Profiling and debugging CUDA

Luca Parisi, EPCC, The University of Edinburgh

l.parisi@epcc.ed.ac.uk

11 Jun 2024

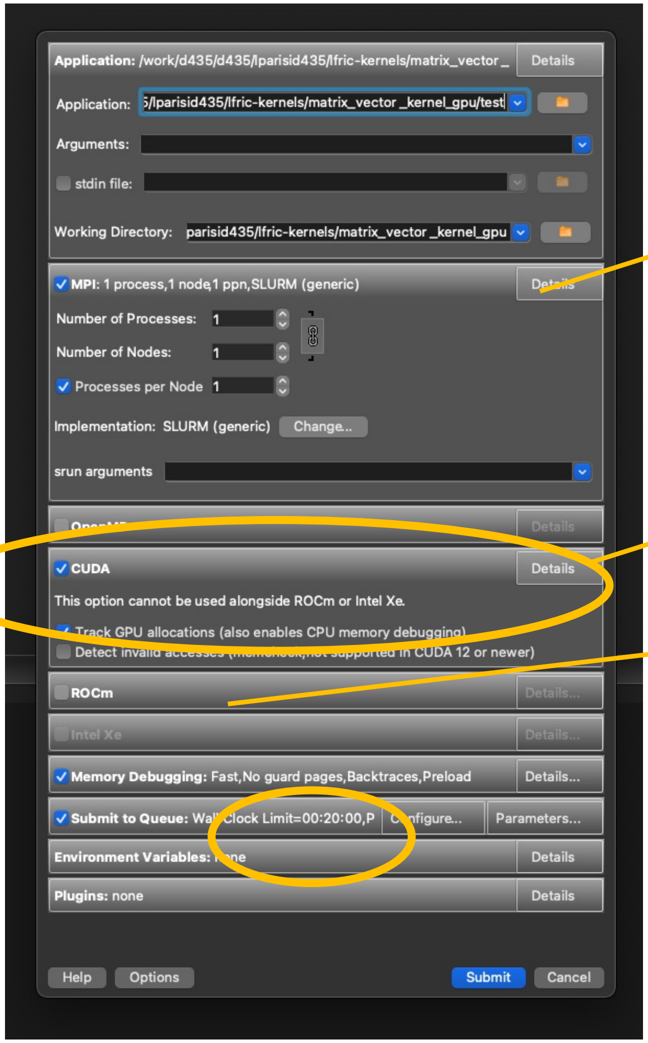
www.archer2.ac.uk

Outline

- Dubbing CUDA with DDT
  - hardware info
  - analyzing a snapshot of the kernel state
- Profiling CUDA with MAP
  - Analise memory transfers
  - Analise warp executions

DDT

DDT



Application: /work/d435/d435/lparisid435/lfric-kernels/matrix_vector_

Application: 5/lparisid435/lfric-kernels/matrix_vector_kernel_gpu/test

Arguments:

stdin file:

Working Directory: parisid435/lfric-kernels/matrix_vector_kernel_gpu

✓ MPI: 1 process,1 node,1 ppn,SLURM (generic)

Number of Processes: 1

Number of Nodes: 1

✓ Processes per Node 1

Implementation: SLURM (generic)

srun arguments

OpenMP

✓ CUDA

This option cannot be used alongside ROCm or Intel Xe.

✓ Track GPU allocations (also enables CPU memory debugging)

Detect invalid accesses (memcheck,not supported in CUDA 12 or newer)

ROCm

Intel Xe

✓ Memory Debugging: Fast,No guard pages,Backtraces,Preload

✓ Submit to Queue: Wall Clock Limit=00:20:00,P

Environment Variables: none

Plugins: none

Run as usual

Select CUDA runtime

Limited support for AMD GPU

Compiling

Before running DDT you need to compile with the appropriate flags.

```
nvcc -g -G kernel.cu
```

Debugging information on the host

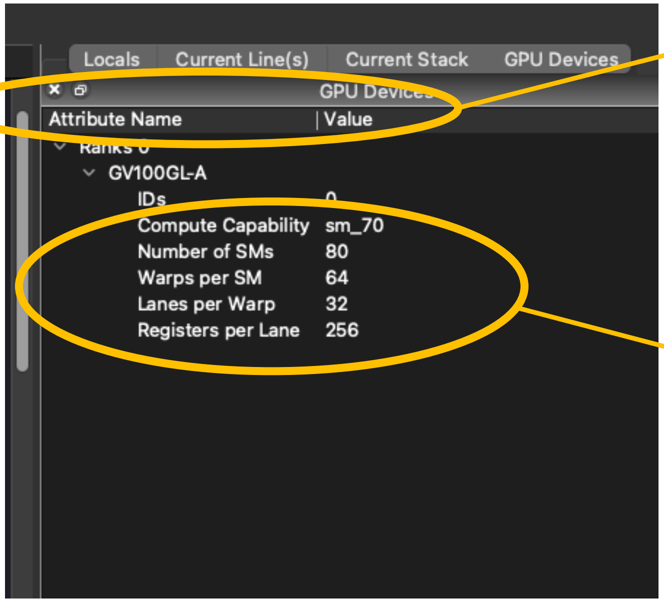
Debugging information on the device

```
nvfortran -O2 -g -gpu=debug kernel.f90
```

Equivalent to `-G` option for NVIDIA C compiler

DDT

Show information about the device



Locals | Current Line(s) | Current Stack | GPU Devices

GPU Devices

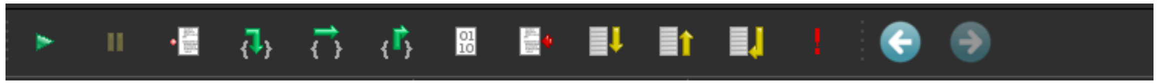
Attribute Name	Value
Ranks 0	
GV100GL-A	
IDs	0
Compute Capability	sm_70
Number of SMs	80
Warps per SM	64
Lanes per Warp	32
Registers per Lane	256

Which rank is using which GPU ?

