Omnitrace By Example

Bob Robey, Gina Sitaraman, Ian Bogle, Giacomo Capodaglio, Asitav Mishra

AMD @ ORNL
29 May 2024

AMD
together we advance_

Acknowledgements

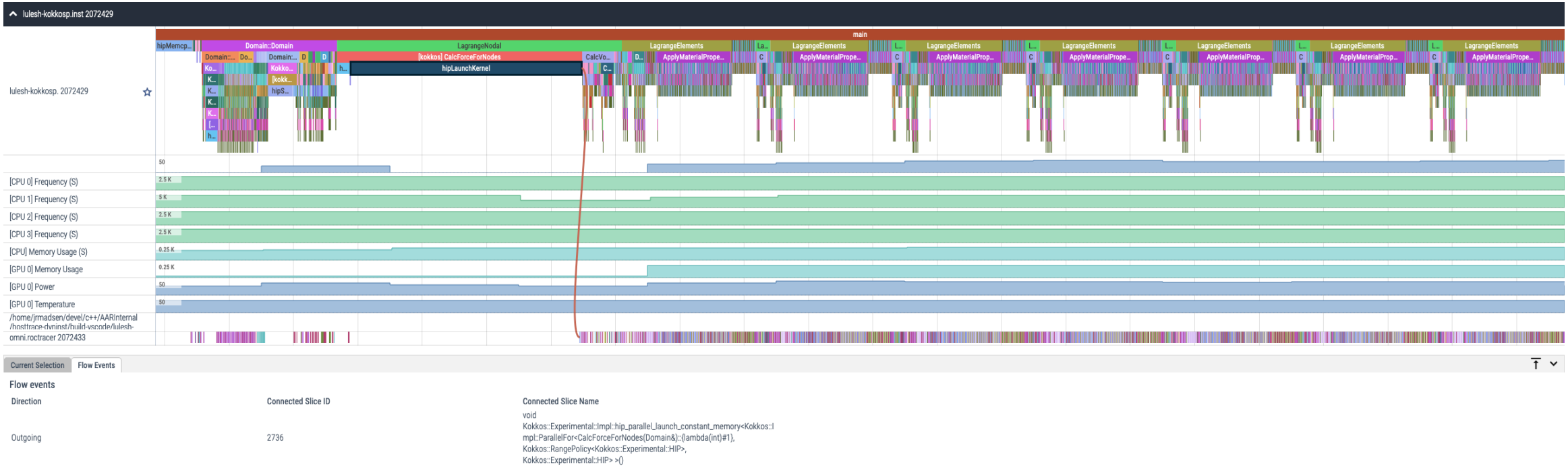
- Jonathan Madsen
- David Galiffi
- Nicholas Curtis
- and the rest of the DCGPU team

Agenda

- Omnitrace for Application Profiling and Tracing
- A Simple Ghost Exchange MPI Example Suite
- Orig: CPU implementation
- Ver1: OpenMP® offload port with Managed Memory
- Ver2: Add roctx ranges
- Ver3: Allocate MPI buffers on device (Under construction)
- Ver4: Allocate all buffers once
- Ver5: Convert from 2D to 1D indexing
- Ver6: Add explicit data map directives

Omnitrace for Application Profiling and Tracing

- Get high level view of entire application run
- Holistic view of CPU, GPU, and system activity
- Sampling and binary instrumentation modes
- Visualize in Perfetto



MPI Ghost Exchange Example Suite

- Many applications need to exchange ghost cells with adjacent processes
- This example suite, developed by Bob Robey, implements the exchange for a regular cartesian grid
- We start with the examples in Chapter 8 of Parallel and High Performance Computing, Manning Publications

```
git clone https://EssentialsOfParallelComputing/Chapter8
```

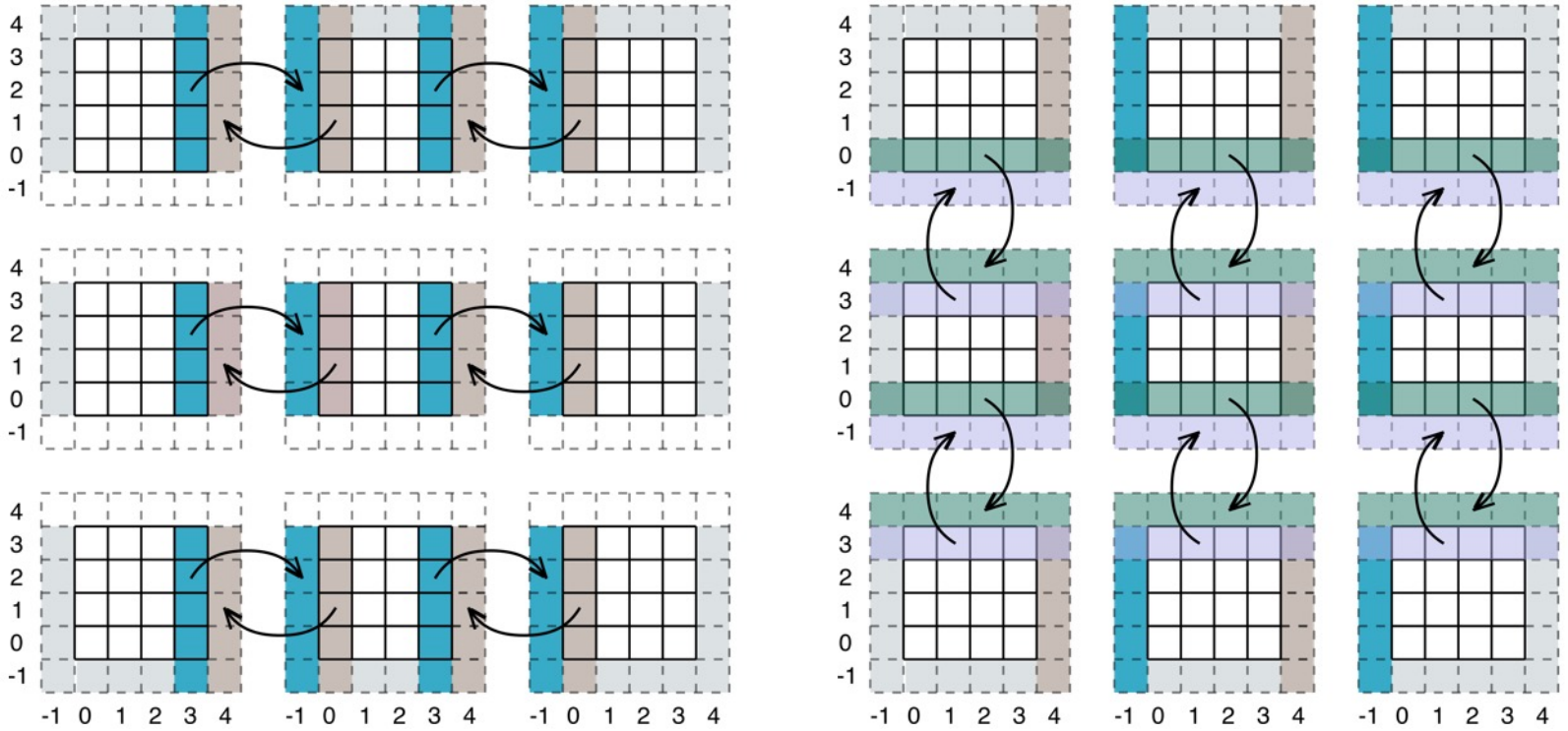
https://github.com/essentialsofparallelcomputing/Chapter8

```
cd GhostExchange
```

  - These examples include various versions ranging from simple methods to those using MPI Datatypes and MPI Cartesian topology capabilities
  - For GPU-Aware MPI, the versions using MPI Datatypes have not been optimized in most MPI implementations
  - We will use the simpler methods. We will pack the column data into to a buffer and send the buffer. We can send row data directly. If corner data is needed, synchronization is required
- From CPU code, we port to GPU and optimize incrementally using Omnitrace to guide the process
- Uses OpenMP target offload mechanism for offloading compute to AMD GPUs
- Repo: https://github.com/amd/HPCTrainingExamples/tree/main/MPI-examples/GhostExchange/GhostExchange_ArrayAssign

MPI Ghost Exchange Example – How does it work?

- A rectangular domain is partitioned into a 2D computational grid, distributed among MPI processes
- An initial solution in specified on a cell-wise basis, then advanced with a 5-point stencil averaging operator
- Halo cells are located along the boundary, and around MPI domains (*ghost cells*) when doing parallel runs
- Boundary conditions are of *outflow* type, enforced prior to ghost halo exchanges
- Example of 2-step halo exchange with 3x3 grid of processes, each owning a 4x4 subset of the mesh:



MPI Ghost Exchange Examples – How to run it?

- Parameters:

```
-x nprocx  -y nprocy  -i imax -j jmax -h nhalo -t (0 or 1) -c (0 or 1) -I maxIter
```

`nprocx` = number of processes in x dimension
`nprocy` = number of processes in y dimension
`imax` = number of mesh cells in x dimension
`jmax` = number of mesh cells in y dimension
`nhalo` = number of halo layers
`maxIter` = maximum number of iterations
`-t` = enable/disable sync before MPI calls to accurately time MPI overhead
`-c` = include/exclude corner cell updates

- Example run on Frontier with 4 ranks:

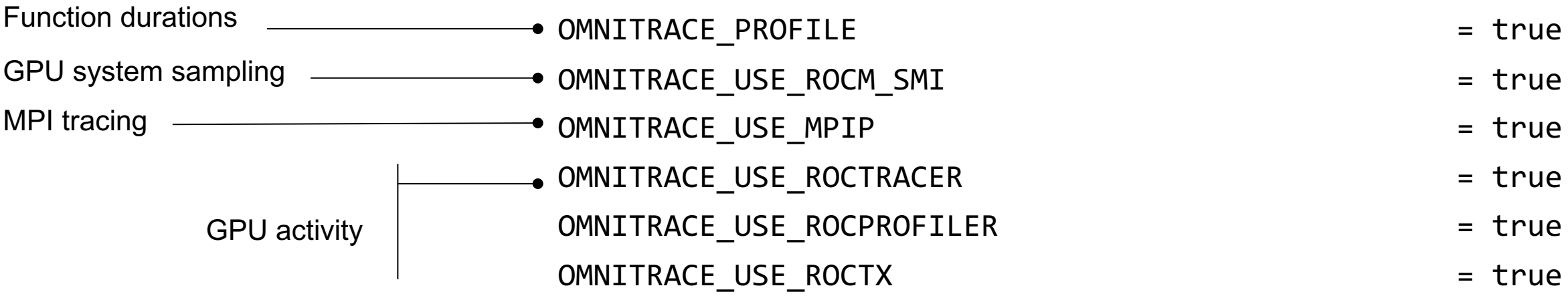
```
srun –N1 –n4 –c7 ./GhostExchange -x 2  -y 2  -i 20000 -j 20000 -h 2 -t -c -I 100
```

Getting Started with Omnitrace - Configuring Omnitrace Runtime

- First, create a default configuration file

```
omnitrace-avail -G ~/.omnitrace.cfg
export OMNITRACE_CONFIG_FILE=~/.omnitrace.cfg
```

- Contains settings to control Omnitrace runtime behavior, modify settings as desired



Refer to documentation for more `omnitrace-avail` capabilities: https://rocm.github.io/omnitrace/runtime.html

Running Omnitrace on Ghost Exchange Examples

- Set up your environment on Frontier

```
module load cce/17.0.0
module load rocm/5.7.0
module load omnitrace/1.11.2
module load craype-accel-amd-gfx90a
module load cmake/3.23.2
```

- Build the code

```
mkdir build; cd build; cmake ..; make -j8
```

- Instrument the binary

```
omnitrace-instrument -o ./GhostExchange.inst -- ./GhostExchange
```

- Profile the instrumented binary

```
srun -N1 -n4 -c7 --gpu-bind=closest -A <proj> -t 05:00 omnitrace-run --
./GhostExchange.inst -x 2  -y 2  -i 20000 -j 20000 -h 2 -t -c -I 100
```

