LUMI-G Pre-Hackathon

Note: Reservation: small_g (--reservation=small_g)`

Rocprof

- Get the exercise:

https://github.com/ROCm-Developer-Tools/HIP-Examples/tree/master/mini-nbody (https://github.com/ROCm-Developer-Tools/HIP-Examples/tree/master/mini-nbody)

- Compile and run the code

```
cd mini-nbody/hip
```

Can compile and run all with

```
./HIP-nbody-orig.sh
```

Or just run compile and run one case

```
hipcc --offload-arch=gfx90a -I../ -DSHMOO nbody-orig.cpp -o nbody-orig
./nbody-orig 65536
65536, 161.871
```

- Note that we can find the compile line and the executable name by checking the file `HIP-nbody-orig.sh` or by watching the output from running the script.

```
cat HIP-nbody-orig.sh

...

EXE=nbody-orig
...
./$EXE 65536
...
```

- The binary is called `nbody-orig`

- Use rocprof with `--stats`

```
rocprof --stats nbody-orig 65536

RPL: on '221130_200946' from '/global/software/rocm/rocm-5.3.0' in '/global/hom
RPL: profiling '"nbody-orig" "65536"'
RPL: input file ''
RPL: output dir '/tmp/rpl_data_221130_200946_3670592'
RPL: result dir '/tmp/rpl_data_221130_200946_3670592/input_results_221130_20094
ROCProfiler: input from "/tmp/rpl_data_221130_200946_3670592/input.xml"
  0 metrics
65536, 159.960

ROCPRofiler: 10 contexts collected, output directory /tmp/rpl_data_221130_20094
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.csv' is generatin
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.stats.csv' is gen
```

Files with the prefix results are created

- Check the files results.csv

You can see information for each kernel call with their duration

```
 cat results.csv

"Index","KernelName","gpu-id","queue-id","queue-index","pid","tid","grd","wgr",
0,"bodyForce(Body*, float, int) [clone .kd]",0,0,0,3670615,3670615,65536,256,0,
1,"bodyForce(Body*, float, int) [clone .kd]",0,0,2,3670615,3670615,65536,256,0,
2,"bodyForce(Body*, float, int) [clone .kd]",0,0,4,3670615,3670615,65536,256,0,
3,"bodyForce(Body*, float, int) [clone .kd]",0,0,6,3670615,3670615,65536,256,0,
4,"bodyForce(Body*, float, int) [clone .kd]",0,0,8,3670615,3670615,65536,256,0,
5,"bodyForce(Body*, float, int) [clone .kd]",0,0,10,3670615,3670615,65536,256,0
6,"bodyForce(Body*, float, int) [clone .kd]",0,0,12,3670615,3670615,65536,256,0
7,"bodyForce(Body*, float, int) [clone .kd]",0,0,14,3670615,3670615,65536,256,0
8,"bodyForce(Body*, float, int) [clone .kd]",0,0,16,3670615,3670615,65536,256,0
9,"bodyForce(Body*, float, int) [clone .kd]",0,0,18,3670615,3670615,65536,256,0
```

- Check the statistics result file, one line per kernel

```
cat results.stats.csv

"Name","Calls","TotalDurationNs","AverageNs","Percentage"
"bodyForce(Body*, float, int) [clone .kd]",10,261124944,26112494,100.0
```

- Profile the HIP calls with `--hip-trace`

```
rocprof --stats --hip-trace nbody-orig 65536
RPL: on '221130_201416' from '/global/software/rocm/rocm-5.3.0' in '/global/hom
RPL: profiling '"nbody-orig" "65536"'
RPL: input file ''
RPL: output dir '/tmp/rpl_data_221130_201416_3670892'
RPL: result dir '/tmp/rpl_data_221130_201416_3670892/input_results_221130_20141
ROCTracer (pid=3670915):
    HIP-trace()
65536, 161.051
hsa_copy_deps: 0
scan ops data 29:30
dump json 19:20
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.json' is generati
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.hip_stats.csv' is
dump json 51:52
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.json' is generati
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.stats.csv' is gen
dump json 9:10
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.json' is generati
```

