Considerable variations by rank

- erratic writing time
  - varies from run to run
- time for writing not particularly correlated with amount written

Also results in additional MPI waiting before starting next simulation time step

HemeLB_GPU Simulation CPU time balance (v1.20a w/o file writing)



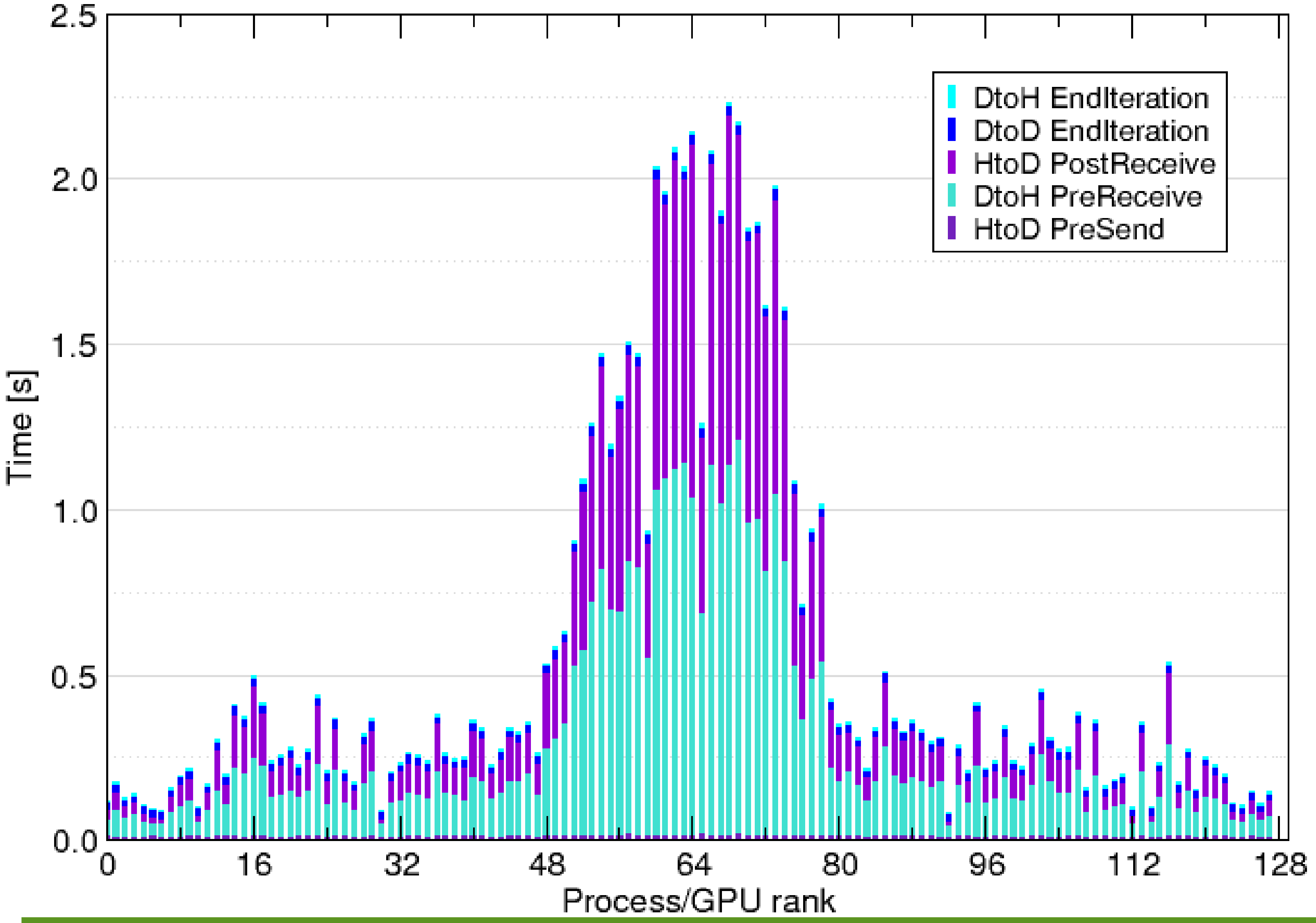
32 nodes: 128 MPI processes/GPUs

- disabled Simulation file writing

Considerable variations by rank

- computation time roughly constant, but noticably higher for highest and lowest ranks
- CUDA memory management much higher for processes in the middle
- CUDA synchronization time varies significantly
- MPI communication time dominates
  - however, almost all waiting time while GPUs compute kernels