Understanding output from `omnitrace-instrument`

```
[omnitrace][exe] Outputting 'omnitrace-GhostExchange.inst-output/2024-05-22_16.24/instrumentation/available.json'... Done
[omnitrace][exe] Outputting 'omnitrace-GhostExchange.inst-output/2024-05-22_16.24/instrumentation/available.txt'... Done
[omnitrace][exe] Outputting 'omnitrace-GhostExchange.inst-output/2024-05-22_16.24/instrumentation/instrumented.json'... Done
[omnitrace][exe] Outputting 'omnitrace-GhostExchange.inst-output/2024-05-22_16.24/instrumentation/instrumented.txt'... Done
[omnitrace][exe] Outputting 'omnitrace-GhostExchange.inst-output/2024-05-22_16.24/instrumentation/excluded.json'... Done
[omnitrace][exe] Outputting 'omnitrace-GhostExchange.inst-output/2024-05-22_16.24/instrumentation/excluded.txt'... Done
[omnitrace][exe] Outputting 'omnitrace-GhostExchange.inst-output/2024-05-22_16.24/instrumentation/overlapping.json'... Done
[omnitrace][exe] Outputting 'omnitrace-GhostExchange.inst-output/2024-05-22_16.24/instrumentation/overlapping.txt'... Done
```

Functions that could be instrumented

```
[ssitaram@login05.frontier build]$ cat omnitrace-GhostExchange.inst-output/2024-05-22_16.24/instrumentation/available.txt
```

StartAddress	AddressRange	#Instructions	Ratio	Linkage	Visibility	Module	Function	FunctionSignature
0x21162c	9	3	3.00	local	hidden	../sysdeps/x86_64/crti.S	_fini	_fini
0x211614	23	7	3.29	local	hidden	../sysdeps/x86_64/crti.S	_init	_init
0x20250e	45	13	3.46	global	default	../sysdeps/x86_64/start.S	_start	_start
0x210af0	1963	485	4.05	unknown	unknown	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	Cartesian_print	Cartesian_print
0x20c4d0	7069	1509	4.68	unknown	unknown	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	boundarycondition_update	boundarycondition_update
0x20e070	10868	2403	4.52	unknown	unknown	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	ghostcell_update	ghostcell_update
0x209d00	10184	2198	4.63	unknown	unknown	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	haloupdate_test	haloupdate_test
0x2025e0	29900	6532	4.58	global	default	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	main	main
0x209ab0	591	157	3.76	unknown	unknown	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	parse_input_args	parse_input_args
0x211300	303	88	3.44	global	default	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	malloc2D	malloc2D
0x211430	15	4	3.75	global	default	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	malloc2D_free	malloc2D_free
0x2112a0	13	3	4.33	global	default	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	cpu_timer_start	cpu_timer_start
0x2112b0	71	19	3.74	global	default	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	cpu_timer_stop	cpu_timer_stop
0x20257f	85	23	3.70	local	default	GhostExchange	__do_fini	__do_fini
0x20253b	68	19	3.58	local	default	GhostExchange	__do_init	__do_init
0x21143f	320	88	3.64	global	default	GhostExchange	__no_mmap_for_malloc	__no_mmap_for_malloc
0x21160e	4	3	1.33	global	default	elf-init.c	__libc_csu_fini	__libc_csu_fini
0x21158e	103	36	2.86	global	default	elf-init.c	__libc_csu_init	__libc_csu_init

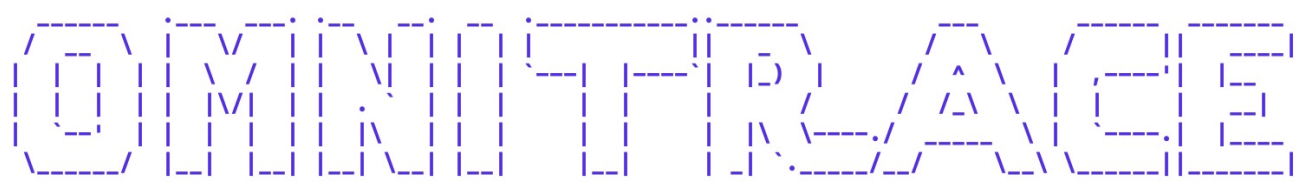
```
[ssitaram@login05.frontier build]$ cat omnitrace-GhostExchange.inst-output/2024-05-22_16.24/instrumentation/instrumented.txt
```

StartAddress	AddressRange	#Instructions	Ratio	Linkage	Visibility	Module	Function	FunctionSignature
0x20c4d0	7069	1509	4.68	unknown	unknown	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	boundarycondition_update	boundarycondition_update
0x20e070	10868	2403	4.52	unknown	unknown	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	ghostcell_update	ghostcell_update
0x209d00	10184	2198	4.63	unknown	unknown	/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostEx...	haloupdate_test	haloupdate_test

Functions that were instrumented

Understanding output from `omnitrace-run`

Omnitrace ASCII art is proof that Omnitrace is running, shows version used



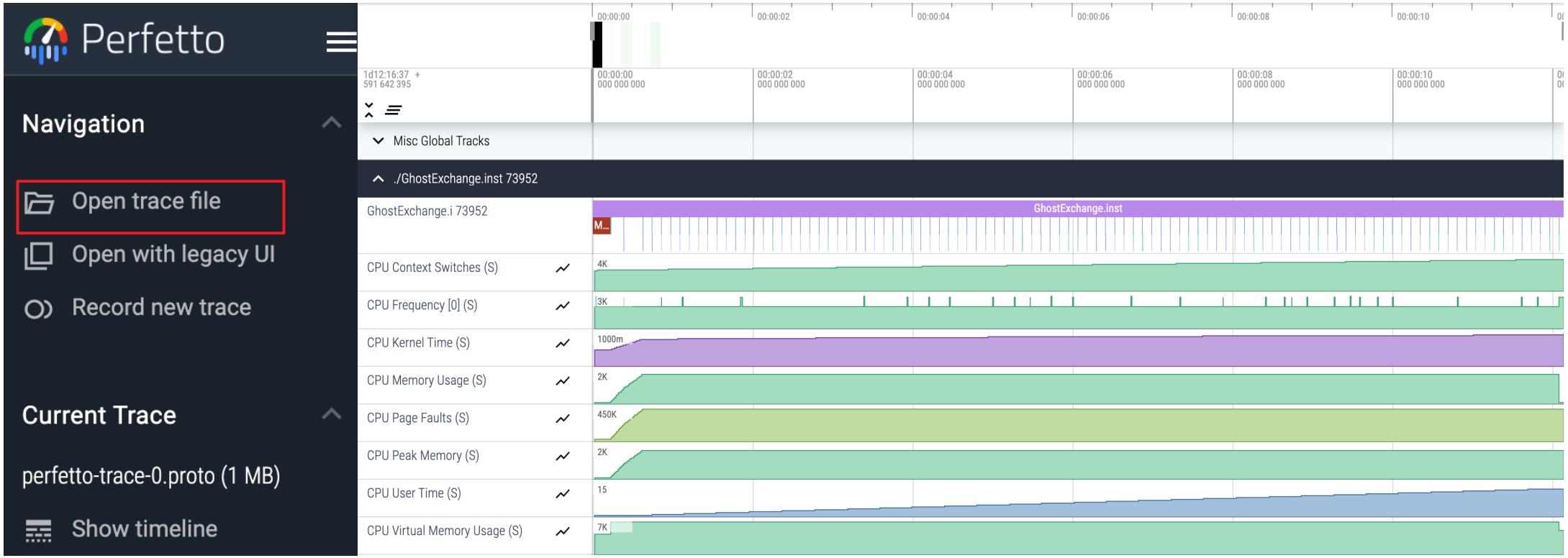
```
omnitrace v1.11.2 (rev: 1df597e049b240fb263e7fcd7bddc78097d27f00, tag: v1.11.2, compiler: GNU v7.5.0, rocm: v5.7.x)
```

```
[omnitrace][130341][perfetto]> Outputting '/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostExchange_ArrayAssign/Orig/build/omnitrace-GhostExchange.inst-output/2024-05-22_16.50/perfetto-trace-1.proto' (765.48 KB / 0.77 MB / 0.00 GB)... Done
[omnitrace][130342][perfetto]> Outputting '/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostExchange_ArrayAssign/Orig/build/omnitrace-GhostExchange.inst-output/2024-05-22_16.50/perfetto-trace-2.proto' (765.47 KB / 0.77 MB / 0.00 GB)... Done
[omnitrace][130343][perfetto]> Outputting '/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostExchange_ArrayAssign/Orig/build/omnitrace-GhostExchange.inst-output/2024-05-22_16.50/perfetto-trace-3.proto' (765.47 KB / 0.77 MB / 0.00 GB)... Done
[omnitrace][130340][perfetto]> Outputting '/ccs/home/ssitaram/git/HPCTrainingExamples/MPI-examples/GhostExchange/GhostExchange_ArrayAssign/Orig/build/omnitrace-GhostExchange.inst-output/2024-05-22_16.50/perfetto-trace-0.proto' (765.48 KB / 0.77 MB / 0.00 GB)... Done
```

Paths to output trace files

Visualizing Omnitrace .proto files

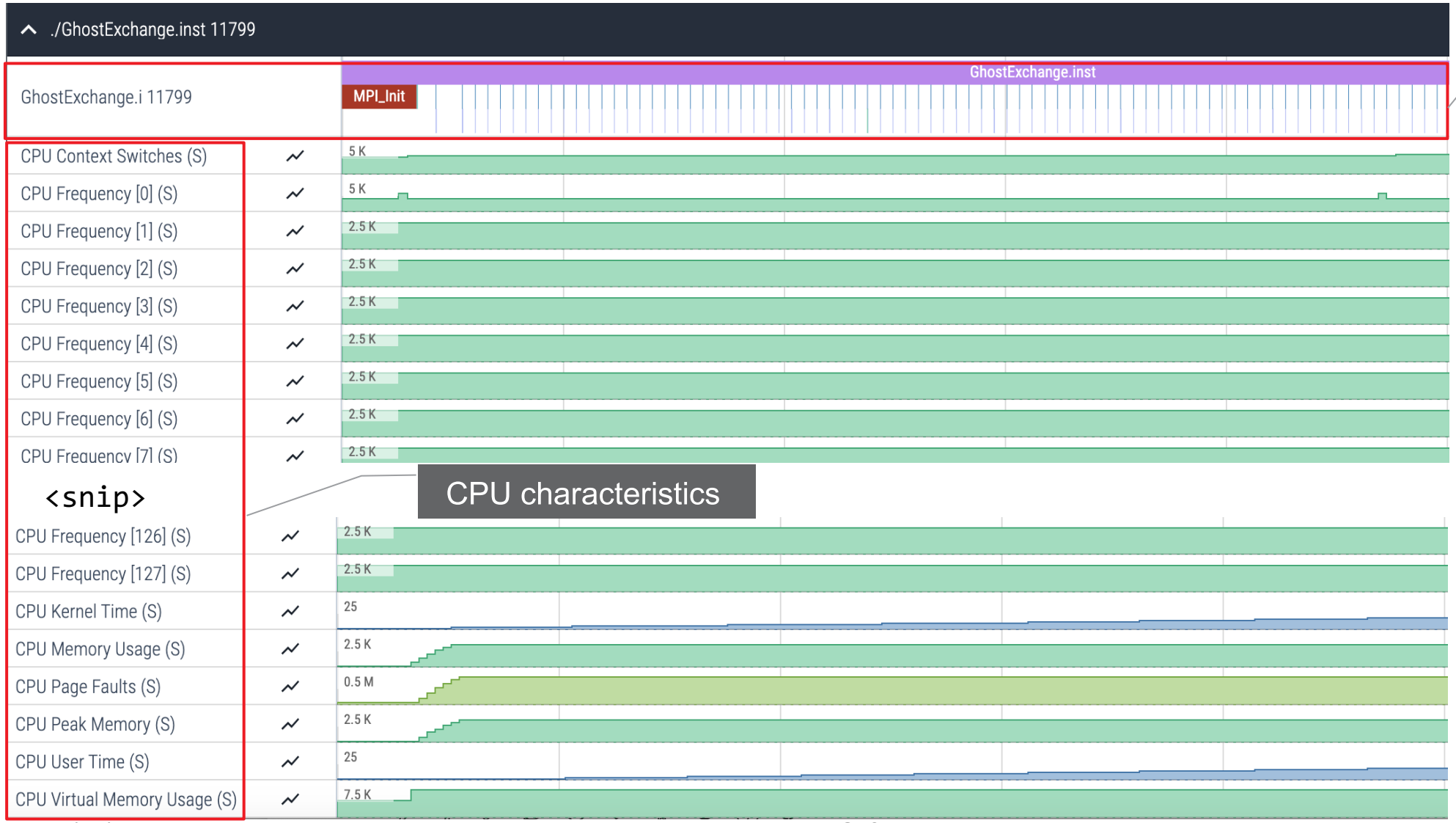
Copy .proto file to local workstation or laptop, open in Perfetto: https://ui.perfetto.dev/