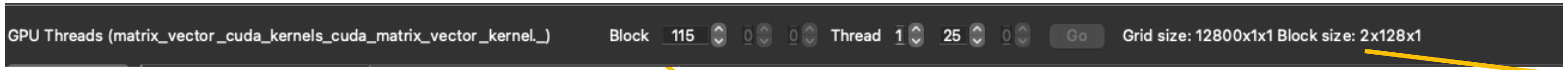
Easy access to hardware information needed for analyzing performance.

80 * 64 * 32 = 1.6384 10^5 threads can be on the device

Stepping trough threads



Advance, break and pause inside a CUDA kernel



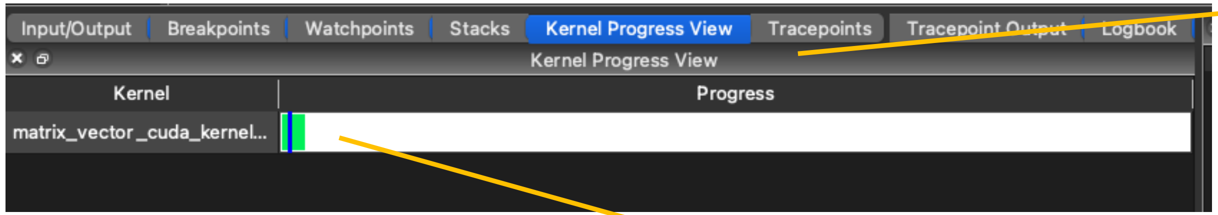
GPU Threads (matrix_vector_cuda_kernels_cuda_matrix_vector_kernel_) Block 115 0 0 Thread 1 25 0 Go Grid size: 12800x1x1 Block size: 2x128x1

When execution is paused switch between threads

Check number of threads launched.

3.2768 10^6 threads,

x20 the capacity of the device



Kernel	Progress
matrix_vector_cuda_kernel...	

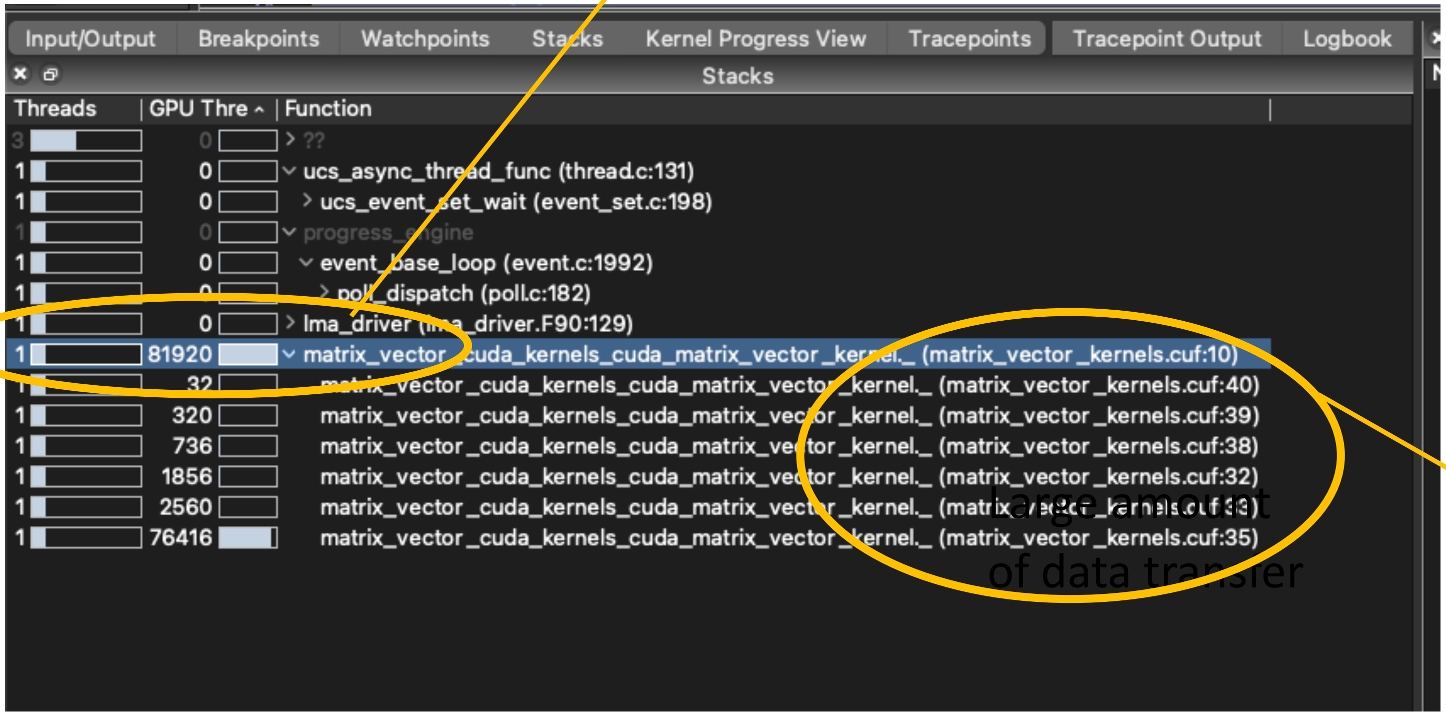
Threads that have already executed or are yet to be executed