HemeLB_GPU Simulation CPU cudaMemcpyAsync



32 nodes: 128 MPI processes/GPUs

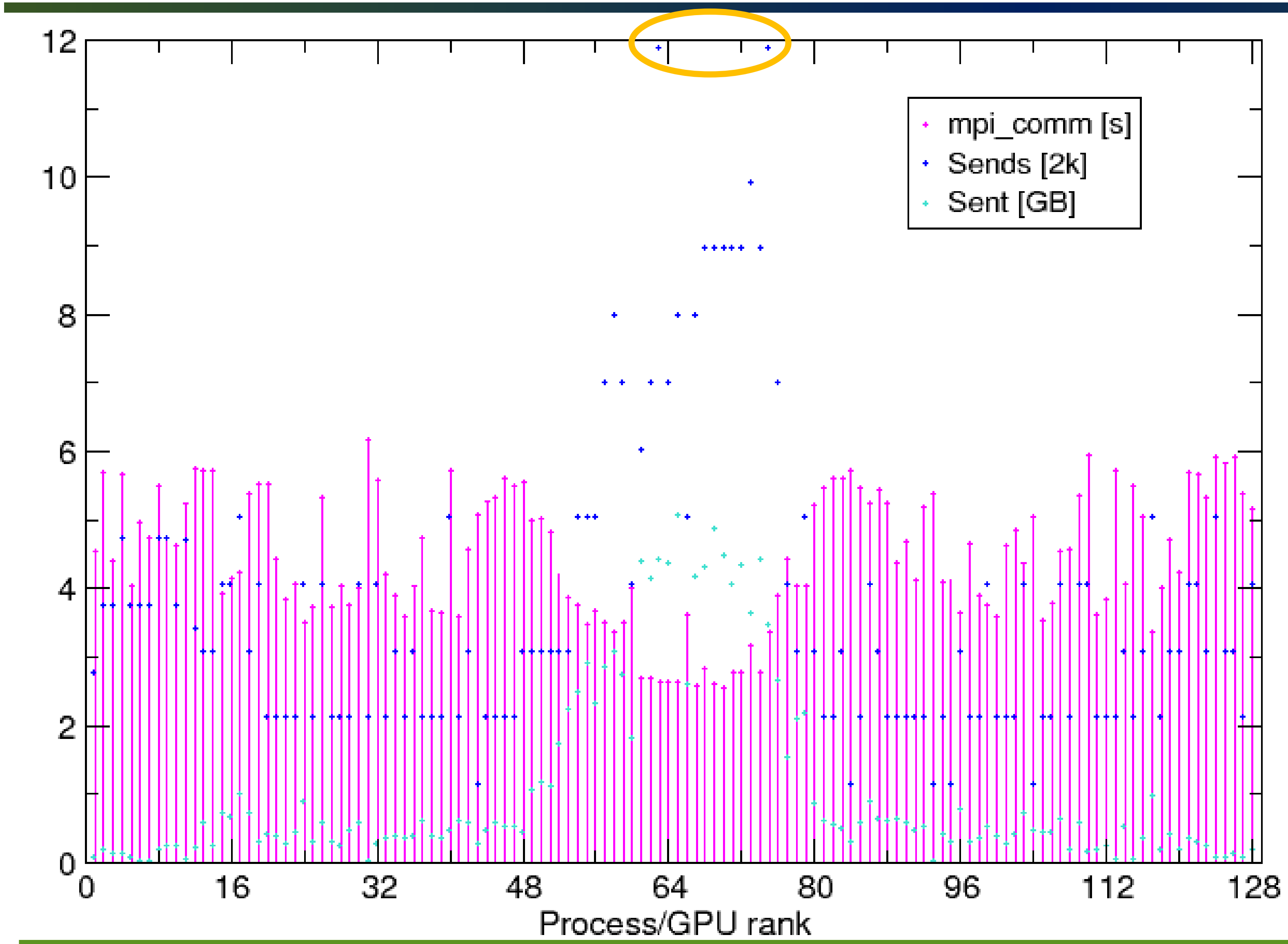
Considerable variations by rank

- CUDA memory management time much higher for processes in the middle

Some copies require much longer than others to initiate

- 51% PreReceive DtoH (Read_DistrFunctions_GPU_to_CPU_totalSharedFs)
- 41% PostReceive HtoD (Read_DistrFunctions_CPU_to_GPU_totalSharedFs)
- 4% EndIteration DtoD