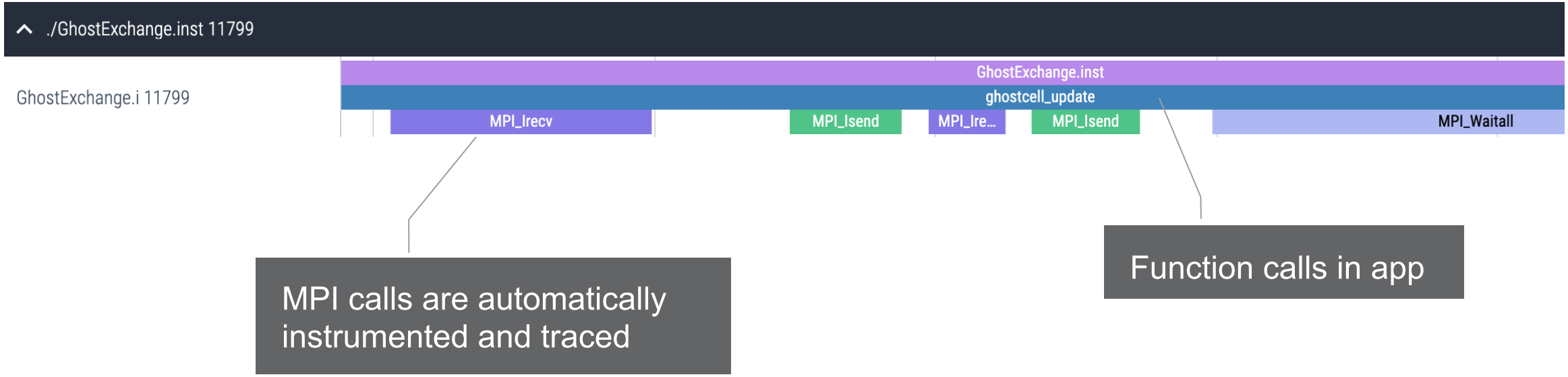
Orig

CPU implementation of Ghost Exchange

Orig: First look at Omnitrace profile for Rank 0



Orig: First profile – zoom in

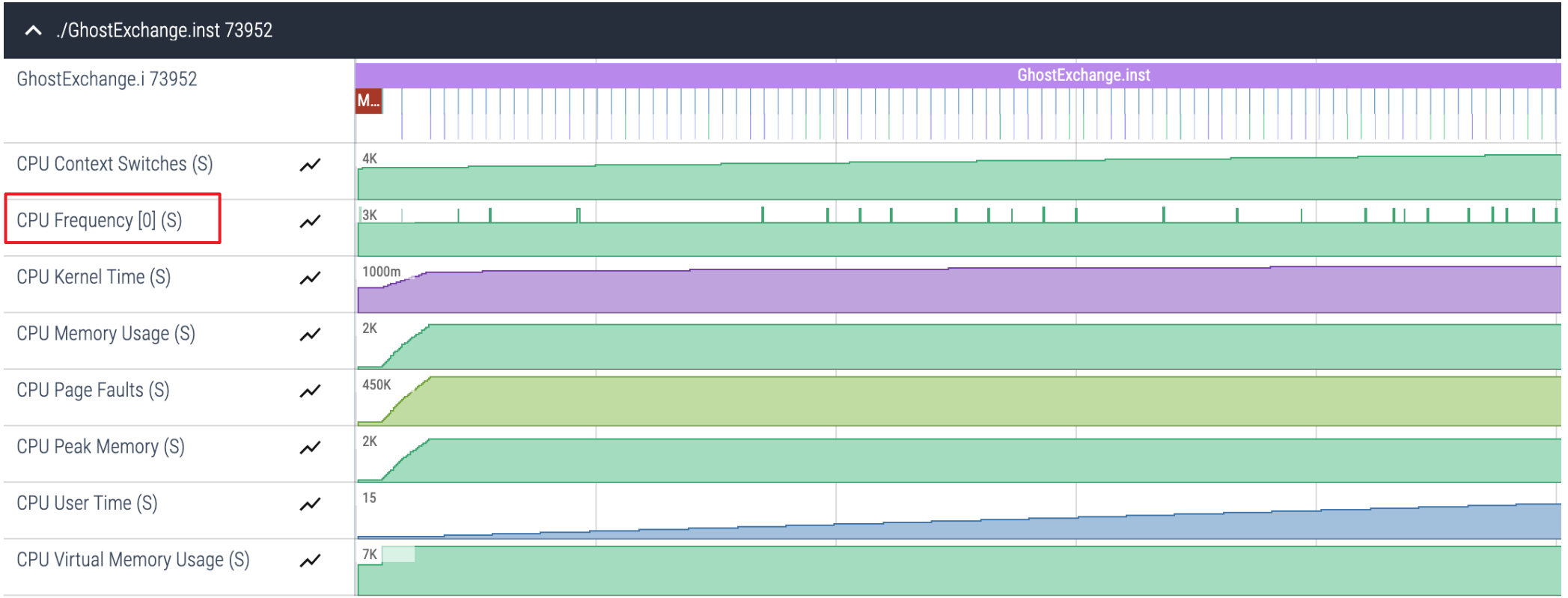


Orig: Keep profile short - sample one CPU core

- Update config file to sample only 1 CPU core:

```
OMNITRACE_SAMPLING_CPUS                    = 0
```

- Just rerun, no need to instrument again



Orig: Generate CPU-side wall clock times in profile

- Enable tracking of function durations in config file

```
OMNITRACE_PROFILE                                    = true
```

- Omnitrace generates wall_clock files with durations of each instrumented function

```
[omnitrace][57701][wall_clock]> Outputting 'omnitrace-GhostExchange.inst-output/2024-05-22_16.44/wall_clock-3.txt'
[omnitrace][57699][wall_clock]> Outputting 'omnitrace-GhostExchange.inst-output/2024-05-22_16.44/wall_clock-1.txt'
[omnitrace][57698][wall_clock]> Outputting 'omnitrace-GhostExchange.inst-output/2024-05-22_16.44/wall_clock-0.txt'
```

- Look for durations of MPI calls here

REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> GhostExchange.inst	1	0	wall_clock	sec	12.179875	12.179875	12.179875	12.179875	0.000000	0.000000	97.7
|0>>> |_MPI_Init	1	1	wall_clock	sec	0.194357	0.194357	0.194357	0.194357	0.000000	0.000000	100.0
|0>>> |_MPI_Comm_rank	4	1	wall_clock	sec	0.000278	0.000069	0.000001	0.000270	0.000000	0.000134	100.0
|0>>> |_MPI_Comm_size	2	1	wall_clock	sec	0.000011	0.000006	0.000001	0.000010	0.000000	0.000006	100.0
|0>>> |_MPI_Allreduce	1	1	wall_clock	sec	0.001482	0.001482	0.001482	0.001482	0.000000	0.000000	100.0
|0>>> |_boundarycondition_update	1	1	wall_clock	sec	0.000352	0.000352	0.000352	0.000352	0.000000	0.000000	100.0
|0>>> |_ghostcell_update	101	1	wall_clock	sec	0.081816	0.000810	0.000415	0.002370	0.000000	0.000408	43.2
|0>>> |_MPI_Irecv	404	2	wall_clock	sec	0.003434	0.000009	0.000003	0.000171	0.000000	0.000011	100.0
|0>>> |_MPI_Isend	404	2	wall_clock	sec	0.002443	0.000006	0.000003	0.000038	0.000000	0.000007	100.0
|0>>> |_MPI_Waitall	202	2	wall_clock	sec	0.040556	0.000201	0.000009	0.001980	0.000000	0.000321	100.0

Orig: Use flat profiles for finding hotspots

- To flatten the hierarchy in the wall clock profile, enable flat profile:

```
OMNITRACE_FLAT_PROFILE = true
```

- Now each function appears once, all timings are consolidated for each function:

REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> GhostExchange.inst	1	0	wall_clock	sec	12.252487	12.252487	12.252487	12.252487	0.000000	0.000000	100.0
|0>>> MPI_Init	1	0	wall_clock	sec	0.198300	0.198300	0.198300	0.198300	0.000000	0.000000	100.0
|0>>> MPI_Comm_rank	4	0	wall_clock	sec	0.000474	0.000118	0.000001	0.000466	0.000000	0.000232	100.0
|0>>> MPI_Comm_size	2	0	wall_clock	sec	0.000019	0.000009	0.000001	0.000017	0.000000	0.000011	100.0
|0>>> MPI_Allreduce	1	0	wall_clock	sec	0.002375	0.002375	0.002375	0.002375	0.000000	0.000000	100.0
|0>>> boundarycondition_update	1	0	wall_clock	sec	0.000304	0.000304	0.000304	0.000304	0.000000	0.000000	100.0
|0>>> ghostcell_update	101	0	wall_clock	sec	0.079793	0.000790	0.000439	0.002256	0.000000	0.000282	100.0
|0>>> MPI_Irecv	404	0	wall_clock	sec	0.004957	0.000012	0.000003	0.000165	0.000000	0.000013	100.0
|0>>> MPI_Isend	404	0	wall_clock	sec	0.002565	0.000006	0.000003	0.000043	0.000000	0.000008	100.0
|0>>> MPI_Waitall	202	0	wall_clock	sec	0.033038	0.000164	0.000010	0.001354	0.000000	0.000221	100.0

Not much here other than MPI calls not being the bottleneck

Ver1

First GPU implementation of Ghost Exchange
OpenMP offload + Managed Memory Programming Model

Ver1: Code changes

- Pragma for unified memory added to each translation unit

```
#pragma omp requires unified_shared_memory
```

- Target offload pragma added to all compute loops

```
#pragma omp target teams distribute parallel for collapse(2)
```

- Data still resides in host memory, but accessed from compute kernels and MPI calls
- On Frontier nodes, this means data is moved across AMD Infinity Fabric™ link between CPU and GPU