Now we have new files with the `hip` in their name like below, check the file `results.hip_stats.csv`

```
  cat results.hip_stats.csv
"Name","Calls","TotalDurationNs","AverageNs","Percentage"
"hipMemcpy",20,486845521,24342276,99.89375113830629
"hipLaunchKernel",10,467008,46700,0.09582337501179998
"hipMalloc",1,30570,30570,0.006272527610042495
"hipFree",1,14210,14210,0.0029156891507590398
"__hipPushCallConfiguration",10,3510,351,0.0007202018943817191
"__hipPopCallConfiguration",10,2520,252,0.0005170680267355932
```

- Profile also the HSA API with the `--hsa-trace`

```
rocprof --stats --hip-trace --hsa-trace nbody-orig 65536
RPL: on '221130_201737' from '/global/software/rocm/rocm-5.3.0' in '/global/hom
RPL: profiling '"nbody-orig" "65536"'
RPL: input file ''
RPL: output dir '/tmp/rpl_data_221130_201737_3671219'
RPL: result dir '/tmp/rpl_data_221130_201737_3671219/input_results_221130_20173
ROCProfiler: input from "/tmp/rpl_data_221130_201737_3671219/input.xml"
  0 metrics
ROCTracer (pid=3671242):
    HSA-trace()
    HSA-activity-trace()
    HIP-trace()
65536, 155.978

ROCPRofiler: 10 contexts collected, output directory /tmp/rpl_data_221130_20173
hsa_copy_deps: 1
scan hsa API data 5953:5954
scan hip API data 51:52
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.stats.csv' is gen
dump json 9:10
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.json' is generati
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.hsa_stats.csv' is
dump json 5963:5964
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.json' is generati
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.copy_stats.csv' i
dump json 19:20
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.json' is generati
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.hip_stats.csv' is
dump json 51:52
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/results.json' is generati
```

- See the content of the file `results.hsa_stats.csv`

```
cat results.hsa_stats.csv
"Name","Calls","TotalDurationNs","AverageNs","Percentage"
"hsa_signal_wait_scacquire",50,264955082,5299101,82.69977799679005
"hsa_queue_create",1,39868279,39868279,12.443987854568068
"hsa_amd_memory_async_copy",20,4917141,245857,1.5347751249357586
"hsa_amd_signal_async_handler",20,4262555,213127,1.3304608069344652
"hsa_signal_store_screlease",40,1945998,48649,0.60739956889069
"hsa_amd_memory_lock_to_pool",20,1418202,70910,0.44265990170591873
"hsa_amd_memory_unlock",20,723957,36197,0.22596691758953363
"hsa_agent_get_info",80,671007,8387,0.20943976433821374
"hsa_amd_memory_pool_allocate",5,597926,119585,0.18662917157599068
"hsa_system_get_info",5005,367139,73,0.11459419296574767
"hsa_executable_load_agent_code_object",2,216629,108314,0.0676158768966984
"hsa_executable_freeze",2,156380,78190,0.048810504729771616
"hsa_amd_agents_allow_access",4,89470,22367,0.02792605101785821
"hsa_signal_create",57,51500,903,0.016074568318092074
"hsa_code_object_reader_create_from_memory",2,20940,10470,0.006535950690890253
"hsa_isa_get_info_alt",2,18180,9090,0.005674478680056581
"hsa_signal_load_relaxed",236,17880,75,0.005580840418009442
"hsa_system_get_major_extension_table",3,14920,4973,0.004656942899144344
"hsa_amd_profiling_get_async_copy_time",40,13430,335,0.004191872864310224
"hsa_executable_create_alt",2,9030,4515,0.0028185116876188626
"hsa_amd_memory_pool_get_info",106,7570,71,0.002362805478989456
"hsa_amd_agent_iterate_memory_pools",27,7440,275,0.0023222288987690297
"hsa_amd_memory_pool_free",1,4710,4710,0.0014701207141400712
"hsa_executable_get_symbol_by_name",15,3599,239,0.0011233470170255023
"hsa_queue_add_write_index_screlease",20,3500,175,0.0010924463905499467
"hsa_amd_profiling_get_dispatch_time",20,3110,155,0.0009707166498886669
"hsa_signal_silent_store_relaxed",40,2800,70,0.0008739571124399574
"hsa_amd_agent_memory_pool_get_info",19,2490,131,0.0007771975749912477
"hsa_iterate_agents",2,2250,1125,0.0007022869653535371
"hsa_queue_load_read_index_relaxed",20,2240,112,0.0006991656899519659
"hsa_signal_destroy",20,2190,109,0.0006835593129441095
"hsa_queue_load_read_index_scacquire",20,1790,89,0.0005587082968812585
"hsa_executable_symbol_get_info",30,1540,51,0.00048067641184197657
"hsa_amd_profiling_async_copy_enable",1,440,440,0.00013733611766913615
"hsa_agent_iterate_isas",1,400,400,0.00012485101606285104
"hsa_amd_profiling_set_profiler_enabled",1,140,140,4.3697855621997866e-05
"hsa_dispatch",10,0,0,0.0
```