HemeLB_GPU Simulation MPI communication imbalance

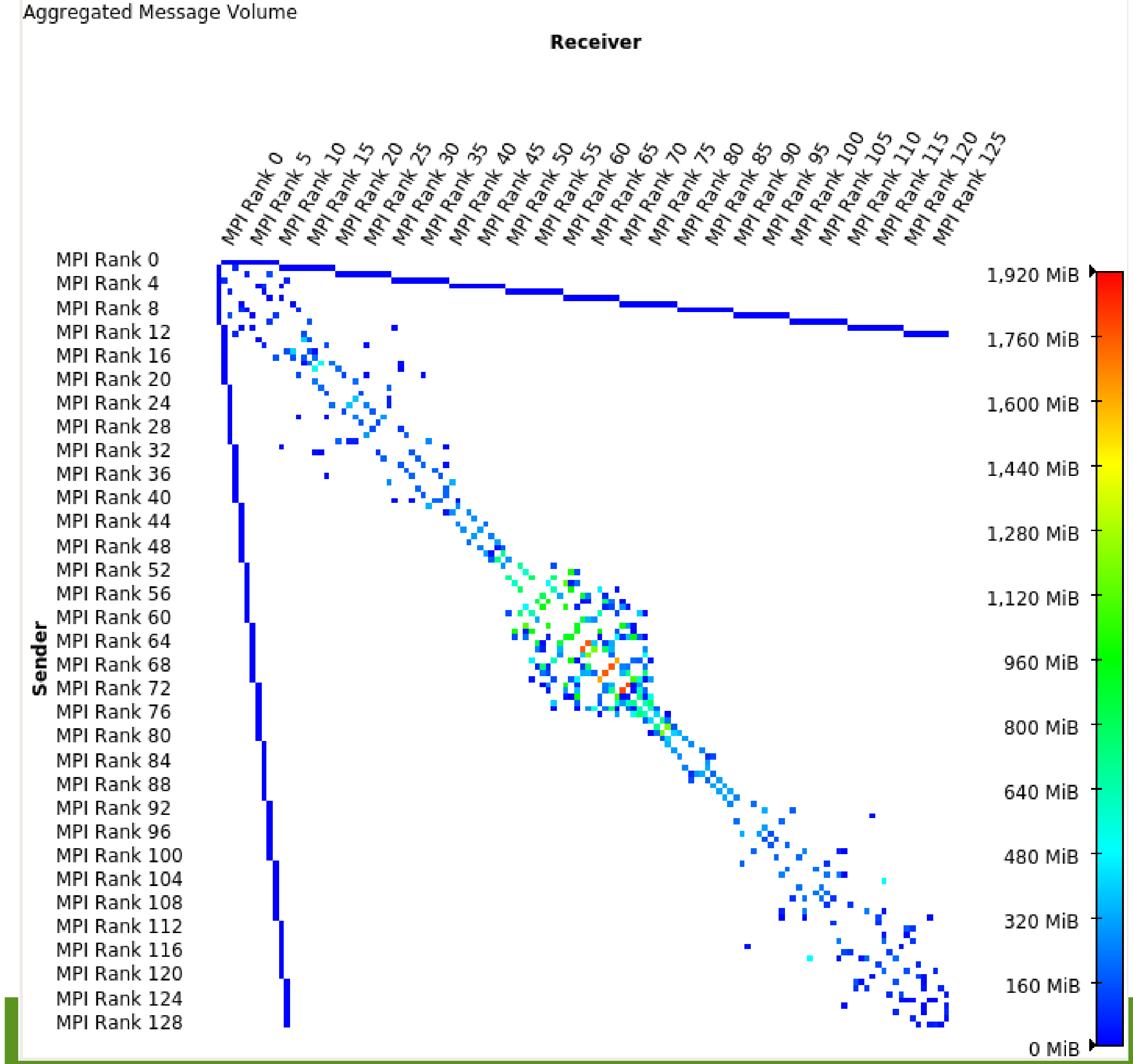


32 nodes: 128 MPI processes/GPUs

Considerable variations by rank

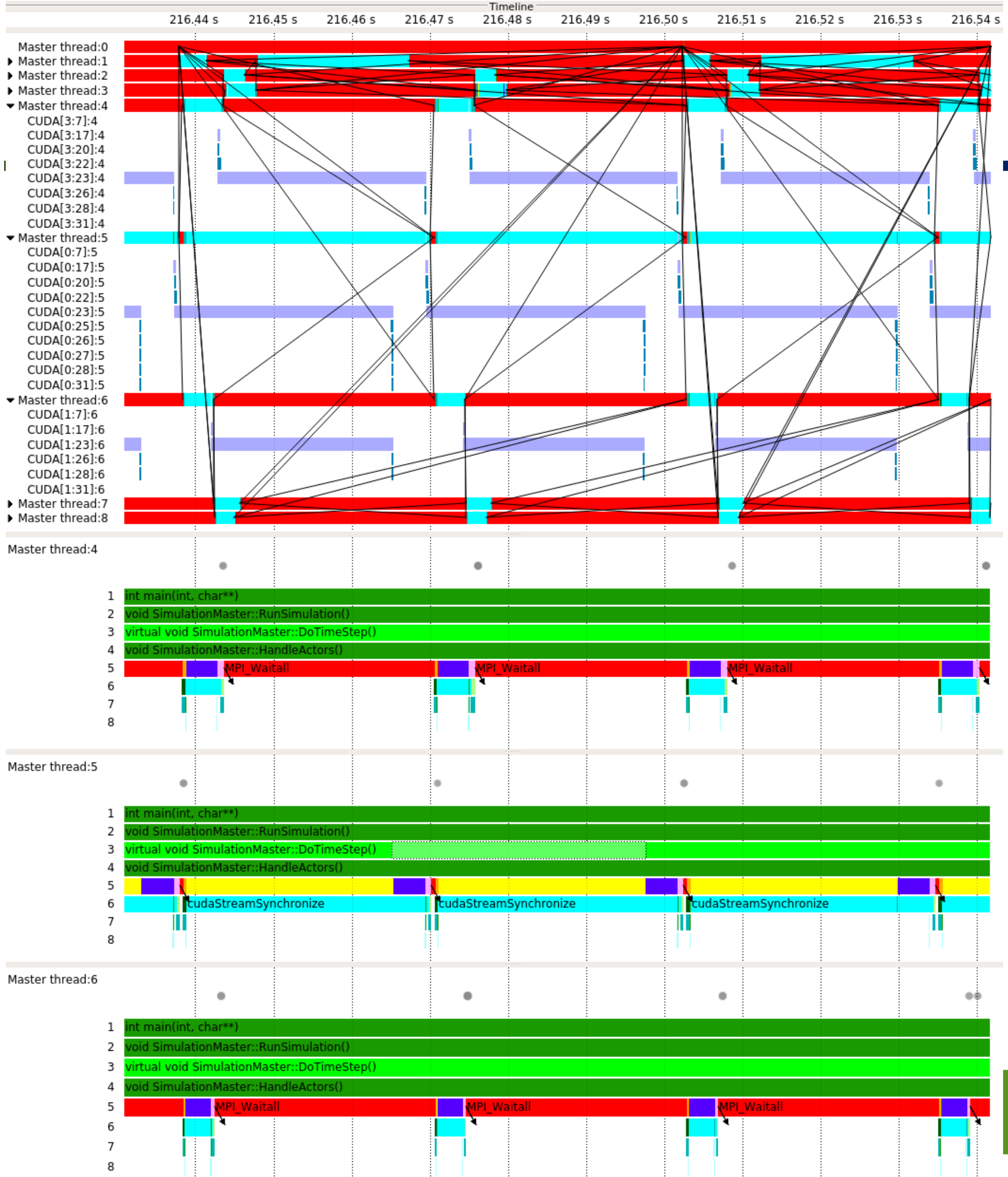
- ranks 63 & 75 have 12 partners, whereas several have only one
- most mid-range ranks 57-76 have at least 6 partners, and also exchange twice as much data as the others
- however, these mid-range ranks spend less time (waiting) in MPI communication