- Needs environment variable to enable OS managed page migration

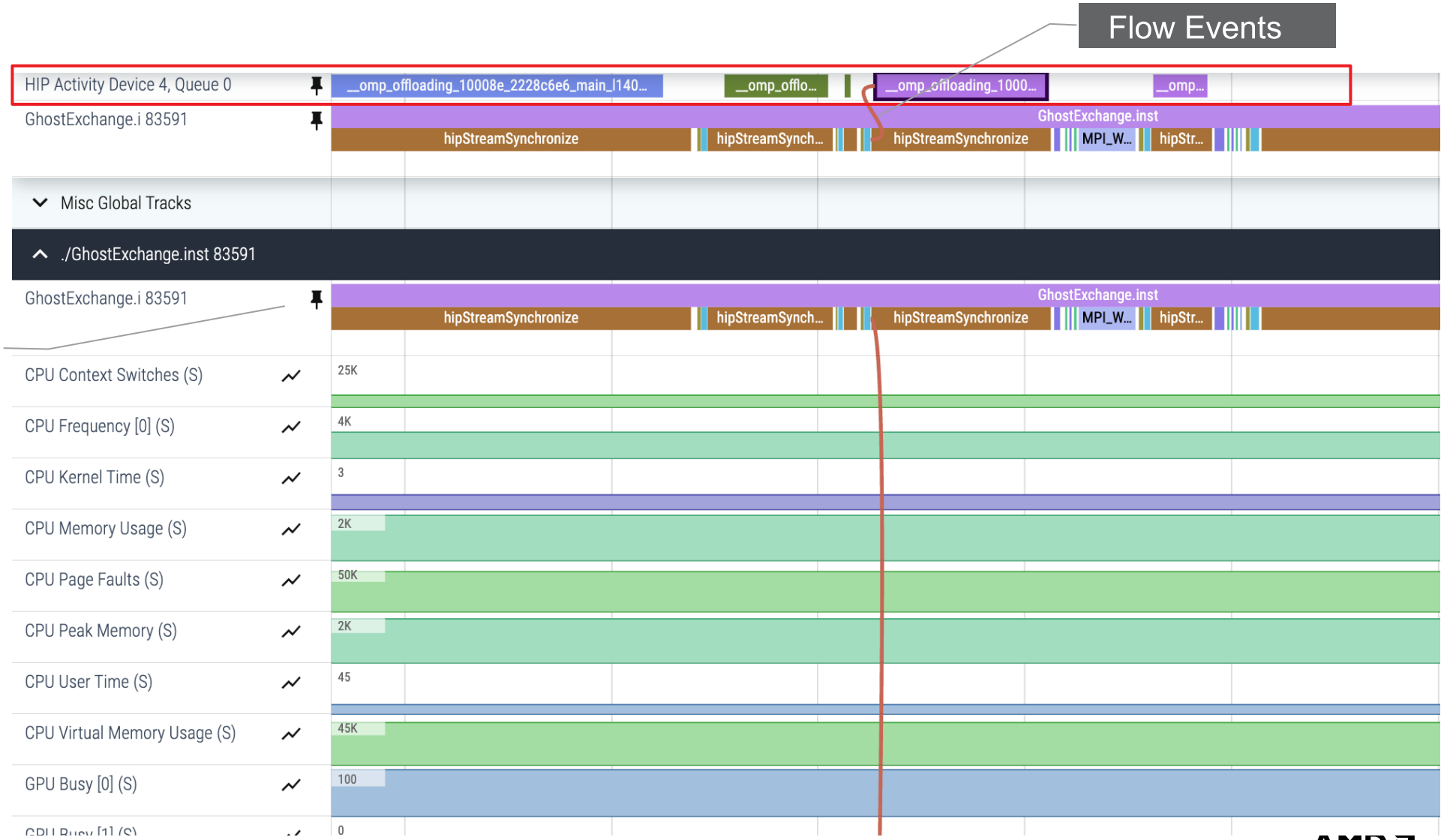
```
export HSA_XNACK=1
```

Ver1: Profile shows offloaded compute regions

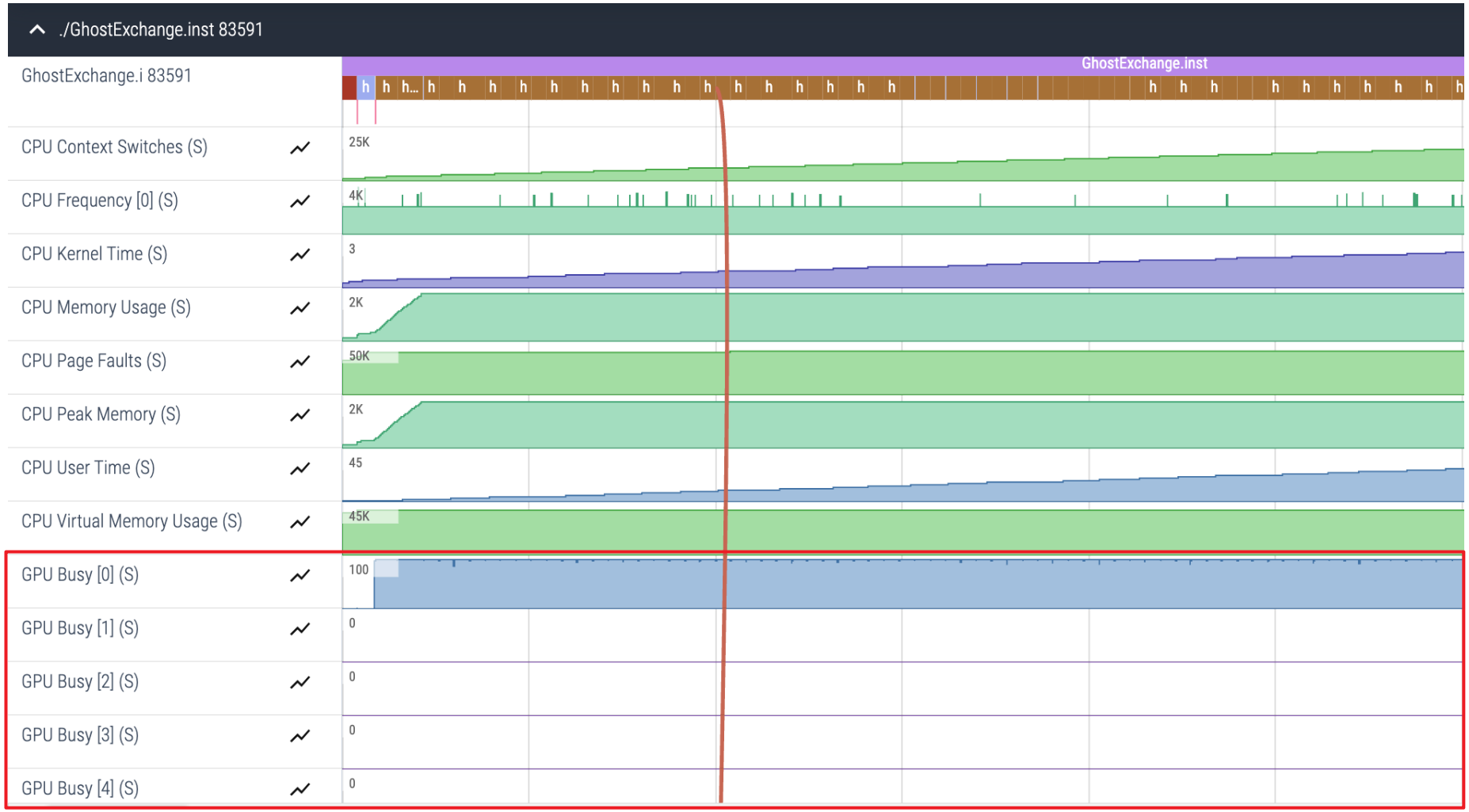
GPU kernels

Flow Events

Many rows in profile, pin select rows to bring them to the top



Ver1: Observe GPU characteristics from rocm-smi in profile

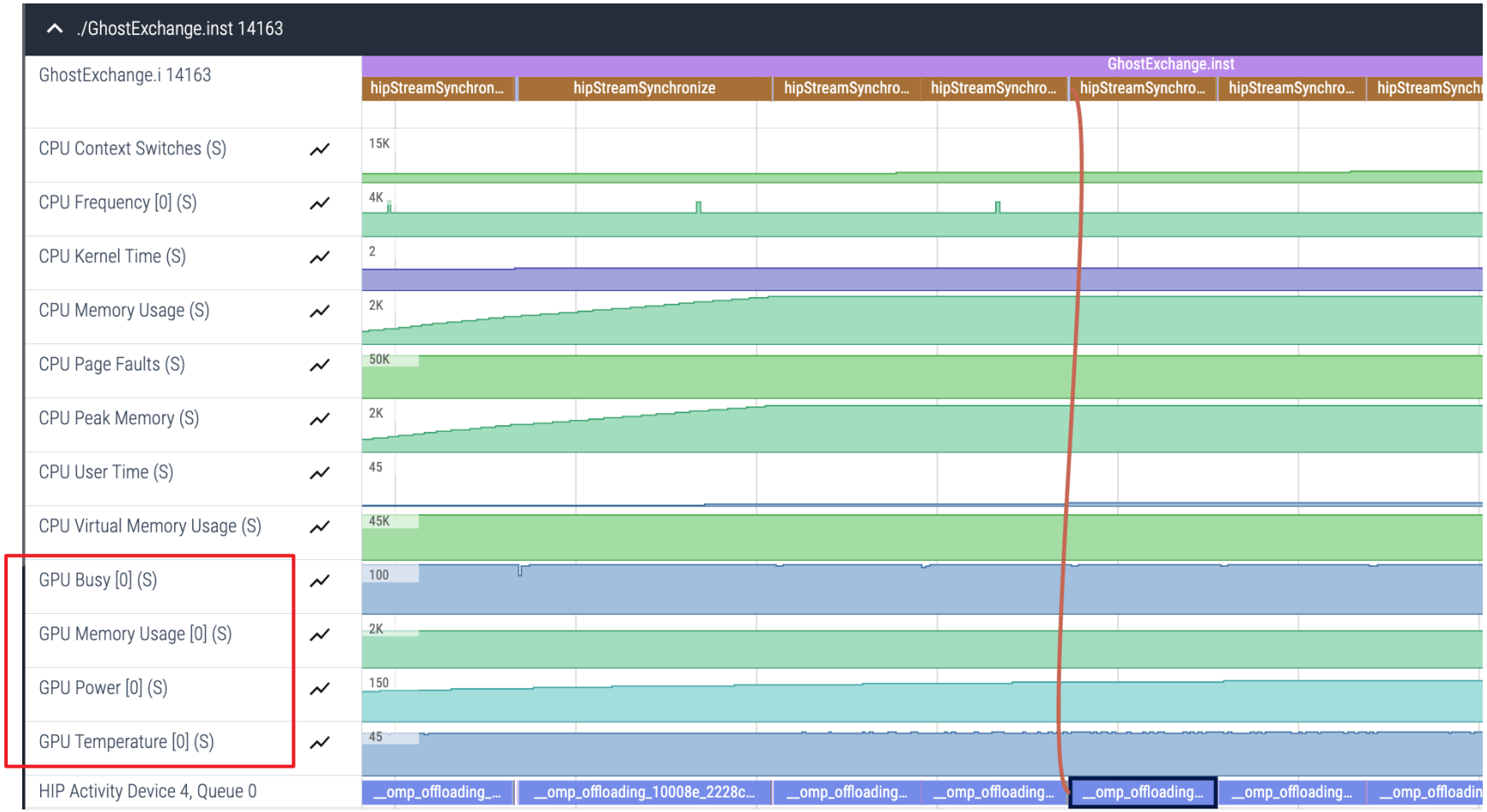


Shows details of all GPUs by default, we see activity only on GPU 0

Ver1: Trim profile to GPU of interest

- Indicate which GPU to sample in config file

```
OMNITRACE_SAMPLING_GPUS                    = 0
```