- Download the results.json file on your laptop

From your laptop:

```
scp username@lumi.csc.fi:/path/results.json .
```

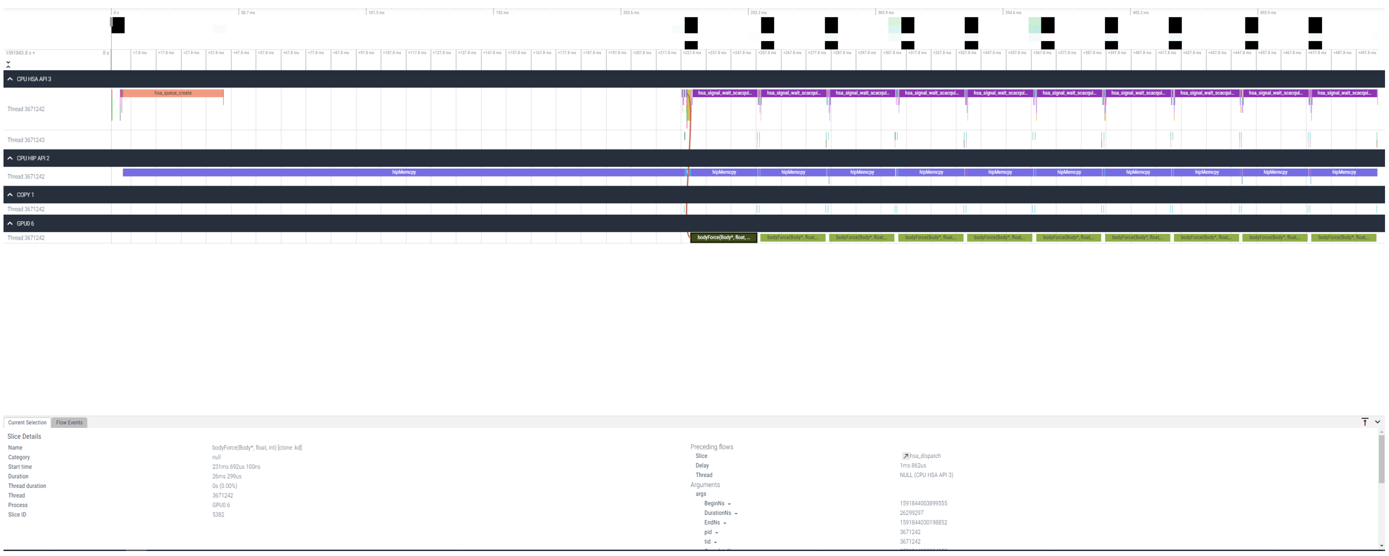
- Visit the web page:

https://ui.perfetto.dev/ (https://ui.perfetto.dev/)

- Click on the top left menu, “Open Trace File on the left top”

- Select the file results.json

Zoom in/out: W/S
Move left/right: A/D



Slice Details

Name	bodyForce(Body*, float, int) [clone .kd]
Category	null
Start time	231ms 692us 100ns
Duration	26ms 299us
Thread duration	0s (0.00%)
Thread	3671242
Process	GPU0 6
Slice ID	5382

Preceding flows

Slice	hsa_dispatch
Delay	1ms 862us
Thread	NULL (CPU HSA API 3)

Arguments

args

BeginNs	1591844003899555
DurationNs	26299297
EndNs	1591844030198852
pid	3671242
tid	3671242

Read about the counters: `vim /opt/rocm/rocprofiler/lib/gfx_metrics.xml`

- Create a file with the contents:

```
cat rocprof_counters.txt
pmc : Wavefronts VALUInsts VFetchInsts VWriteInsts VALUUtilization VALUBusy Wri
pmc : SALUInsts SFetchInsts LDSInsts FlatLDSInsts GDSInsts SALUBusy FetchSize
pmc : L2CacheHit MemUnitBusy MemUnitStalled WriteUnitStalled ALUStalledByLDS LD
```