Green: Threads Executing on the device

Blue: Selected Thread

DDT

81920 threads are executing. About 50% of the device capacity.



```
Threads | GPU Thre | Function
3          0        > ??
1          0        v ucs_async_thread_func (thread.c:131)
1          0          > ucs_event_set_wait (event_set.c:198)
1          0        v progress_engine
1          0          v event_base_loop (event.c:1992)
1          0            > poll_dispatch (poll.c:182)
1          0        > lma_driver (lma_driver.F90:129)
1      81920        v matrix_vector_cuda_kernels_cuda_matrix_vector_kernel._ (matrix_vector_kernels.cuf:10)
1         32          matrix_vector_cuda_kernels_cuda_matrix_vector_kernel._ (matrix_vector_kernels.cuf:40)
1        320          matrix_vector_cuda_kernels_cuda_matrix_vector_kernel._ (matrix_vector_kernels.cuf:39)
1        736          matrix_vector_cuda_kernels_cuda_matrix_vector_kernel._ (matrix_vector_kernels.cuf:38)
1       1856          matrix_vector_cuda_kernels_cuda_matrix_vector_kernel._ (matrix_vector_kernels.cuf:32)
1       2560          matrix_vector_cuda_kernels_cuda_matrix_vector_kernel._ (matrix_vector_kernels.cuf:33)
1      76416          matrix_vector_cuda_kernels_cuda_matrix_vector_kernel._ (matrix_vector_kernels.cuf:35)
```

Large amount of data transfer

Where are GPU threads ?

MAP

Launching map

Kernel and transfer analysis each add noticeable overhead

Profile the kernel: warp stall information

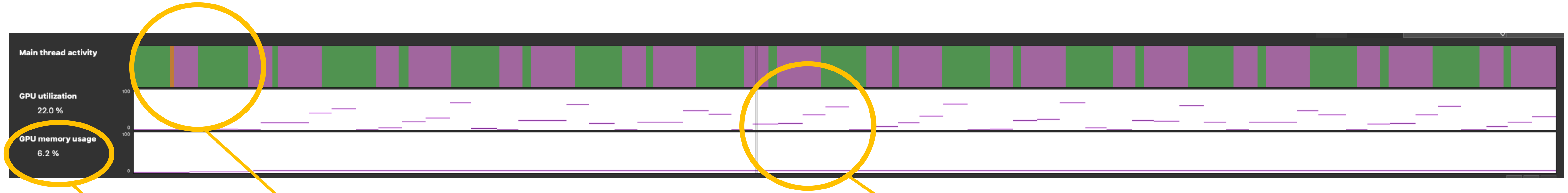
```
map --cuda-kernel-analysis --cuda-transfer-analysis -n 1 --mpi=slurm --
mpiargs="--hint=nomultithread --distribution=block:block --cpus-per-task=1" --
profile ./test
```

Analyze memory transfer

Memory and Kernel analysis are optional. Can add significant overhead

Hotspots

Mix of GPU and CPU computation



Green: computation time

Violet: Time spent waiting on the device

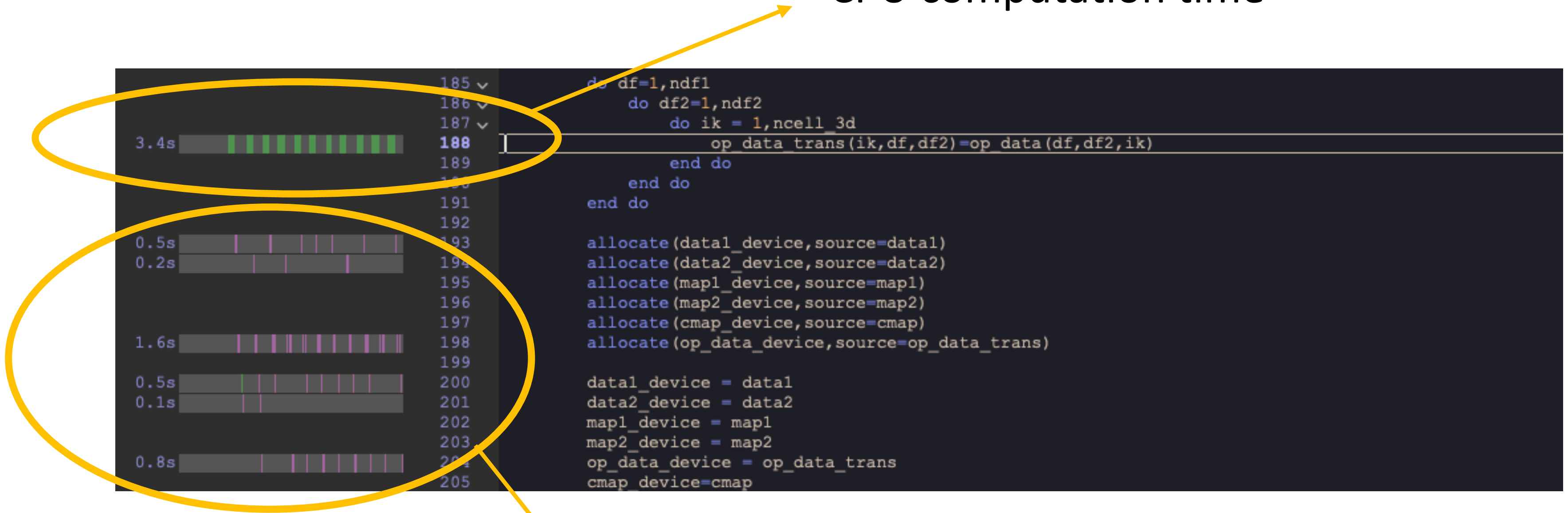
Orange: Time spent in IO

Fraction of the time

spent in GPU utilization. Sensitive to undersampling.