Concise trace, easier to analyze

Ver1: Wall clock profile shows OMP offload kernels and HIP APIs

REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> mbind	34	0	wall_clock	sec	0.000226	0.00					
|0>>> hipRuntimeGetVersion	1	0	wall_clock	sec	0.000132	0.00					
|0>>> hipDeviceGet	2	0	wall_clock	sec	0.000005	0.000003	0.000002	0.000004	0.000000	0.000001	100.0
|0>>> hipGetDeviceCount	1	0	wall_clock	sec	0.000006	0.000006	0.000006	0.000006	0.000000	0.000000	100.0
|0>>> GhostExchange.inst	1	0	wall_clock	sec	42.833165	42.833165	42.833165	42.833165	0.000000	0.000000	100.0
|0>>> MPI_Init	1	0	wall_clock	sec	0.195783	0.195783	0.195783	0.195783	0.000000	0.000000	100.0
|0>>> MPI_Comm_rank	4	0	wall_clock	sec	0.000037	0.000009	0.000001	0.000029	0.000000	0.000013	100.0
|0>>> MPI_Comm_size	2	0	wall_clock	sec	0.000012	0.000006	0.000002	0.000010	0.000000	0.000006	100.0
|0>>> MPI_Allreduce	1	0	wall_clock	sec	0.000173	0.000173	0.000173	0.000173	0.000000	0.000000	100.0
|0>>> hipDeviceComputeCapability	1	0	wall_clock	sec	0.000002	0.000002	0.000002	0.000002	0.000000	0.000000	100.0
|0>>> hipDeviceGetName	1	0	wall_clock	sec	0.000001	0.000001	0.000001	0.000001	0.000000	0.000000	100.0
|0>>> hipDeviceGetAttribute	1	0	wall_clock	sec	0.000003	0.000003	0.000003	0.000003	0.000000	0.000000	100.0
|0>>> hipGetDeviceProperties	1	0	wall_clock	sec	0.000005	0.000005	0.000005	0.000005	0.000000	0.000000	100.0
|0>>> hipGetDevice	543	0	wall_clock	sec	0.001256	0.000002	0.000001	0.000016	0.000000	0.000002	100.0
|0>>> hipHostMalloc	1	0	wall_clock	sec	0.000101	0.000101	0.000101	0.000101	0.000000	0.000000	100.0
|0>>> hipEventCreate	2	0	wall_clock	sec	0.000013	0.000007	0.000001	0.000012	0.000000	0.000008	100.0
|0>>> hipStreamCreate	1	0	wall_clock	sec	0.562414	0.562414	0.562414	0.562414	0.000000	0.000000	100.0
|0>>> hipModuleLoadData	1	0	wall_clock	sec	0.002285	0.002285					
|0>>> hipPointerGetAttributes	1	0	wall_clock	sec	0.000004	0.000004					
|0>>> hipModuleGetGlobal	38	0	wall_clock	sec	0.000120	0.000003					
|0>>> hipModuleGetFunction	7	0	wall_clock	sec	0.000026	0.000004					
|0>>> hipFuncGetAttribute	14	0	wall_clock	sec	0.000017	0.000001					
|0>>> hipFuncSetCacheConfig	7	0	wall_clock	sec	0.000009	0.000001	0.000001	0.000002	0.000000	0.000000	100.0
|0>>> hipModuleLaunchKernel	506	0	wall_clock	sec	0.003388	0.000007	0.000003	0.000033	0.000000	0.000003	100.0
|0>>> hipStreamSynchronize	506	0	wall_clock	sec	41.607516	0.082228	0.000232	0.697088	0.026855	0.163875	100.0
|0>>> __omp_offloading_10008e_2228c6e6_main_l115_cce$noloop$form	1	0	wall_clock	sec	0.000007	0.000007	0.000007	0.000007	0.000000	0.000000	100.0
|0>>> __omp_offloading_10008e_2228c6e6_main_l125_cce$noloop$form	1	0	wall_clock	sec	0.000000	0.000000	0.				
|0>>> __omp_offloading_10008e_2228c6e6__Z24boundarycondition_updatePPdiiiiiii_l179_cce$noloop$form	101	0	wall_clock	sec	0.000017	0.000000	0.				
|0>>> MPI_Irecv	404	0	wall_clock	sec	0.003960	0.000010	0.				
|0>>> MPI_Isend	404	0	wall_clock	sec	0.002559	0.000006	0.				
|0>>> MPI_Waitall	202	0	wall_clock	sec	0.309031	0.001530	0.				
|0>>> __omp_offloading_10008e_2228c6e6__Z24boundarycondition_updatePPdiiiiiii_l197_cce$noloop$form	101	0	wall_clock	sec	0.000022	0.000000	0.				
|0>>> __omp_offloading_10008e_2228c6e6__Z16ghostcell_updatePPdiiiiiiiii_l252_cce$noloop$form	101	0	wall_clock	sec	0.000026	0.000000	0.				
|0>>> __omp_offloading_10008e_2228c6e6__Z16ghostcell_updatePPdiiiiiiiii_l273_cce$noloop$form	101	0	wall_clock	sec	0.000020	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> __omp_offloading_10008e_2228c6e6_main_l140_cce$noloop$form	100	0	wall_clock	sec	0.000017	0.000000	0.000000	0.000000	0.000000	0.000000	100.0

Runtime increased to 42 secs

Mostly waiting for GPU kernels to complete

Only CPU-side launch durations of OpenMP offload kernels shown in wall-clock profile

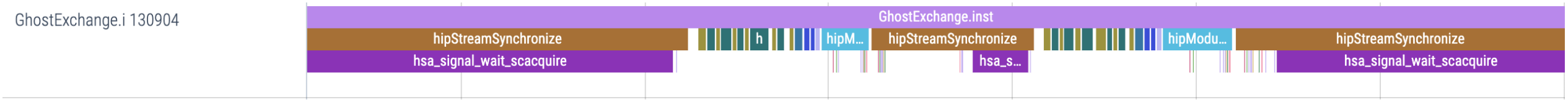
Managed memory affects kernel performance – but profile does not show in what way, yet

Ver1: Seeing HSA runtime activity

- To implement OpenMP offload capability,
  - the AMD compiler uses the HSA layer
  - the Cray compiler uses the HIP layer

- Set up config file to see HSA activity in profile:

```
OMNITRACE_ROCTRACER_HSA_ACTIVITY = true
OMNITRACE_ROCTRACER_HSA_API = true
```