HemeLB_GPU Simulation MPI communication matrix

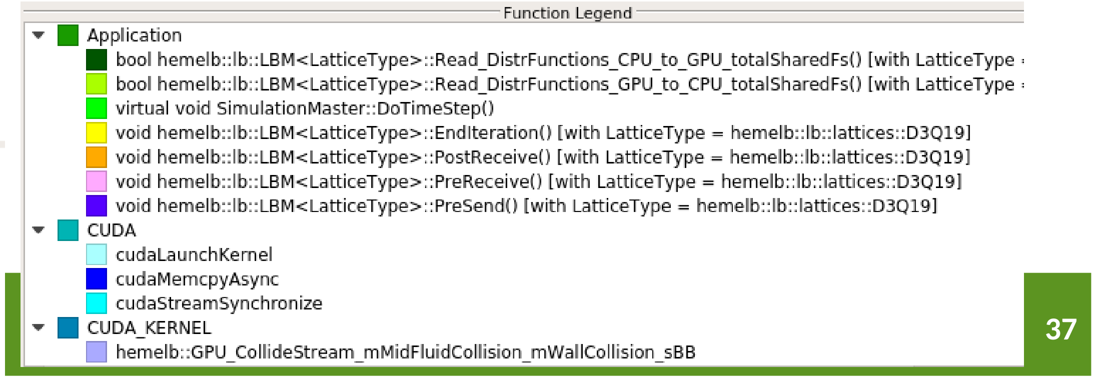


- 32 nodes: 128 MPI processes/GPUs
  - 4000 MPI_Waitall calls by each rank
  - 2334-24334 MPI_Isend/Irecv calls (messages)
    - max for ranks 63 & 75
  - 0.03 - 5.19 GB sent/received
    - max for rank 65
- Considerable variations by rank
  - heaviest communication for interior processes

Timeline detail of 3 time-steps



- 2 nodes: 9 MPI processes driving 8 GPUs
  - monitor rank 0 not executing simulation
- One MPI process (#5) mostly doing CUDA synchronization in EndIteration & next PreSend actions
  - others mostly in MPI_Waitall
  - neighbours waiting for return messages from rank 5 (and their neighbours waiting for them)
- Additional CUDA kernels executed by that process/GPU
  - iolet processing both for domain edge and interior (both PreSend and PreReceive actions)



Function Legend

- 🟩 Application
  - bool hemelb::lb::LBM<LatticeType>::Read_DistrFunctions_CPU_to_GPU_totalSharedFs() [with LatticeType :
  - bool hemelb::lb::LBM<LatticeType>::Read_DistrFunctions_GPU_to_CPU_totalSharedFs() [with LatticeType :
  - virtual void SimulationMaster::DoTimeStep()
  - void hemelb::lb::LBM<LatticeType>::EndIteration() [with LatticeType = hemelb::lb::lattices::D3Q19]
  - void hemelb::lb::LBM<LatticeType>::PostReceive() [with LatticeType = hemelb::lb::lattices::D3Q19]
  - void hemelb::lb::LBM<LatticeType>::PreReceive() [with LatticeType = hemelb::lb::lattices::D3Q19]
  - void hemelb::lb::LBM<LatticeType>::PreSend() [with LatticeType = hemelb::lb::lattices::D3Q19]
- CUDA
  - cudaLaunchKernel
  - cudaMemcpyAsync
  - cudaStreamSynchronize
- CUDA_KERNEL
  - hemelb::GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB

Notes

- Time for CUDA kernels from each GPU aggregated
  - ignores concurrent execution on GPU!
  - more v1.20a concurrency results in more kernel execution time dilation?
- GPUs assumed to idle when no kernels executing
  - don’t see asynchronous memory transfers on GPU!
- *IncompressibilityChecker* only executed by (diminishing) subset of processes on CPUs?
  - and not every 200 steps?
- MPI File I/O writing time & bytes by CPUs is imbalanced (particularly at growing scale)
  - associated with additional CPU computation and MPI communication time
  - but otherwise no apparent impact on CUDA kernels (since GPUs idle)

Comparison

- v1.20a Simulation is faster, particularly with more nodes
  - but v1.20b CUDA kernels execute faster and scale better
  - however, in both cases they execute less than half of the total Simulation time, and suffer from progressively worsening load balance (0.5 efficiency with 32 nodes)
- ***GPU_CollideStream_mMidFluidCollision_mWallCollision_sBB*** kernel constitutes most GPU (non-idle) time: 99% on 1 node decreasing progressively to 92% on 32 nodes (load balance efficiency of 0.7)
  - further load imbalance originates from (less time consuming) iolet kernels
- Simulation is dominated by MPI communication (waiting) time, with MPI file writing becoming most significant for 16 or more nodes
  - v1.20a significantly reduces MPI communication time (particularly without file writing), despite slightly increasing CUDA overheads

Summary (GPU)

