Score-P Hands-On

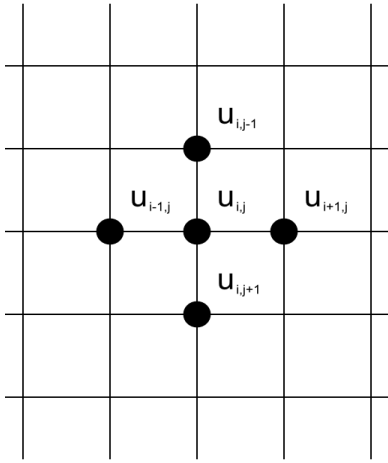
CUDA: Jacobi example

- Jacobi Example
  - Iterative solver for system of equations

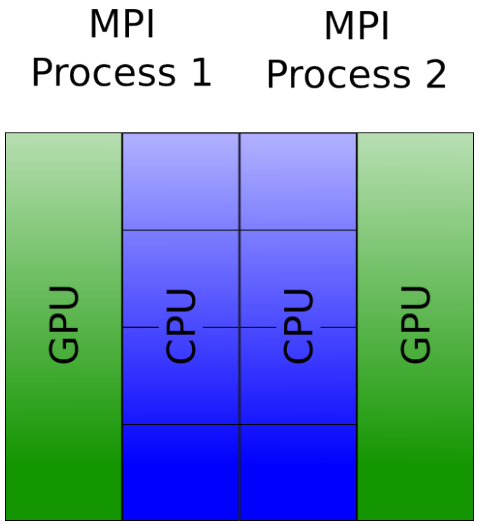
$$U_{old} = U$$

$$u_{i,j} = bu_{old,i,j} + a_x(u_{old,i-1,j} + u_{old,i+1,j}) + a_y(u_{old,i,j-1} + u_{old,i,j+1}) - rHs / b$$

  - Code uses OpenMP, CUDA and MPI for parallelization



- Domain decomposition
  - Halo exchange at boundaries:
    - Via MPI between processes
    - Via CUDA between hosts and accelerators



Jacobi Without Instrumentation

```
# Compile host code
% mpicc -O3 -fopenmp -DUSE_MPI -I<path_to_cuda_header>
                -c jacobi_cuda.c -o jacobi_mpi+cuda.o

# Compile CUDA kernel
% nvcc -O3 -c jacobi_cuda_kernel.cu
                -o jacobi_cuda_kernel.o

# Link executable
% mpicc -fopenmp -lm -L<path_tocuda_libs> -lcudart
                jacobi_mpi+cuda.o jacobi_cuda_kernel.o -o ./jacobi_mpi+cuda
```

- Switch to Score-P installation including CUDA support

```
% module switch scorep scorep/1.2.3-intel13-impi-cuda50
```

- Load corresponding CUDA module

```
% module load cuda/5.0
```

Instrumentation with Score-P

```
# Compile host code
% scorep mpicc -O3 -fopenmp -DUSE_MPI -I<path_to_cuda_header>
                  -c jacobi_cuda.c -o jacobi_mpi+cuda.o

# Compile CUDA kernel
% scorep nvcc -O3 -c jacobi_cuda_kernel.cu
                  -o jacobi_cuda_kernel.o

# Link executable
% scorep mpicc -fopenmp -lm -L<path_tocuda_libs> -lcudart
                  jacobi_mpi+cuda.o jacobi_cuda_kernel.o -o ./jacobi_mpi+cuda
```

CUDA Advanced Measurement Configuration

- Enable recording of CUDA events with the CUPTI interface via environment variable
  SCOREP_CUDA_ENABLE

- Provide a list of recording types, e.g.

```
% export SCOREP_CUDA_ENABLE=runtime,driver,gpu,kernel,idle
```

- Start with using the default configuration

```
% export SCOREP_CUDA_ENABLE=yes
```

- Adjust CUPTI buffer size (in bytes) as needed

```
% export SCOREP_CUDA_BUFFER=100000
```

SCOREP_CUDA_ENABLE: Recording Types

Recording type	Remark
yes/DEFAULT/1	"runtime, kernel, concurrent, memcpy"
no	Disable CUDA measurement (same as unset SCOREP_CUDA_ENABLE)
runtime	CUDA runtime API
driver	CUDA driver API
kernel	CUDA kernels
kernel_counter	Fixed CUDA kernel metrics
concurrent	Concurrent kernel recording
idle	GPU compute idle time
pure_idle	GPU idle time (memory copies are not idle)
memcpy	CUDA memory copies
sync	Record implicit and explicit CUDA synchronization
gpumemusage	Record CUDA memory (de)allocations as a counter
stream_reuse	Reuse destroyed/closed CUDA streams
device_reuse	Reuse destroyed/closed CUDA devices

Measurement (Profiling)

```
% export OMP_NUM_THREADS=6
% export SCOREP_CUDA_ENABLE=yes
% export SCOREP_CUDA_BUFFER=500000
% export SCOREP_EXPERIMENT_DIRECTORY=jacobi_cuda_profile

% mpirun -n 2 ./jacobi_mpi+cuda 4096 4096 0.15

Jacobi relaxation Calculation: 4096 x 4096 mesh with
 2 processes and 6 threads / one Tesla T10 Processor for each process.
 307 of 2049 local rows are calculated on the CPU to balance the load
 between the CPU and the GPU.
    0, 0.113429
  … … … … … …
  900, 0.000101
 total: 12.83581
```