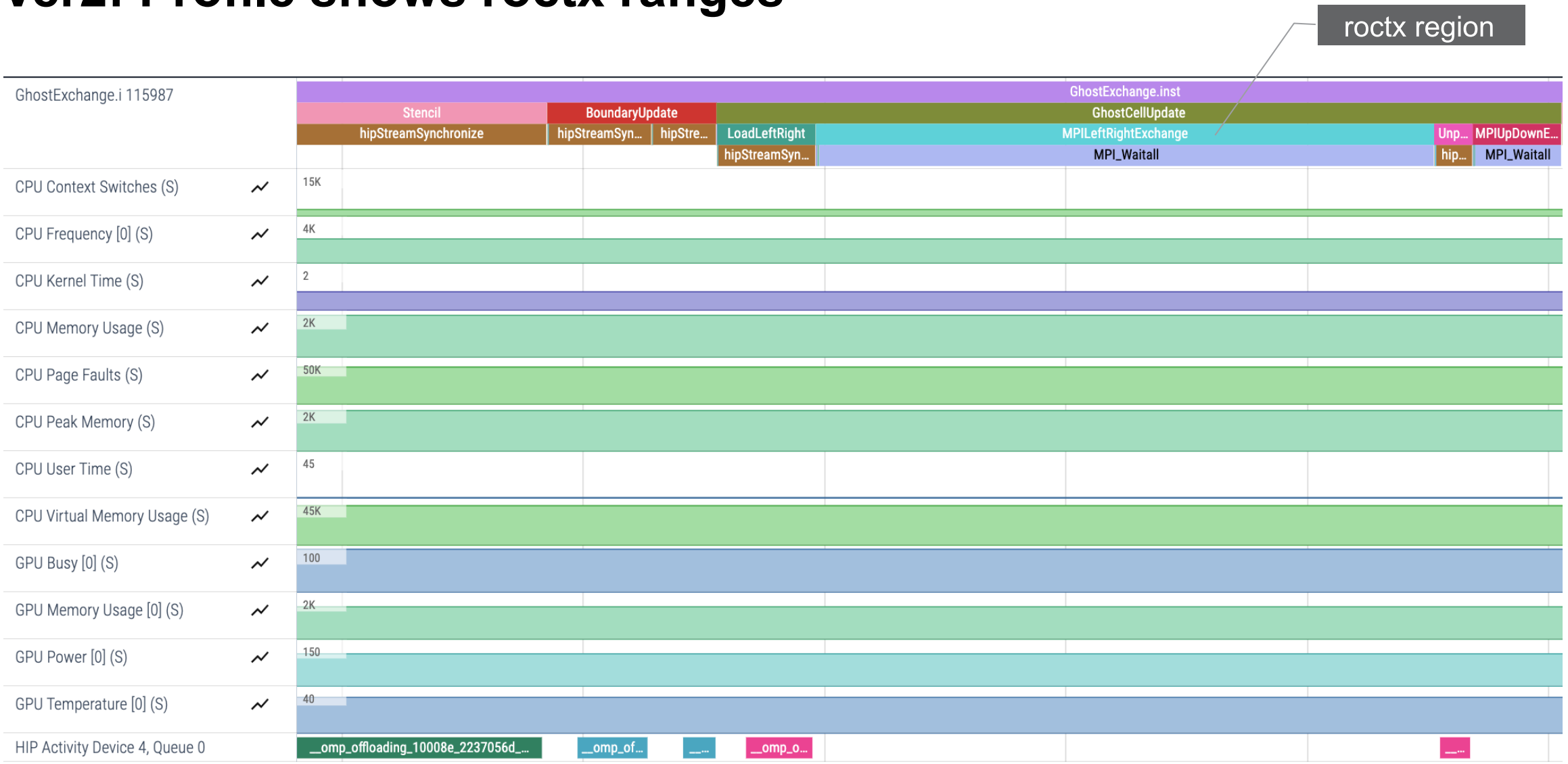


- More details about HIP and HSA runtime libraries

Ver2

Manually instrument code with roctx ranges to study regions of code

Ver2: Profile shows roctx ranges



Ver2: Wall clock files show timings of roctx regions

REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> mbind	34	0	wall_clock	sec	0.000207	0.000006	0.000003	0.000070	0.000000	0.000011	100.0
|0>>> hipRuntimeGetVersion	1	0	wall_clock	sec	0.000152	0.000152	0.000152	0.000152	0.000000	0.000000	100.0
|0>>> hipDeviceGet	2	0	wall_clock	sec	0.000005	0.000003	0.000002	0.000003	0.000000	0.000001	100.0
|0>>> hipGetDeviceCount	1	0	wall_clock	sec	0.000016	0.000016	0.000016	0.000016	0.000000	0.000000	100.0
|0>>> GhostExchange.inst	1	0	wall_clock	sec	42.948441	42.948441	42.948441	42.948441	0.000000	0.000000	100.0
|0>>> MPI_Init	1	0	wall_clock	sec	0.222832	0.222832	0.222832	0.222832	0.000000	0.000000	100.0
|0>>> MPI_Comm_rank	4	0	wall_clock	sec	0.000039	0.000010	0.000001	0.000030	0.000000	0.000014	100.0
|0>>> MPI_Comm_size	2	0	wall_clock	sec	0.000012	0.000006	0.000002	0.000011	0.000000	0.000007	100.0
|0>>> MPI_Allreduce	1	0	wall_clock	sec	0.000697	0.000697	0.000697	0.000697	0.000000	0.000000	100.0
|0>>> hipDeviceComputeCapability	1	0	wall_clock	sec	0.000003	0.000003	0.000003	0.000003	0.000000	0.000000	100.0
|0>>> hipDeviceGetName	1	0	wall_clock	sec	0.000001	0.000001	0.000001	0.000001	0.000000	0.000000	100.0
|0>>> hipDeviceGetAttribute	1	0	wall_clock	sec	0.000003	0.000003	0.000003	0.000003	0.000000	0.000000	100.0
|0>>> hipGetDeviceProperties	1	0	wall_clock	sec	0.000002	0.000002	0.000002	0.000002	0.000000	0.000000	100.0
|0>>> hipGetDevice	543	0	wall_clock	sec	0.001649	0.000003	0.000001	0.000018	0.000000	0.000002	100.0
|0>>> hipHostMalloc	1	0	wall_clock	sec	0.000110	0.000110	0.000110	0.000110	0.000000	0.000000	100.0
|0>>> hipEventCreate	2	0	wall_clock	sec	0.000007	0.000003	0.000001	0.000005	0.000000	0.000003	100.0
|0>>> hipStreamCreate	1	0	wall_clock	sec	0.557462	0.557462	0.557462	0.557462	0.000000	0.000000	100.0
|0>>> hipModuleLoadData	1	0	wall_clock	sec	0.002125	0.002125	0.002125	0.002125	0.000000	0.000000	100.0
|0>>> hipPointerGetAttributes	1	0	wall_clock	sec	0.000005	0.000005	0.000005	0.000005	0.000000	0.000000	100.0
|0>>> hipModuleGetGlobal	38	0	wall_clock	sec	0.000120	0.000003	0.000001	0.000010	0.000000	0.000002	100.0
|0>>> hipModuleGetFunction	7	0	wall_clock	sec	0.000023	0.000003	0.000002	0.000005	0.000000	0.000001	100.0
|0>>> hipFuncGetAttribute	14	0	wall_clock	sec	0.000019	0.000001	0.000001	0.000003	0.000000	0.000001	100.0
|0>>> hipFuncSetCacheConfig	7	0	wall_clock	sec	0.000008	0.000001	0.000001	0.000001	0.000000	0.000000	100.0
|0>>> hipModuleLaunchKernel	506	0	wall_cl								
|0>>> hipStreamSynchronize	506	0	wall_cl								
|0>>> __omp_offloading_10008e_2237056d_main_l116_cce$noloop$form	1	0	wall_cl								
|0>>> __omp_offloading_10008e_2237056d_main_l126_cce$noloop$form	1	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> MPIRequest	101	0	wall_clock	sec	0.000101	0.000001	0.000001	0.000012	0.000000	0.000001	100.0
|0>>> BufAlloc	101	0	wall_clock	sec	0.000110	0.000001	0.000001	0.000009	0.000000	0.000001	100.0
|0>>> LoadLeftRight	101	0	wall_clock	sec	0.032994	0.000327	0.000122	0.002041	0.000000	0.000264	100.0
|0>>> __omp_offloading_10008e_2237056d__Z24boundarycondition_updatePPdiiiiiii_l191_cce$noloop$form	101	0	wall_clock	sec	0.000020	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> MPILeftRightExchange	101	0	wall_clock	sec	0.039038	0.000387	0.000068	0.012803	0.000002	0.001273	100.0
|0>>> MPI_Irecv	404	0	wall_clock	sec	0.003454	0.000009	0.000005	0.000035	0.000000	0.000005	100.0
|0>>> MPI_Isend	404	0	wall_clock	sec	0.002998	0.000007	0.000005	0.000033	0.000000	0.000004	100.0
|0>>> MPI_Waitall	202	0	wall_clock	sec	0.047174						
|0>>> UnpackLeftRight	101	0	wall_clock	sec	0.028112						
|0>>> __omp_offloading_10008e_2237056d__Z24boundarycondition_updatePPdiiiiiii_l209_cce$noloop$form	101	0	wall_clock	sec	0.000030	0.000000	0.000000	0.000001	0.000000	0.000000	100.0
|0>>> MPIUpDownExchange	101	0	wall_clock	sec	0.020360	0.000202	0.000067	0.001836	0.000000	0.000257	100.0
|0>>> Stencil	100	0	wall_clock	sec	41.254781	0.412548	0.405178	0.684404	0.000755	0.027483	100.0
|0>>> __omp_offloading_10008e_2237056d__Z16ghostcell_updatePPdiiiiiiiii_l269_cce$noloop$form	101	0	wall_clock	sec	0.000031	0.000000	0.000000	0.000001	0.000000	0.000000	100.0
|0>>> BoundaryUpdate	100	0	wall_clock	sec	0.096786	0.000968	0.000297	0.003505	0.000000	0.000704	100.0
|0>>> __omp_offloading_10008e_2237056d__Z16ghostcell_updatePPdiiiiiiiii_l294_cce$noloop$form	101	0	wall_clock	sec	0.000024	0.000000	0.000000	0.000001	0.000000	0.000000	100.0
|0>>> __omp_offloading_10008e_2237056d_main_l144_cce$noloop$form	100	0	wall_clock	sec	0.000021	0.000000	0.000000	0.000001	0.000000	0.000000	100.0
|0>>> GhostCellUpdate	100	0	wall_clock	sec	0.120245	0.001202	0.000502	0.017501	0.000003	0.001710	100.0

Call count shows we allocate buffers many times

Time spent mostly in compute kernels

Ver3

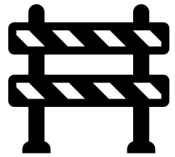
Allocate MPI buffers on device

Ver3: Enable GPU Aware MPI

- On Frontier nodes, there are 4 Network Interface Cards (NICs) directly attached to odd numbered GCDs
- Using device buffers in MPI calls is expected to improve communication performance
- Need environment variable to enable GPU Aware MPI:

```
export MPICH_GPU_SUPPORT_ENABLED=1
```

- In this example, communication is not our bottleneck, so we don't expect a big difference in MPI overhead



- Our attempt to achieve this is not working with ROCm 5.7.0 today on Frontier
- Known issue that is being actively investigated

Ver4

Allocate all host buffers just once

Ver4: Profile shows only 1 allocation

REAL-CLOCK TIMER (I.E. WALL-CLOCK TIMER)

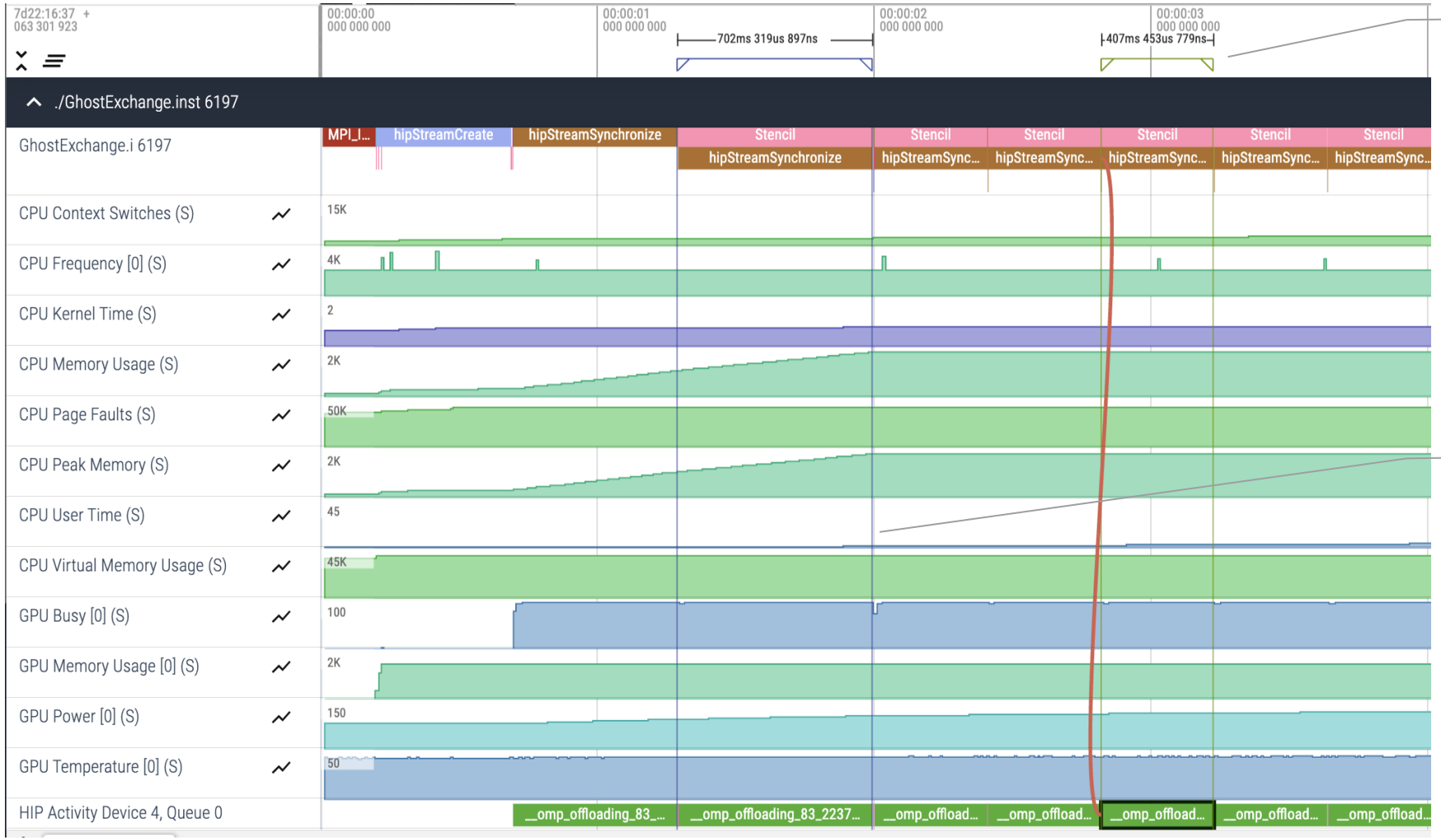
LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> mbind	34	0	wall_clock	sec	0.000204	0.000006	0.000002	0.000070	0.000000	0.000012	100.0
|0>>> hipRuntimeGetVersion	1	0	wall_clock	sec	0.000132	0.000132	0.000132	0.000132	0.000000	0.000000	100.0
|0>>> hipDeviceGet	2	0	wall_clock	sec	0.000005	0.000003	0.000002	0.000003	0.000000	0.000001	100.0
|0>>> hipGetDeviceCount	1	0	wall_clock	sec	0.000006	0.000006	0.000006	0.000006	0.000000	0.000000	100.0
|0>>> GhostExchange.inst	1	0	wall_clock	sec	43.091367	43.091367	43.091367	43.091367	0.000000	0.000000	100.0
|0>>> MPI_Init	1	0	wall_clock	sec	0.288942	0.288942	0.288942	0.288942	0.000000	0.000000	100.0
|0>>> MPI_Comm_rank	4	0	wall_c						0.000000	0.000017	100.0
|0>>> MPI_Comm_size	2	0	wall_c						0.000000	0.000006	100.0
|0>>> MPI_Allreduce	1	0	wall_c						0.000000	0.000000	100.0
|0>>> BufAlloc	1	0	wall_clock	sec	0.000013	0.000013	0.000013	0.000013	0.000000	0.000000	100.0
|0>>> hipDeviceComputeCapability	1	0	wall_clock	sec	0.000002	0.000002	0.000002	0.000002	0.000000	0.000000	100.0
|0>>> hipDeviceGetName	1	0	wall_clock	sec	0.000001	0.000001	0.000001	0.000001	0.000000	0.000000	100.0
|0>>> hipDeviceGetAttribute	1	0	wall_clock	sec	0.000003	0.000003	0.000003	0.000003	0.000000	0.000000	100.0
|0>>> hipGetDeviceProperties	1	0	wall_clock	sec	0.000005	0.000005	0.000005	0.000005	0.000000	0.000000	100.0
|0>>> hipGetDevice	543	0	wall_clock	sec	0.001684	0.000003	0.000001	0.000023	0.000000	0.000002	100.0
|0>>> hipHostMalloc	1	0	wall_clock	sec	0.000098	0.000098	0.000098	0.000098	0.000000	0.000000	100.0
|0>>> hipEventCreate	2	0	wall_clock	sec	0.000012	0.000006	0.000001	0.000011	0.000000	0.000007	100.0
|0>>> hipStreamCreate	1	0	wall_clock	sec	0.481629	0.481629	0.481629	0.481629	0.000000	0.000000	100.0
|0>>> hipModuleLoadData	1	0	wall_clock	sec	0.002023	0.002023	0.002023	0.002023	0.000000	0.000000	100.0
|0>>> hipPointerGetAttributes	1	0	wall_clock	sec	0.000004	0.000004	0.000004	0.000004	0.000000	0.000000	100.0
|0>>> hipModuleGetGlobal	38	0	wall_clock	sec	0.000118	0.000003	0.000001	0.000007	0.000000	0.000002	100.0
|0>>> hipModuleGetFunction	7	0	wall_clock	sec	0.000023	0.000003	0.000002	0.000005	0.000000	0.000001	100.0
|0>>> hipFuncGetAttribute	14	0	wall_clock	sec	0.000020	0.000001	0.000001	0.000004	0.000000	0.000001	100.0
|0>>> hipFuncSetCacheConfig	7	0	wall_clock	sec	0.000008	0.000001	0.000001	0.000001	0.000000	0.000000	100.0
|0>>> hipModuleLaunchKernel	506	0	wall_clock	sec	0.003772	0.000007	0.000004	0.000033	0.000000	0.000003	100.0
|0>>> hipStreamSynchronize	506	0	wall_clock	sec	41.886630	0.082780	0.000013	0.699256	0.027512	0.165867	100.0
|0>>> __omp_offloading_83_22370944_main_l133_cce$noloop$form	1	0	wall_clock	sec	0.000007	0.000007	0.000007	0.000007	0.000000	0.000000	100.0
|0>>> __omp_offloading_83_22370944_main_l143_cce$noloop$form	1	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> MPIRequest	101	0	wall_clock	sec	0.000130	0.000001	0.000001	0.000016	0.000000	0.000002	100.0
|0>>> LoadLeftRight	101	0	wall_clock	sec	0.027696	0.000274	0.000127	0.001400	0.000000	0.000202	100.0
|0>>> __omp_offloading_83_22370944__Z24boundarycondition_updatePPdiiiiiii_l213_cce$noloop$form	101	0	wall_clock	sec	0.000020	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> MPILeftRightExchange	101	0	wall_clock	sec	0.012126	0.000120	0.000059	0.001038	0.000000	0.000166	100.0
|0>>> MPI_Irecv	404	0	wall_clock	sec	0.003255	0.000008	0.000004	0.000033	0.000000	0.000004	100.0
|0>>> MPI_Isend	404	0	wall_clock	sec	0.002938	0.000007	0.000005	0.000033	0.000000	0.000004	100.0
|0>>> MPI_Waitall	202	0	wall_clock	sec	0.020724	0.000103	0.000005	0.000990	0.000000	0.000159	100.0
|0>>> UnpackLeftRight	101	0	wall_clock	sec	0.027247	0.000270	0.000172	0.000389	0.000000	0.000064	100.0
|0>>> __omp_offloading_83_22370944__Z24boundarycondition_updatePPdiiiiiii_l231_cce$noloop$form	101	0	wall_clock	sec	0.000030	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> MPIUpDownExchange	101	0	wall_clock	sec	0.020473	0.000203	0.000058	0.000538	0.000000	0.000139	100.0
|0>>> Stencil	100	0	wall_clock	sec	41.184269	0.411843	0.405242	0.699454	0.000845	0.029071	100.0
|0>>> __omp_offloading_83_22370944__Z16ghostcell_updatePPdiiiiiiiii_l284_cce$noloop$form	101	0	wall_clock	sec	0.000030	0.000000	0.000000	0.000001	0.000000	0.000000	100.0
|0>>> BoundaryUpdate	100	0	wall_clock	sec	0.102319	0.001023	0.000251	0.002748	0.000001	0.000757	100.0
|0>>> __omp_offloading_83_22370944__Z16ghostcell_updatePPdiiiiiiiii_l309_cce$noloop$form	101	0	wall_clock	sec	0.000023	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> __omp_offloading_83_22370944_main_l161_cce$noloop$form	100	0	wall_clock	sec	0.000022	0.000000	0.000000	0.000001	0.000000	0.000000	100.0
|0>>> GhostCellUpdate	100	0	wall_clock	sec	0.088821	0.000888	0.000478	0.002956	0.000000	0.000395	100.0