- Good GPU computation scaling, however, …
- GPU kernel load imbalance is significant
  - use alternative decomposition scheme? (such as Zoltan+ParMETIS or ALL)
    - likely to require more host memory and longer initialisation time
  - revise weighting used by BasicDecomposition?
    - currently default is unweighted, but weighted GMY+ should be used instead
- cudaMemcpyAsync becomes expensive when not overlapped with kernels
  - particularly Read_DistrFunctions_CPU_to_GPU_totalSharedFs cuMemcpyDtoHAsync
  - CUDA-aware MPI should avoid need for these copies between CPU and GPU, and generally be much more efficient for data transfers between GPUs

Summary (CPU)

- Major inefficiencies arise from file I/O and non-GPU computation
- GPUs are idle much of the time with CPU-only activities during Simulation *DoTimeStep*
  - periodic property file writing with MPI File I/O (by all ranks)
    - possible to do asynchronously (with dedicated processes/threads)
    - or non-blocking MPI_File_iwrite_at (+ MPI_Wait)
  - periodic incompressibility check (only by some ranks): may be better done on GPUs?
- Additional ‘monitor’ process is extra complication
  - necessary for future coupled codes
- Initialisation of simulation takes a significant time and warrants detailed investigation
  - time grows with size of geometry, but not specifically file reading
    - investigate additional reader ranks (possibly all simulation ranks)?

Performance Optimisation and Productivity

A Centre of Excellence in Computing Applications & HPC

Contact:

https://www.pop-coe.eu

mailto:pop@bsc.es

This project has received funding from the European Union Horizon 2020 research and innovation programme under grant agreements No 676553 & 824080.

HemeLB_GPU Simulation strong scaling efficiency (v1.20a w/o file writing)

	1n 5p	2n 9p	4n 17p	8n 33p	16n 65p	32n 129p
Simulation time [s]	137.73	82.33	44.98	21.00	10.87	6.66
Global scaling efficiency	0.68	0.57	0.52	0.56	0.54	0.44
– Parallel efficiency	0.68	0.57	0.53	0.59	0.59	0.51
– – Load balance efficiency (GPU)	0.95	0.78	0.73	0.73	0.65	0.49
– – Communication efficiency (GPU)	0.72	0.73	0.73	0.80	0.92	1.03
– Computation scaling (GPU)	*1.00*	0.99	0.98	0.95	0.91	0.87

Key:
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

- Only considering GPUs (ignoring all CPU cores, 90% of which are completely unused)
- Parallel efficiency determined by load balance
- Single (quad-GPU) node already suffers significant communication inefficiency
  - but improves as additional nodes are included?
- Load balance of GPUs deteriorates progressively
- GPU computation scaling remains reasonably good

HemeLB_GPU Simulation strong scaling efficiency (v1.20a) hybrid CPU+GPU

	1n 5p	2n 9p	4n 17p	8n 33p	16n 65p	32n 129p	
Simulation time [s]	147.87	88.38	48.13	22.66	13.68	11.67	
Global scaling efficiency	0.35	0.33	0.32	0.35	0.29	0.30	CPU+GPU
– Parallel efficiency	0.35	0.33	0.30	0.31	0.27	0.28	
– – Load balance efficiency	0.77	0.69	0.65	0.75	0.66	0.52	
– – Communication efficiency	0.46	0.47	0.46	0.41	0.40	0.54	
– Computation scaling	1.00	1.00	1.06	1.12	1.11	1.07	
Global scaling efficiency (CPU)	0.19	0.18	0.17	0.19	0.16	0.09	CPU only
– Parallel efficiency (CPU)	0.19	0.18	0.13	0.10	0.07	0.04	
– – Load balance efficiency (CPU)	0.78	0.68	0.52	0.45	0.38	0.64	
– – Communication efficiency (CPU)	0.25	0.26	0.26	0.21	0.17	0.06	
– Computation scaling (CPU)	1.00	1.00	1.32	2.01	2.47	2.63	
File writing scaling	1.00	1.02	0.94	0.35	0.05	0.02	
– Bytes written scaling	1.00	1.00	1.00	1.00	1.00	1.00	
Global scaling efficiency (GPU)	0.64	0.53	0.49	0.52	0.43	0.25	GPU only
– Parallel efficiency (GPU)	0.64	0.53	0.50	0.54	0.47	0.29	
– – Load balance efficiency (GPU)	0.95	0.78	0.73	0.73	0.65	0.50	
– – Communication efficiency (GPU)	0.67	0.68	0.68	0.75	0.73	0.58	
– Computation scaling (GPU)	*1.00*	1.00	0.99	0.96	0.92	0.87	

Key:

Key:
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0