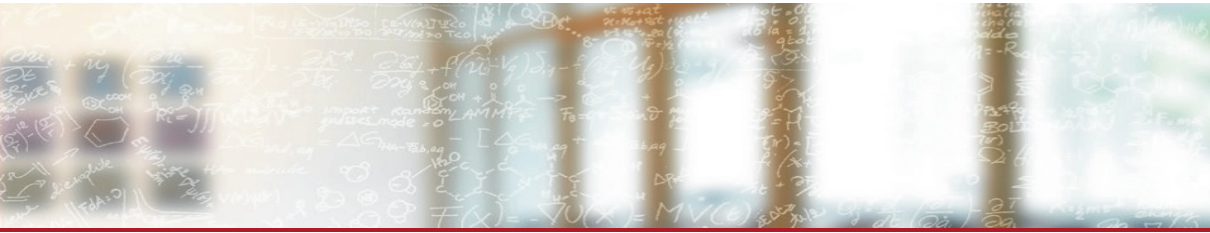


Profiling and Debugging

Directive Based GPU programming course 2018

Vasileios Karakasis, CSCS

May 14–15, 2018

Overview

- Why and where my code crashes?
- Why my code does not perform as “expected”?

Debugging

OpenACC translates to CUDA code, so you may use the corresponding tools:

- `cuda-memcheck`: Check for memory erros and race conditions
- `cuda-gdb`: Debug the generated CUDA kernels
- `nvprof+nvvp`: Detailed performance profiling

Other CUDA-aware tools:

- Allinea DDT: Debug MPI, CUDA, OpenMP applications + memory checking

Compiler-related diagnostics:

- Code generation diagnostics
- PGI debugger (`pgdb`)
- PGI profiler (`pgprof`)
- CrayPAT profiler

Debugging

cuda-memcheck

```
$ srun -n1 -Cgpu cuda-memcheck ./blur.openacc

========= Invalid __global__ read of size 8
=========     at 0x00000098 in blur_twice_gpu_nocopies_84_gpu(double*, double*, int, int)
=========     by thread (66,0,0) in block (8192,0,0)
=========     Address 0x10253e00210 is out of bounds
=========     Saved host backtrace up to driver entry point at kernel launch time
=========     Host Frame:/opt/cray/nvidia/default/lib64/libcuda.so (cuLaunchKernel + 0x2cd) [0x23ce3d]
=========     Host Frame:/apps/common/UES/pgi/18.4/linux86-64/18.4/lib/libaccn.so (__pgi_uacc_cuda_launch3 + 0x1
=========     Host Frame:/apps/common/UES/pgi/18.4/linux86-64/18.4/lib/libaccn.so [0x17950]
=========     Host Frame:/apps/common/UES/pgi/18.4/linux86-64/18.4/lib/libaccn.so (__pgi_uacc_cuda_launch + 0x10
=========     Host Frame:/apps/common/UES/pgi/18.4/linux86-64/18.4/lib/libaccg.so (__pgi_uacc_launch + 0x1ac) [0
=========     Host Frame:./blur.openacc [0x52a5]
=========     Host Frame:./blur.openacc [0x57df]
=========     Host Frame:/lib64/libc.so.6 (__libc_start_main + 0xf5) [0x206e5]
=========     Host Frame:./blur.openacc [0x3489]
```

Debugging

cuda-gdb

Compile with `-g -Mcuda=debug`

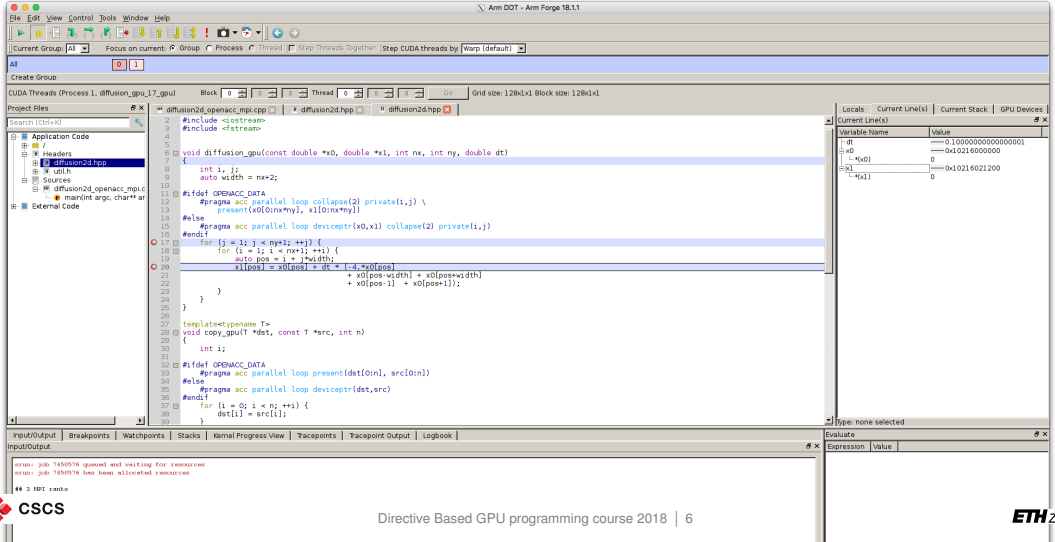
```
$ srun -n1 -u -Cgpu cuda-gdb ./blur.openacc

(cuda-gdb) b blur_openacc.cpp:86
Breakpoint 1 at 0x40542b: file ./blur_openacc.cpp, line 86.
(cuda-gdb) r
Starting program: /users/karakasv/Devel/openacc-training/solutions/shared/./blur.openacc
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib64/libthread_db.so.1".
dispersion 1D test of length n = 1048580 : 8.00003MB
[New Thread 0x2aaab52b2700 (LWP 17350)]
[New Thread 0x2aaab54b3700 (LWP 17351)]
[Switching focus to CUDA kernel 0, grid 3, block (1,0,0), thread (0,0,0), device 0, sm 2, warp 0, lane 0]

Breakpoint 1, blur_twice_gpu_nocopies_84_gpu<<<(8193,1,1),(128,1,1)>>> (
    out=0x10253200000, in=0x10252800000) at blur_openacc.cpp:87
87              for (auto i = 0; i < n; ++i) {
```