Profile shows only 1 allocation

Ver5

Convert indexing from 2D to 1D – a step towards allocating buffers directly on device

Ver5: Changing indexing to 1D does not change performance



Quickly see durations by selecting kernel and pressing “m”

Extremely helpful to see activity in this duration on CPU, GPU, other HIP streams, etc.

Ver6

Use explicit data management directives to allocate buffers on device and keep them on device for entire run

Ver6: Adding explicit OpenMP map directives

- Allocate buffers once on device at the beginning:

```
#pragma omp target enter data map(alloc: xbuf_left_send[0:bufcount], xbuf_rght_send[0:bufcount])
#pragma omp target enter data map(alloc: xbuf_rght_recv[0:bufcount], xbuf_left_recv[0:bufcount])
#pragma omp target enter data map(alloc: x[0:totcells], xnew[0:totcells])
```

- Release buffers at the end:

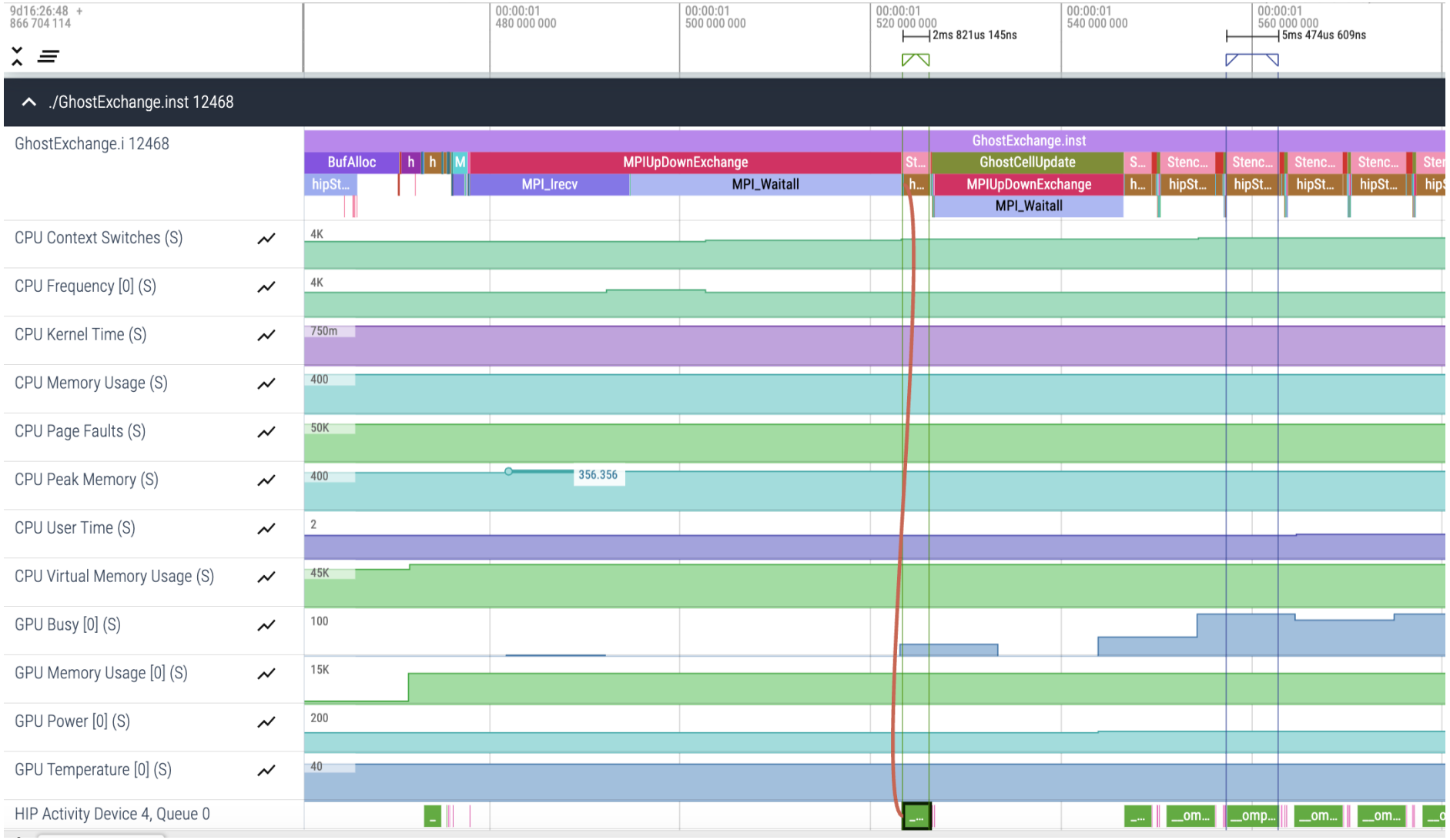
```
#pragma omp target exit data map(release: x, xnew)
#pragma omp target exit data map(release: xbuf_left_send, xbuf_rght_send)
#pragma omp target exit data map(release: xbuf_rght_recv, xbuf_left_recv)
```

- Keeping data on HBM improves performance of memory bound kernels on MI250X GPUs
- Managed memory support no longer needed:

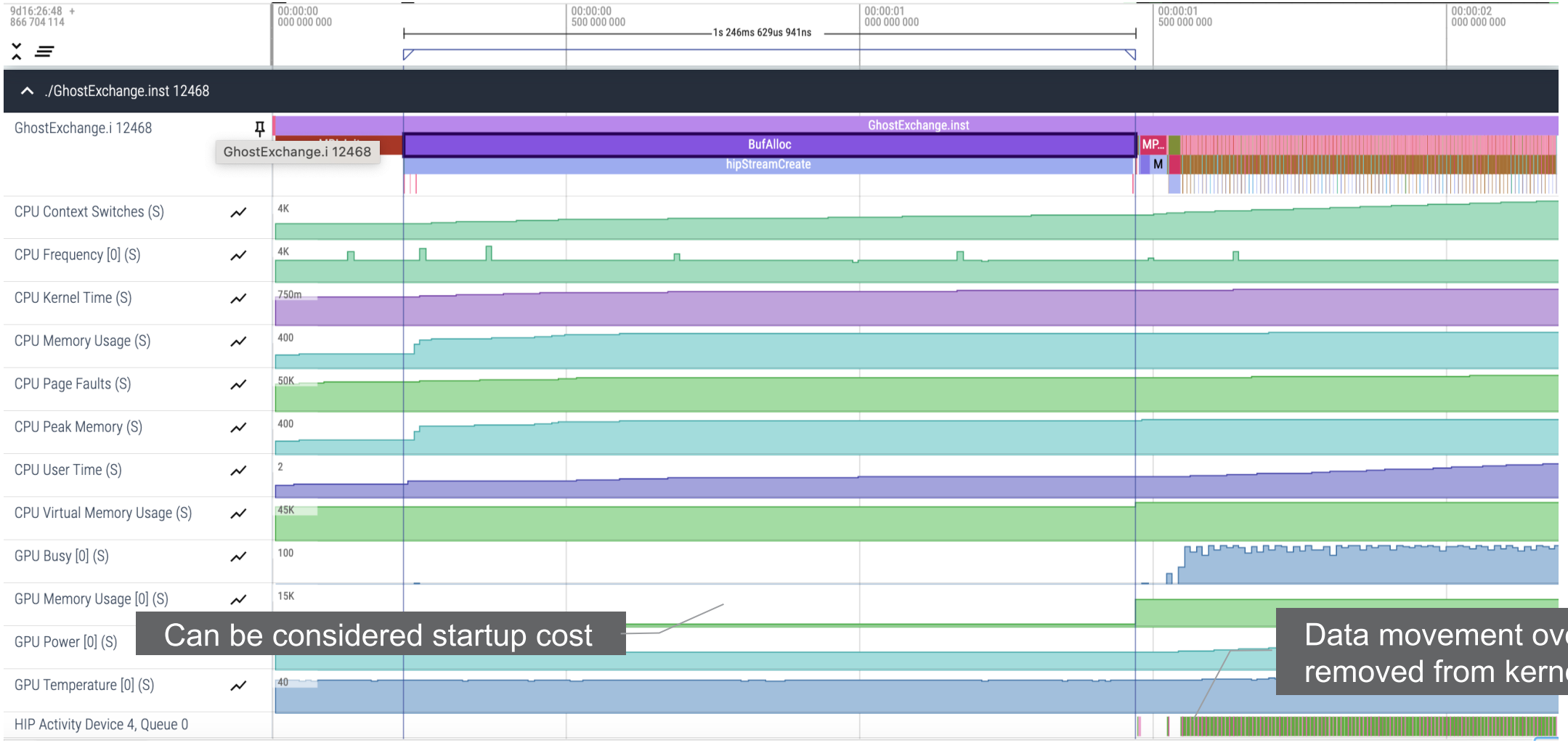
~~`#pragma omp requires unified_shared_memory`~~

```
unset HSA_XNACK
```