- Execute with using the counters

```
 rocprof --timestamp on -i rocprof_counters.txt  nbody-orig 65536
RPL: on '221130_205737' from '/global/software/rocm/rocm-5.3.0' in '/global/hom
RPL: profiling '"nbody-orig" "65536"'
RPL: input file 'rocprof_counters.txt'
RPL: output dir '/tmp/rpl_data_221130_205737_3673574'
RPL: result dir '/tmp/rpl_data_221130_205737_3673574/input0_results_221130_2057
ROCProfiler: input from "/tmp/rpl_data_221130_205737_3673574/input0.xml"
  gpu_index =
  kernel =
  range =
  7 metrics
    Wavefronts, VALUInsts, VFetchInsts, VWriteInsts, VALUUtilization, VALUBusy,
65536, 155.389

ROCPRofiler: 10 contexts collected, output directory /tmp/rpl_data_221130_20573
RPL: result dir '/tmp/rpl_data_221130_205737_3673574/input1_results_221130_2057
ROCProfiler: input from "/tmp/rpl_data_221130_205737_3673574/input1.xml"
  gpu_index =
  kernel =
  range =
  7 metrics
    SALUInsts, SFetchInsts, LDSInsts, FlatLDSInsts, GDSInsts, SALUBusy, FetchSi
65536, 156.996

ROCPRofiler: 10 contexts collected, output directory /tmp/rpl_data_221130_20573
RPL: result dir '/tmp/rpl_data_221130_205737_3673574/input2_results_221130_2057
ROCProfiler: input from "/tmp/rpl_data_221130_205737_3673574/input2.xml"
  gpu_index =
  kernel =
  range =
  6 metrics
    L2CacheHit, MemUnitBusy, MemUnitStalled, WriteUnitStalled, ALUStalledByLDS,
65536, 155.264

ROCPRofiler: 10 contexts collected, output directory /tmp/rpl_data_221130_20573
File '/global/home/gmarko/HIP-Examples/mini-nbody/hip/rocprof_counters.csv' is
```

- Contents of the rocprof_counters.csv file

```
cat rocprof_counters.csv
Index,KernelName,gpu-id,queue-id,queue-index,pid,tid,grd,wgr,lds,scr,arch_vgpr,
0,"bodyForce(Body*, float, int) [clone .kd]",0,0,0,3673711,3673711,65536,256,0,
...
```

Omnitrace

- Load Omnitrace

```
module use /project/project_465000502/software/omnitrace192/share/modulefiles/
module load omnitrace/1.9.2
```

- Use your own space, for example:
  `mkdir /scratch/project_465000502/$USER`
  `cd /scratch/project_465000502/$USER`

- Allocate resources with `salloc`

```
salloc -N 1 -p small-g --gpus=1 -t 10:00 -A project_465000502
```

- Check the various options and their values and also a second command for description

```
srun -n 1 --gpus 1 omnitrace-avail --categories omnitrace
srun -n 1 --gpus 1 omnitrace-avail --categories omnitrace --brief --description
```

- Create an Omnitrace configuration file with description per option

```
srun -n 1 omnitrace-avail -G omnitrace.cfg --all
```

- Declare to use this configuration file:

```
export OMNITRACE_CONFIG_FILE=/path/omnitrace.cfg
```

- Get the training examples:

```
cp -r /project/project_465000502/exercises/AMD/HPCTrainingExamples .
```

- Compile and execute saxpy
  - `cd HIP/saxpy`
  - `hipcc --offload-arch=gfx90a -O3 -o saxpy saxpy.cpp`
  - `time srun -n 1 ./saxpy`

- Check the duration

- Compile and execute Jacobi
  - `cd HIP/jacobi`
  - `make`
  - `time srun -n 1 --gpus 1 Jacobi_hip -g 1 1`