Law fraction of the GPU global memory is being used

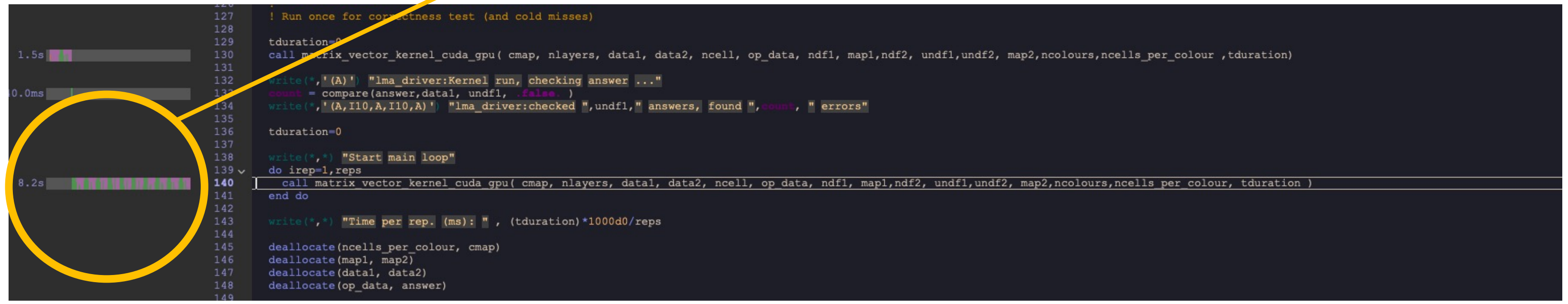
CPU and GPU concurrent analysis



```
do df=1,ndf1
    do df2=1,ndf2
        do ik = 1,ncell_3d
            op_data_trans(ik,df,df2)=op_data(df,df2,ik)
        end do
    end do
end do

allocate(data1_device,source=data1)
allocate(data2_device,source=data2)
allocate(map1_device,source=map1)
allocate(map2_device,source=map2)
allocate(cmap_device,source=cmap)
allocate(op_data_device,source=op_data_trans)

data1_device = data1
data2_device = data2
map1_device = map1
map2_device = map2
op_data_device = op_data_trans
cmap_device=cmap
```

Where is the GPU time ?

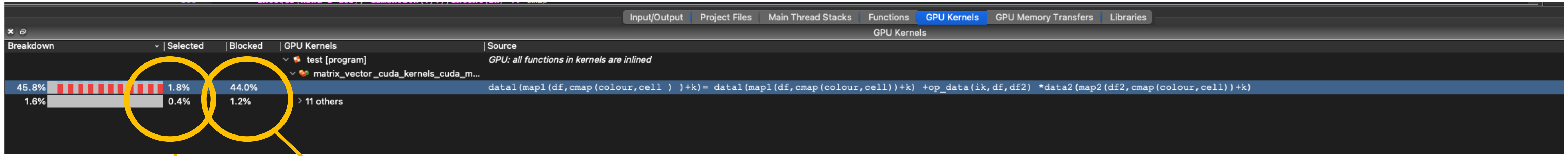
Mix of GPU and CPU computation



```
    ! Run once for correctness test (and cold misses)

    tduration=0
    call matrix_vector_kernel_cuda_gpu( cmap, nlayers, data1, data2, ncell, op_data, ndf1, map1,ndf2, undf1,undf2, map2,ncolours,ncells_per_colour ,tduration)

    write(*,'(A)') "lma_driver:Kernel run, checking answer ..."
    count = compare(answer,data1, undf1, .false. )
    write(*,'(A,I10,A,I10,A)') "lma_driver:checked ",undf1," answers, found ",count, " errors"

    tduration=0

    write(*,*) "Start main loop"
    do irep=1,reps
      call matrix_vector_kernel_cuda_gpu( cmap, nlayers, data1, data2, ncell, op_data, ndf1, map1,ndf2, undf1,undf2, map2,ncolours,ncells_per_colour, tduration )
    end do

    write(*,*) "Time per rep. (ms): " , (tduration)*1000d0/reps

    deallocate(ncells_per_colour, cmap)
    deallocate(map1, map2)
    deallocate(data1, data2)
    deallocate(op_data, answer)
```