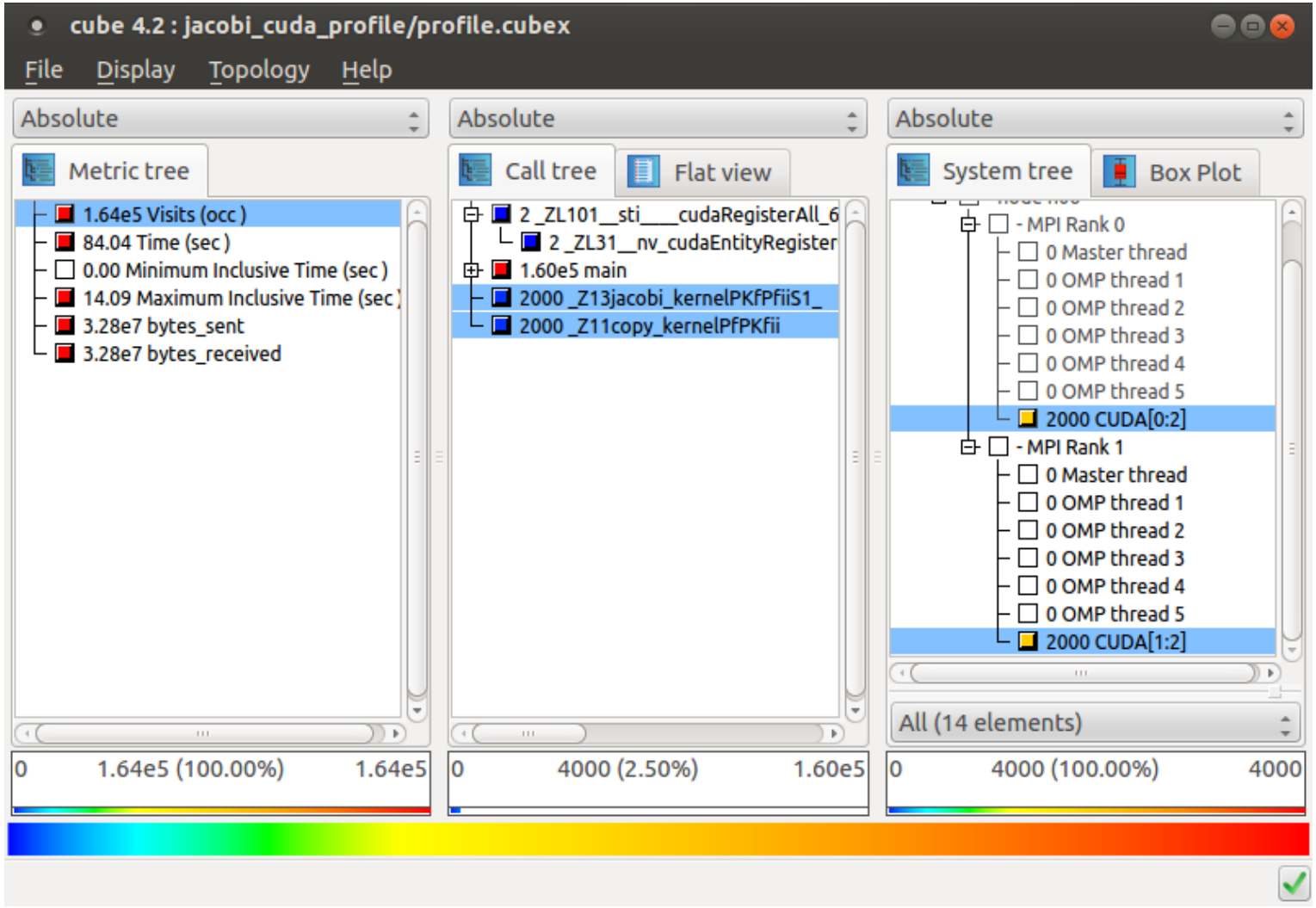
Problem size (x dimension)

Problem size (y dimension)

Load balancing factor (in this example 15% of the computations are calculated on the CPU)

CUBE4 Analysis

```
% cube jacobi_cuda_profile/profile.cubex
```



- Do we need to filter? (Overhead and memory footprint)

```
% scorep-score jacobi_cuda_profile/profile.cubex
Estimated aggregate size of event trace (total_tbc):       3.875.472 bytes
Estimated requirements for largest trace buffer (max_tbc): 1.937.936 bytes
(hint: When tracing set SCOREP_TOTAL_MEMORY > max_tbc to avoid
       intermediate flushes or reduce requirements using file listing
       names of USR regions to be filtered.)

flt type         max_tbc         time      % region
     ALL         1937936        24.97  100.0 ALL
     OMP         1154110        18.78   75.2 OMP
     USR          667480         5.95   23.8 USR
     MPI          116192         0.14    0.5 MPI
     COM             154         0.10    0.4 COM
```

☞ Very small example => no filtering

Measurement (Tracing)

```
% export OMP_NUM_THREADS=6
% export SCOREP_CUDA_ENABLE=yes
% export SCOREP_CUDA_BUFFER=500000
% export SCOREP_EXPERIMENT_DIRECTORY=jacobi_cuda_trace
% export SCOREP_ENABLE_PROFILING=false
% export SCOREP_ENABLE_TRACING=true

% mpirun -n 2 ./jacobi_mpi+cuda 4096 4096 0.15

Jacobi relaxation Calculation: 4096 x 4096 mesh with
 2 processes and 6 threads + one Tesla T10 Processor for each process.
 307 of 2049 local rows are calculated on the CPU to balance the load
 between the CPU and the GPU.
    0, 0.113429
  … … … … … …
  900, 0.000101
 total: 12.875220 s
```

Vampir Analysis

```
% vampir jacobi_cuda_trace/traces.otf2
```