- Check the duration

Dynamic instrumentation

- Execute dynamic instrumentation: `time srun -n 1 --gpus 1 omnitrace-instrument -- ./saxpy` and check the duration
- Execute dynamic instrumentation: `time srun -n 1 --gpus 1 omnitrace-instrument -- ./Jacobi_hip -g 1 1` and check the duration
- Check what the binary calls and gets instrumented: `nm --demangle Jacobi_hip | egrep -i ' (t|u) '`
- Available functions to instrument: `srun -n 1 --gpus 1 omnitrace -v 1 --simulate --print-available functions -- ./Jacobi_hip -g 1 1`
  - the simulate option means that it will not execute the binary

Binary rewriting (to be used with MPI codes and decreases overhead)

- Binary rewriting: `srun -n 1 --gpus 1 omnitrace-instrument -v -1 --print-available functions -o jacobi.inst -- ./Jacobi_hip`
  - We created a new instrumented binary called jacobi.inst
- Executing the new instrumented binary: `time srun -n 1 --gpus 1 omnitrace-run -- ./jacobi.inst -g 1 1` and check the duration
- See the list of the instrumented GPU calls: `cat omnitrace-jacobi.inst-output/TIMESTAMP/roctracer.txt`

Visualization

- Copy the `perfetto-trace.proto` to your laptop, open the web page https://ui.perfetto.dev/ (https://ui.perfetto.dev/) click to open the trace and select the file

Hardware counters

- See a list of all the counters: `srun -n 1 --gpus 1 omnitrace-avail --all`
- Declare in your configuration file: `OMNITRACE_ROCM_EVENTS = GPUBusy,Wavefronts,VALUBusy,L2CacheHit,MemUnitBusy`
- Execute: `srun -n 1 --gpus 1 omnitrace-run -- ./jacobi.inst -g 1 1` and copy the perfetto file and visualize

Sampling

Activate in your configuration file `OMNITRACE_USE_SAMPLING = true` and `OMNITRACE_SAMPLING_FREQ = 100`, execute and visualize

Kernel timings

- Open the file `omnitrace-binary-output/timestamp/wall_clock.txt` (replace binary and timestamp with your information)

- In order to see the kernels gathered in your configuration file, make sure that `OMNITRACE_USE_TIMEMORY = true` and `OMNITRACE_FLAT_PROFILE = true`, execute the code and open again the file `omnitrace-binary-output/timestamp/wall_clock.txt`

Call-stack

Edit your omnitrace.cfg:

```
OMNITRACE_USE_SAMPLING = true;
OMNITRACE_SAMPLING_FREQ = 100
```

Execute again the instrumented binary and now you can see the call-stack when you visualize with perfetto.

Omniperf

- Lod Omniperf:

```
module load cray-python
module use /project/project_465000502/software/omniperf108/modules
module load omniperf
export ROOFLINE_BIN=/project/project_465000502/software/omniperf108/bin/utils/r
```

- Reserve a GPU, compile the exercise and execute Omniperf, observe how many times the code is executed

```
salloc -N1 -n 2 -p small-g --gpus 1 -t 00:40:00
cp -r /project/project_465000502/exercises/AMD/HPCTrainingExamples .
cd HPCTrainingExamples/HIP/dgemm/
mkdir build
cd build
cmake ..
make
cd bin
srun -n 1 omniperf profile -n dgemm -- ./dgemm -m 8192 -n 8192 -k 8192 -i 1 -r
```

- Run `srun -n 1 --gpus 1 omniperf profile -h` to see all the options

- Now is created a workload in the directory workloads with the name dgemm (the argument of the -n). So, we can analyze it

```
srun -n 1 --gpus 1 omniperf analyze -p workloads/dgemm/mi200/ &> dgemm_analyze.
```

- If you want to only roofline analysis, then execute: `srun -n 1 omniperf profile -n dgemm --roof-only -- ./dgemm -m 8192 -n 8192 -k 8192 -i 1 -r 10 -d 0 -o dgemm.csv`

There is no need for srun to analyze but we want to avoid everybody to use the login node. Explore the file `dgemm_analyze.txt`

- We can select specific IP Blocks, like:

```
srun -n 1 --gpus 1 omniperf analyze -p workloads/dgemm/mi200/ -b 7.1.2
```

But you need to know the code of the IP Block

- If you have installed Omniperf on your laptop (no ROCm required for analysis) then you can download the data and execute:

```
omniperf analyze -p workloads/dgemm/mi200/ --gui
```

- Open the web page: http://IP:8050/ (http://IP:8050/) The IP will be displayed in the output