Transfers from device after kernel execution

Kernel execution

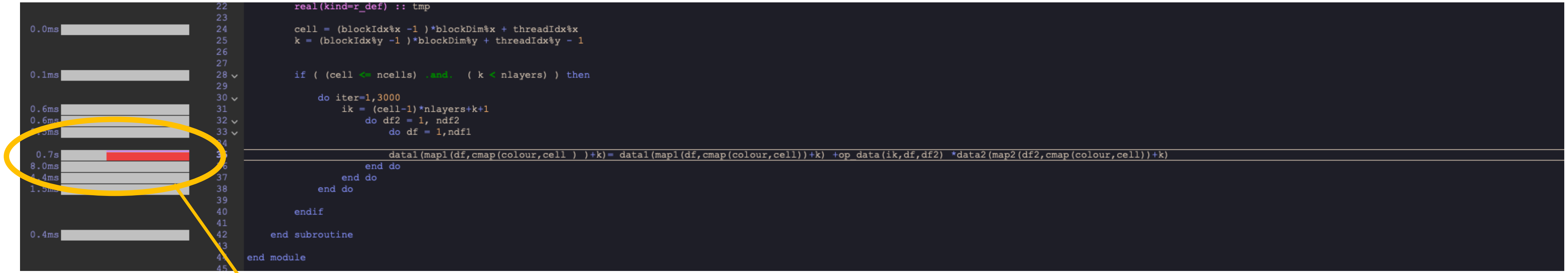


Breakdown	Selected	Blocked	GPU Kernels	Source
			test [program]	*GPU: all functions in kernels are inlined*
			matrix_vector_cuda_kernels_cuda_m...	
45.8%	1.8%	44.0%		`data1(map1(df,cmap(colour,cell ) )+k)= data1(map1(df,cmap(colour,cell))+k) +op_data(ik,df,df2) *data2(map2(df2,cmap(colour,cell))+k)`
1.6%	0.4%	1.2%	11 others	

Fraction of warps executing

Fraction of warps stalled for some reason

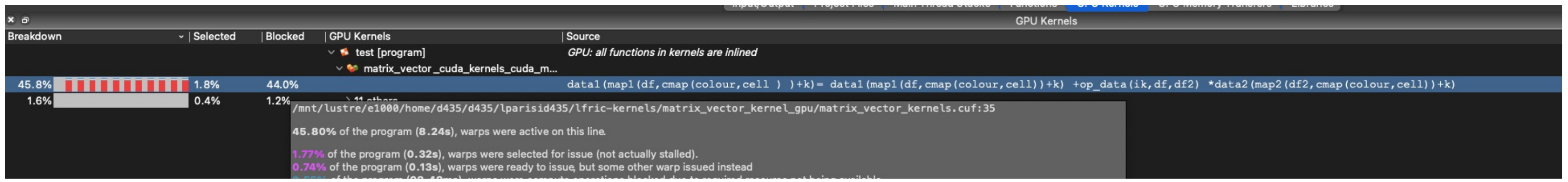
Kernel execution



```
0.0ms
0.1ms
0.6ms
0.6ms
0.7s
8.0ms
1.5ms
0.4ms
```

```
        real(kind=r_def) :: tmp

        cell = (blockIdx%x -1 )*blockDim%x + threadIdx%x
        k = (blockIdx%y -1 )*blockDim%y + threadIdx%y - 1


        if ( (cell <= ncells) .and.  ( k < nlayers) ) then

            do iter=1,3000
                ik = (cell-1)*nlayers+k+1
                    do df2 = 1, ndf2
                        do df = 1,ndf1

                        data1(map1(df,cmap(colour,cell ) )+k)= data1(map1(df,cmap(colour,cell))+k) +op_data(ik,df,df2) *data2(map2(df2,cmap(colour,cell))+k)
                    end do
                end do
            end do

        endif

    end subroutine

end module
```

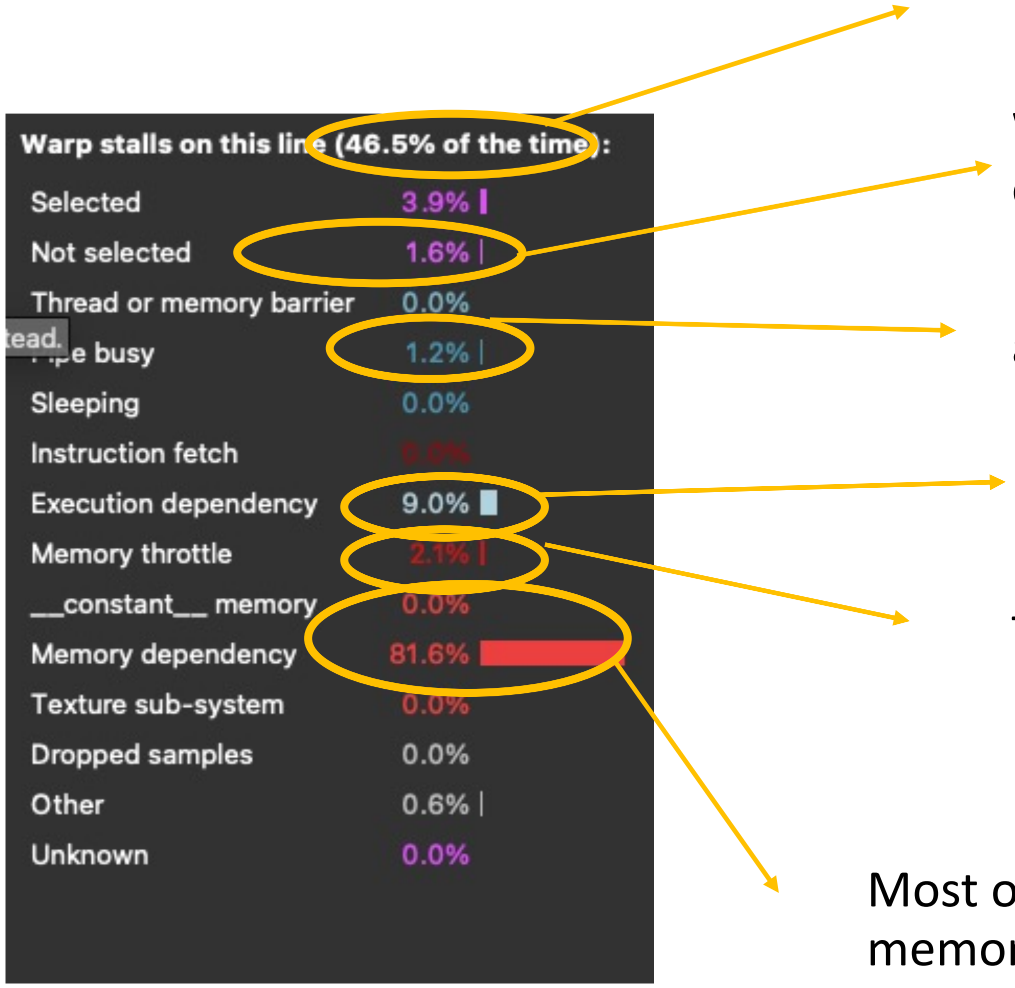
The gpu kernel spent the whole time in a kernel



```
GPU Kernels
Breakdown          | Selected | Blocked | GPU Kernels                      | Source
                                          test [program]                     GPU: all functions in kernels are inlined
                                            matrix_vector_cuda_kernels_cuda_m...
45.8%                1.8%       44.0%                                        data1(map1(df,cmap(colour,cell ) )+k)= data1(map1(df,cmap(colour,cell))+k) +op_data(ik,df,df2) *data2(map2(df2,cmap(colour,cell))+k)
1.6%                 0.4%       1.2%
```

```
/mnt/lustre/e1000/home/d435/d435/lparisid435/lfric-kernels/matrix_vector_kernel_gpu/matrix_vector_kernels.cuf:35

45.80% of the program (8.24s), warps were active on this line.

1.77% of the program (0.32s), warps were selected for issue (not actually stalled).
0.74% of the program (0.13s), warps were ready to issue, but some other warp issued instead
```