Ver6: Profile shows significantly faster kernels



Ver6: Allocation of MPI buffers on device is our new bottleneck



Ver6: Wall clock shows shorter durations of kernels

LABEL	COUNT	DEPTH	METRIC	UNITS	SUM	MEAN	MIN	MAX	VAR	STDDEV	% SELF
|0>>> mbind	34	0	wall_clock	sec	0.000225	0.000007	0.000002	0.000071	0.000000	0.000012	100.0
|0>>> hipRuntimeGetVersion	1	0	wall_clock	sec	0.000128	0.000128	0.000128	0.000128	0.000000	0.000000	100.0
|0>>> hipDeviceGet	2	0	wall_clock	sec	0.000005	0.000003	0.000002	0.000003	0.000000	0.000001	100.0
|0>>> hipGetDeviceCount	1	0	wall_clock	sec	0.000006	0.000006	0.000006	0.000006	0.000000	0.000000	100.0
|0>>> GhostExchange.inst	1	0	wall_clock	sec	2.319124	2.319124	2.319124	2.319124	0.000000	0.000000	100.0
|0>>> MPI_Init	1	0	wall_clock	sec	0.217335	0.217335	0.217335	0.217335	0.000000	0.000000	100.0
|0>>> MPI_Comm_rank	4	0	wall_clock	sec	0.000046	0.000012	0.000001	0.000031	0.000000	0.000013	100.0
|0>>> MPI_Comm_size	2	0	wall_clock	sec	0.000014	0.000007	0.000002	0.000012	0.000000	0.000008	100.0
|0>>> MPI_Allreduce	1	0	wall_clock	sec	0.000917	0.000917	0.000917	0.000917	0.000000	0.000000	100.0
|0>>> BufAlloc	1	0	wall_clock	sec	1.246631	1.246631	1.246631	1.246631	0.000000	0.000000	100.0
|0>>> hipDeviceComputeCapability	1	0	wall_clock	sec	0.000002	0.000002	0.000002	0.000002	0.000000	0.000000	100.0
|0>>> hipDeviceGetName	1	0	wall_clock	sec	0.000003	0.000003	0.000003	0.000003	0.000000	0.000000	100.0
|0>>> hipDeviceGetAttribute	1	0	wall_clock	sec	0.000003	0.000003	0.000003	0.000003	0.000000	0.000000	100.0
|0>>> hipGetDeviceProperties	1	0	wall_clock	sec	0.000005	0.000005	0.000005	0.000005	0.000000	0.000000	100.0
|0>>> hipGetDevice	550	0	wall_clock	sec	0.001796	0.000003	0.000001	0.000022	0.000000	0.000002	100.0
|0>>> hipHostMalloc	1	0	wall_clock	sec	0.000106	0.000106	0.000106	0.000106	0.000000	0.000000	100.0
|0>>> hipEventCreate	2	0	wall_clock	sec	0.000006	0.000003	0.000001	0.000005	0.000000	0.000003	100.0
|0>>> hipStreamCreate	1	0	wall_clock	sec	1.241892	1.241892	1.241892	1.241892	0.000000	0.000000	100.0
|0>>> hipMalloc	7	0	wall_clock	sec	0.000176	0.000025	0.000002	0.000044	0.000000	0.000020	100.0
|0>>> hipModuleLoadData	1	0	wall_clock	sec	0.002024	0.002024	0.002024	0.002024	0.000000	0.000000	100.0
|0>>> hipPointerGetAttributes	1	0	wall_clock	sec	0.000004	0.000004	0.000004	0.000004	0.000000	0.000000	100.0
|0>>> hipModuleGetGlobal	38	0	wall_clock	sec	0.000133	0.000003	0.000001	0.000013	0.000000	0.000002	100.0
|0>>> hipModuleGetFunction	7	0	wall_clock	sec	0.000020	0.000003	0.000002	0.000004	0.000000	0.000001	100.0
|0>>> hipFuncGetAttribute	14	0	wall_clock	sec	0.000025	0.000002	0.000001	0.000005	0.000000	0.000001	100.0
|0>>> hipFuncSetCacheConfig	7	0	wall_clock	sec	0.000011	0.000002	0.000001	0.000003	0.000000	0.000001	100.0
|0>>> hipModuleLaunchKernel	506	0	wall_clock	sec	0.003685	0.000007	0.000004	0.000041	0.000000	0.000003	100.0
|0>>> hipStreamSynchronize	506	0	wall_clock	sec	0.587353	0.001161	0.000010	0.005770	0.000005	0.002202	100.0
|0>>> __omp_offloading_83_22370965_main_l142_cce$noloop$form	1	0	wall_clock	sec	0.000010	0.000010	0.000010	0.000010	0.000000	0.000000	100.0
|0>>> __omp_offloading_83_22370965_main_l152_cce$noloop$form	1	0	wall_clock	sec	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> MPIRequest	101	0	wall_clock	sec	0.000110	0.000001					
|0>>> LoadLeftRight	101	0	wall_clock	sec	0.006125	0.000061					
|0>>> __omp_offloading_83_22370965__Z24boundarycondition_updatePdiiiiiii_l227_cce$noloop$form	101	0	wall_clock	sec	0.000022	0.000000					
|0>>> MPILeftRightExchange	101	0	wall_clock	sec	0.021877	0.000217					
|0>>> MPI_Irecv	404	0	wall_clock	sec	0.021302	0.000053					
|0>>> MPI_Isend	404	0	wall_clock	sec	0.003006	0.000007					
|0>>> MPI_Waitall	202	0	wall_clock	sec	0.071129	0.000352					
|0>>> UnpackLeftRight	101	0	wall_clock	sec	0.008849	0.000088	0.000057	0.000277	0.000000	0.000028	100.0
|0>>> __omp_offloading_83_22370965__Z24boundarycondition_updatePdiiiiiii_l245_cce$noloop$form	101	0	wall_clock	sec	0.000030	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> MPIUpDownExchange	101	0	wall_clock	sec	0.079417	0.000786	0.000063	0.045279	0.000024	0.004885	100.0
|0>>> Stencil	100	0	wall_clock	sec	0.562033	0.005620	0.002907	0.005817	0.000000	0.000396	100.0
|0>>> __omp_offloading_83_22370965__Z16ghostcell_updatePdiiiiiiii_l300_cce$noloop$form	101	0	wall_clock	sec	0.000032	0.000000	0.000000	0.000002	0.000000	0.000000	100.0
|0>>> BoundaryUpdate	100	0	wall_clock	sec	0.028627	0.000286	0.000111	0.000864	0.000000	0.000169	100.0
|0>>> __omp_offloading_83_22370965__Z16ghostcell_updatePdiiiiiiiii_l325_cce$noloop$form	101	0	wall_clock	sec	0.000023	0.000000	0.000000	0.000001	0.000000	0.000000	100.0
|0>>> __omp_offloading_83_22370965_main_l170_cce$noloop$form	100	0	wall_clock	sec	0.000017	0.000000	0.000000	0.000000	0.000000	0.000000	100.0
|0>>> GhostCellUpdate	100	0	wall_clock	sec	0.071708	0.000717	0.000290	0.020150	0.000004	0.001970	100.0

Memory bound kernels, faster when accessing data from HBM

Omnitrace Tips and Status

Tips: Reduce generated output for profiling at scale

- Turn off all options in config file except `OMNITRACE_PROFILE` to reduce generated output

```
OMNITRACE_TRACE                 = false
OMNITRACE_PROFILE               = true
OMNITRACE_FLAT_PROFILE          = true
OMNITRACE_USE_ROCTRACER         = false
OMNITRACE_USE_ROCM_SMI          = false
OMNITRACE_USE_MPIP              = true
OMNITRACE_USE_PID               = true
OMNITRACE_USE_ROCPROFILER       = false
OMNITRACE_USE_ROCTX             = false
```

Tips: If Omnitrace does nothing, check app or environment

- If Omnitrace starts, but does not generate any output files, something prevented the app from running
- To check, unload Omnitrace module, build and run app. Fix errors, then profile with Omnitrace

- If app fails only when being profiled with Omnitrace, try profiling interactively using `srun` instead of `sbatch`
  - Conflict due to mismatch in loaded libraries at runtime

- If you use `omnitrace-run` and it complains saying "Use `omnitrace-run`", then try running your job interactively using `srun` instead of using `sbatch`
  - Conflict due to mismatch in loaded libraries at runtime

Tips: To visualize very large proto files, load into memory first

Linux®

- `curl -LO https://get.perfetto.dev/trace_processor`
- `chmod +x ./trace_processor`
- `./trace_processor –httpd <path to trace file>`
- Open up Chrome browser and go to https://ui.perfetto.dev
- When prompted, click on "Yes, use loaded trace"

Windows®

- Open up https://get.perfetto.dev/trace_processor in a browser to download the Python™ script
- `py trace_processor --httpd <trace file>`
  - You may need to download and install Python on your windows system
- Open up Chrome browser and go to https://ui.perfetto.dev
- When prompted, click on "Yes, use loaded trace"

Research version of Omnitrace is brittle

- Viewing traces of multiple ranks together was possible using simple concatenation of proto files:

```
cat perfetto-trace-0.proto perfetto-trace-1.proto > merged.proto
```

  - Merging broken now due to change in expected data format in Perfetto

- Building Omnitrace with Dyninst from source requires GCC, may interfere with CCE
  - On Frontier, `omnitrace/1.11.2` is set up to work with CCE and show OpenMP offload and HIP activity

- If you load ROCm, reload Omnitrace as the right build has to be made available

- Python version in your environment matters

- Production version of Omnitrace will be more robust

Homework

- See HIP equivalent of Ver1 here:

https://github.com/amd/HPCTrainingExamples/tree/main/MPI-examples/GhostExchange/GhostExchange_ArrayAssign_HIP/Ver1

- Use Omnitrace to obtain traces for a 4-rank run
- Progressively port the changes in Ver2 – Ver6 in the HIP version using HIP APIs for memory copies, etc.
- Look for memory copy activity and HIP API calls in Omnitrace traces
- Submit a PR with your code or add an issue with any concerns

References

- Omnitrace documentation website: https://rocm.github.io/omnitrace/index.html
- Previous talk describing various Omnitrace options: 15: GPU Profiling - Performance Timelines
- Ghost Exchange OpenMP offload Example suite on github
- ROCm docs: https://rocm.docs.amd.com/en/latest/
- ROCm Blog post: Introduction to profiling tools for AMD hardware

Disclaimer

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED ‘AS IS.” AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Third-party content is licensed to you directly by the third party that owns the content and is not licensed to you by AMD. ALL LINKED THIRD-PARTY CONTENT IS PROVIDED “AS IS” WITHOUT A WARRANTY OF ANY KIND. USE OF SUCH THIRD-PARTY CONTENT IS DONE AT YOUR SOLE DISCRETION AND UNDER NO CIRCUMSTANCES WILL AMD BE LIABLE TO YOU FOR ANY THIRD-PARTY CONTENT. YOU ASSUME ALL RISK AND ARE SOLELY RESPONSIBLE FOR ANY DAMAGES THAT MAY ARISE FROM YOUR USE OF THIRD-PARTY CONTENT.


AMD