Debugging

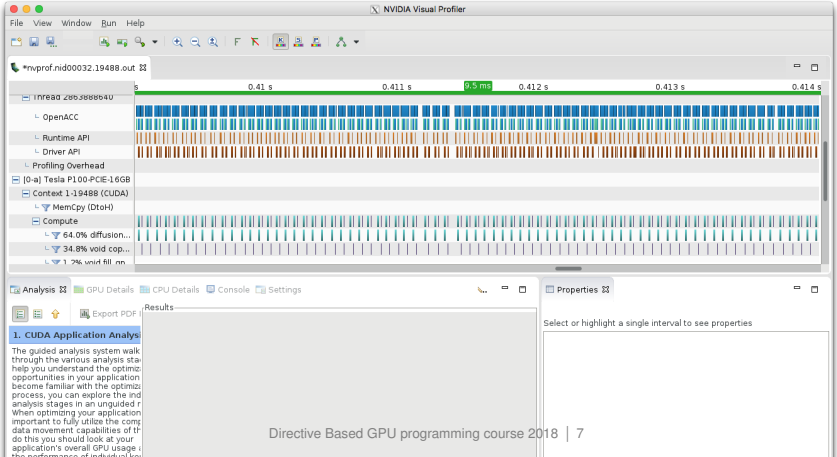
Using DDT



Profiling

Using `nvprof` & `nvvp`

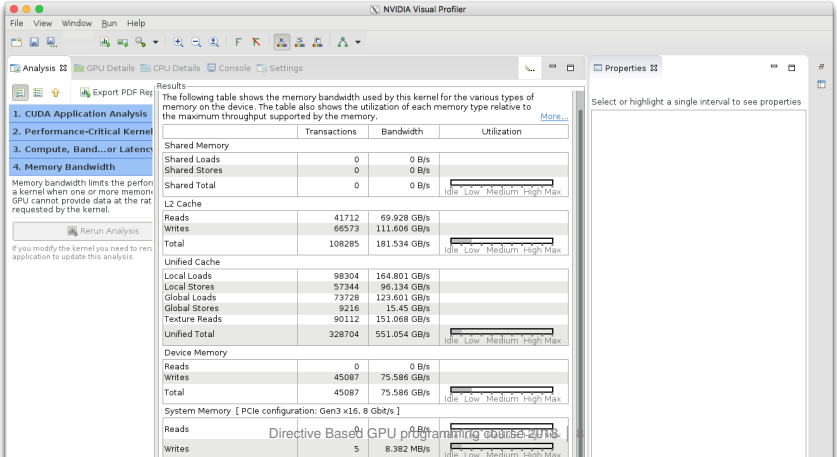
```
srun -N2 -Cgpu nvprof -o nvprof.%h.%p.out
./diffusion2d.openacc.mpi
```



Profiling

Using `nvprof` & `nvvp` – Detailed analysis

```
srun -N2 -Cgpu nvprof --analysis-metrics -o nvprof.%h.%p.out
./diffusion2d.openacc.mpi
```



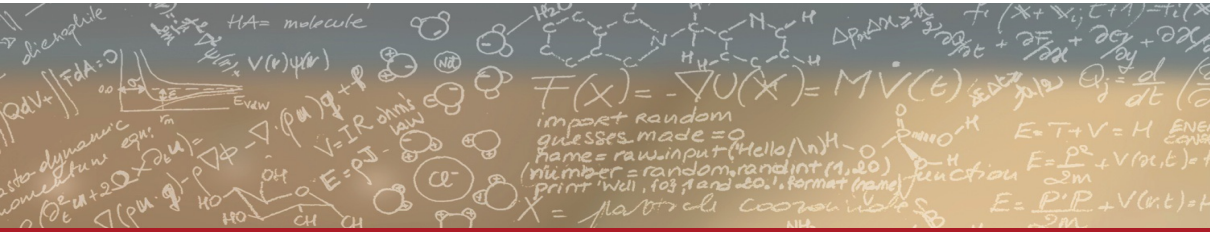
Compiler diagnostics

GPU kernel information

- Compile with special flags (`-Minfo=accel` for PGI, `-hmsgs` for Cray)

```
diffusion_gpu(const double *, double *, int, int, double):
      6, include "diffusion2d.hpp"
          17, Generating present(x0[:nx*ny],x1[:nx*ny])
              Accelerator kernel generated
              Generating Tesla code
              17, #pragma acc loop gang, vector(128) collapse(2) /* blockIdx.x threadIdx.x */
              18,   /* blockIdx.x threadIdx.x collapsed */
main:
     72, Generating create(x0[:buffer_size])
         Generating copyout(x1[:buffer_size])
void fill_gpu<double>(T1 *, T1, int):
      6, include "diffusion2d.hpp"
          53, Generating present(v[:n])
              Accelerator kernel generated
              Generating Tesla code
              53, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
```

More during the hands-on!



Thank you for your attention