Kernel execution


Warp stalls on this line (46.5% of the time):

Selected	3.9%
Not selected	1.6%
Thread or memory barrier	0.0%
Pipe busy	1.2%
Sleeping	0.0%
Instruction fetch	0.0%
Execution dependency	9.0%
Memory throttle	2.1%
__constant__ memory	0.0%
Memory dependency	81.6%
Texture sub-system	0.0%
Dropped samples	0.0%
Other	0.6%
Unknown	0.0%

Warps ready for computation, but not executing as not enough resources available

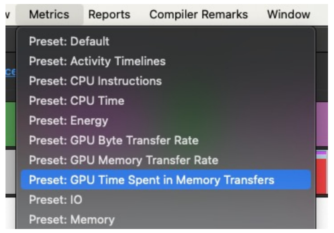
Being executed, but stalling as functional units are not available

Waiting for the result of an arithmetic operation

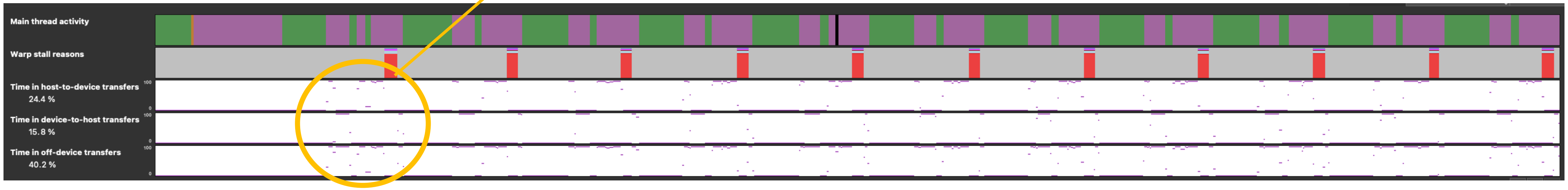
Too many memory requests

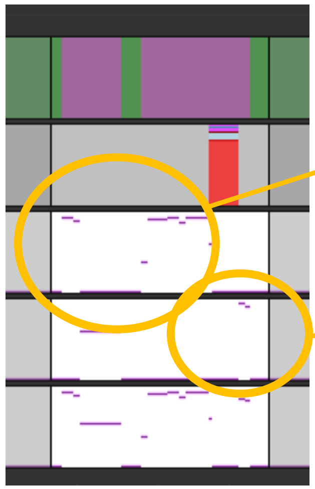
Most of the warps waiting for data to be loaded from memory

Memory transfers


Metrics
Reports
Compiler Remarks
Window
Preset: Default
Preset: Activity Timelines
Preset: CPU Instructions
Preset: CPU Time
Preset: Energy
Preset: GPU Byte Transfer Rate
Preset: GPU Memory Transfer Rate
Preset: GPU Time Spent in Memory Transfers
Preset: IO
Preset: Memory

Many repeated transfers

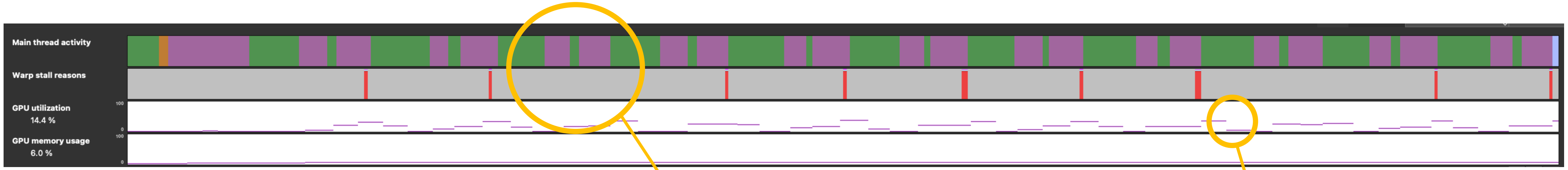

Main thread activity
Warp stall reasons
Time in host-to-device transfers
24.4 %
Time in device-to-host transfers
15.8 %
Time in off-device transfers
40.2 %
100
0



Transfers to device before kernel execution

Transfers from device after kernel execution

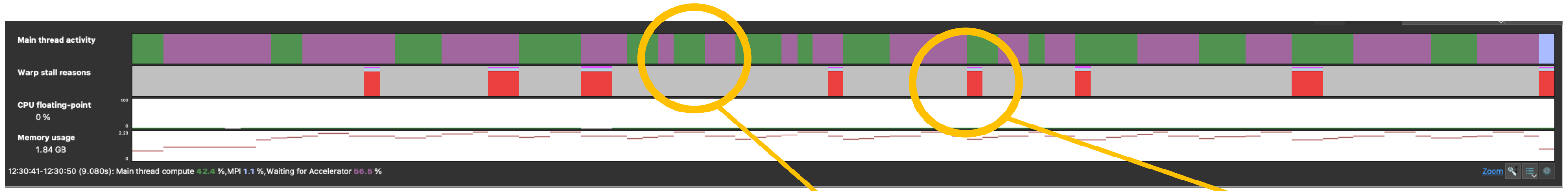
Sampling



Missing information about warp stall reasons

GPU usage in unexpected areas of the code

Sampling



12:30:41-12:30:50 (9.080s): Main thread compute 42.4 %,MPI 1.1 %,Waiting for Accelerator 56.5 %

Kernels in wrong position

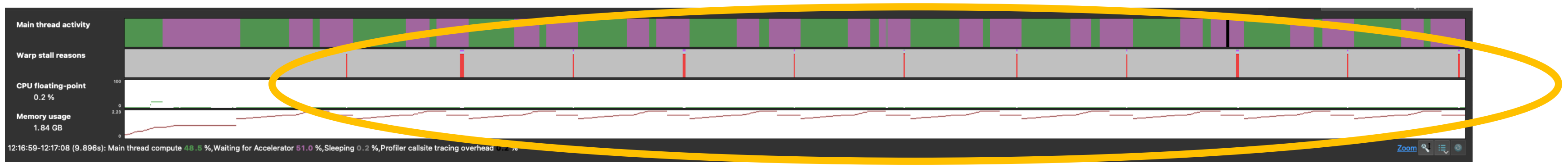
Kernel is missing. Time intervals are fluctuating wildly

Inconsistencies and blocky appearance suggests we are not collecting enough samples

Sampling

```
export FORGE_SAMPLER_INTERVAL=1
export FORGE_SAMPLER_GPU_INTERVAL=1
```

Set sampling interval to 1ms, the maximum supported value

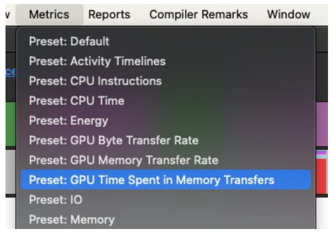


12:16:59-12:17:08 (9.896s): Main thread compute 48.5 %,Waiting for Accelerator 51.0 %,Sleeping 0.2 %,Profiler callsite tracing overhead 0.2 %

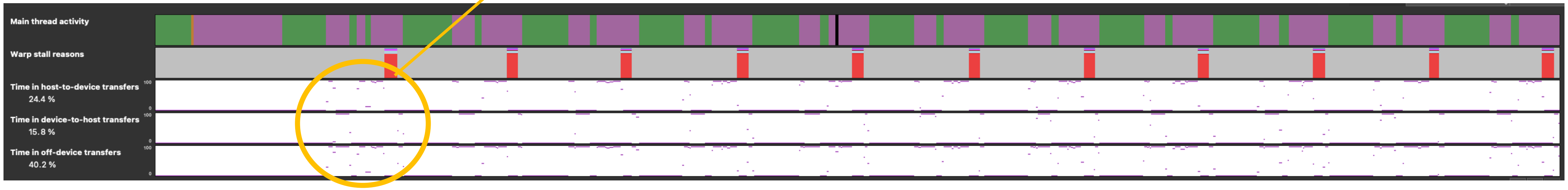
Kernels all presents and in the expected position.

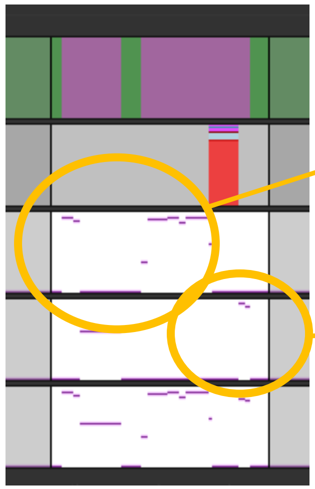
Balance between amount of information and sampler overhead

Memory transfers


Metrics
Reports
Compiler Remarks
Window
Preset: Default
Preset: Activity Timelines
Preset: CPU Instructions
Preset: CPU Time
Preset: Energy
Preset: GPU Byte Transfer Rate
Preset: GPU Memory Transfer Rate
Preset: GPU Time Spent in Memory Transfers
Preset: IO
Preset: Memory

Many repeated transfers

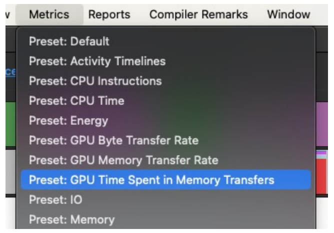

Main thread activity
Warp stall reasons
Time in host-to-device transfers
24.4 %
Time in device-to-host transfers
15.8 %
Time in off-device transfers
40.2 %



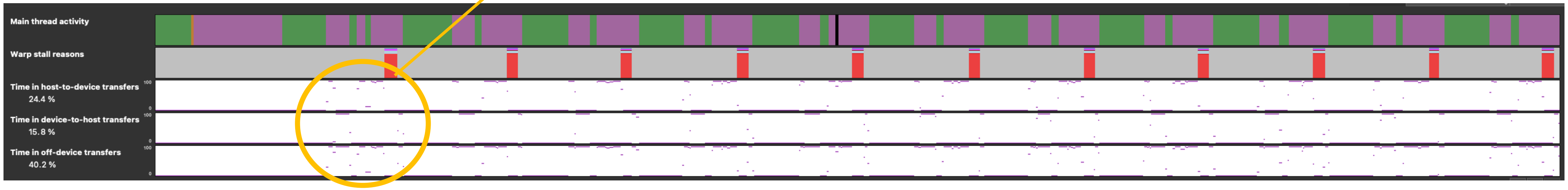
Transfers to device before kernel execution

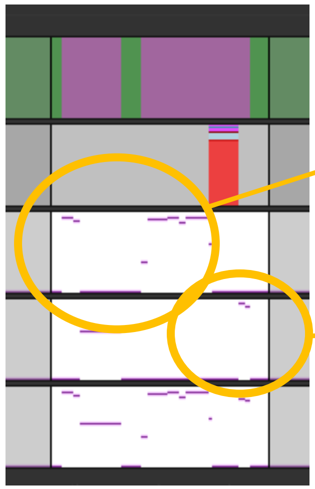
Transfers from device after kernel execution

Memory transfers


Metrics
Reports
Compiler Remarks
Window
Preset: Default
Preset: Activity Timelines
Preset: CPU Instructions
Preset: CPU Time
Preset: Energy
Preset: GPU Byte Transfer Rate
Preset: GPU Memory Transfer Rate
Preset: GPU Time Spent in Memory Transfers
Preset: IO
Preset: Memory

Many repeated transfers

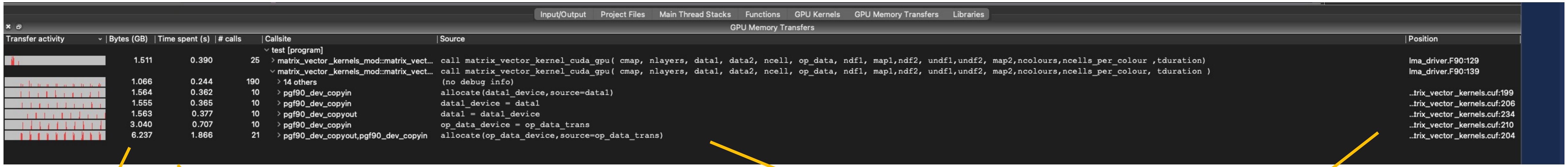

Main thread activity
Warp stall reasons
Time in host-to-device transfers
24.4 %
Time in device-to-host transfers
15.8 %
Time in off-device transfers
40.2 %
100
0



Transfers to device before kernel execution

Transfers from device after kernel execution

Memory transfers



Transfer activity	Bytes (GB)	Time spent (s)	# calls	Callsite	Source	Position
				test [program]		
	1.511	0.390	25	matrix_vector_kernels_mod::matrix_vect...	call matrix_vector_kernel_cuda_gpu(cmap, nlayers, data1, data2, ncell, op_data, ndf1, map1,ndf2, undf1,undf2, map2,ncolours,ncells_per_colour ,tduration)	lma_driver.F90:129
				matrix_vector_kernels_mod::matrix_vect...	call matrix_vector_kernel_cuda_gpu(cmap, nlayers, data1, data2, ncell, op_data, ndf1, map1,ndf2, undf1,undf2, map2,ncolours,ncells_per_colour, tduration)	lma_driver.F90:139
	1.066	0.244	190	14 others	(no debug info)	
	1.564	0.362	10	pgf90_dev_copyin	allocate(data1_device,source=data1)	..trix_vector_kernels.cuf:199
	1.555	0.365	10	pgf90_dev_copyin	data1_device = data1	..trix_vector_kernels.cuf:206
	1.563	0.377	10	pgf90_dev_copyout	data1 = data1_device	..trix_vector_kernels.cuf:234
	3.040	0.707	10	pgf90_dev_copyin	op_data_device = op_data_trans	..trix_vector_kernels.cuf:210
	6.237	1.866	21	pgf90_dev_copyout,pgf90_dev_copyin	allocate(op_data_device,source=op_data_trans)	..trix_vector_kernels.cuf:204

Large amount of data transfer

Sizable amount of time spent in GPU transfers

Source code responsible for the transfer

Conclusions

- Debug both CPU and GPU with DDT
  - Step into both GPU kernels and CPU kernels
- Profile both CPU and GPU with MAP
  - Time spent waiting on device
  - Memory transfers
  - Stalling reasons