NB05-timing-hwcounters

September 22, 2022

Pathfinder Timing Hooks and Hardware Performance Counters

0.0.1 Lesson Objectives

Upon completing this notebook you should be able to understand and apply the following concepts:

- Learn about and utilize timing hooks to measure execution time with the emusim simulator
- Apply the basic concepts of performance measurement and performance counter measurements on the Pathfinder hardware

This notebook goes along with the Lucata profiling and timing slides and the hardware counter slides, so please follow along with the slides for a supplemental resource.

0.0.2 Environment Setup

We first need to initialize our environment to use the Lucata toolchain.

[1]:
```
%load_ext slurm_magic
import os
from IPython.display import Code

#Set the path to the latest toolset
LUCATA_BASE="/tools/emu/pathfinder-sw/22.09-beta"

#Get the path to where all code samples are
os.environ["USER_NOTEBOOK_CODE"]=os.path.dirname(os.getcwd())
os.environ["PATH"]=os.pathsep.join([os.path.join(LUCATA_BASE,"bin"),os.
 ↪environ["PATH"]])
os.environ["FLAGS"]="-I"+LUCATA_BASE+"/include/memoryweb"+" -L"+LUCATA_BASE+"/
 ↪lib -lemu_c_utils -lmemoryweb"
```

0.1 Lucata Timing Hooks

The Lucata toolchain includes a simulation profiler called `emusim_profile`. Running the profiler on your entire program can take a long time, so the toolchain provides timing hooks to specify regions of interest for performance profiling. Here we annotate SAXPY by placing timing hooks around the main computational kernel.

[2]:
```
Code('timing-hooks-saxpy.c')
```

[2]:
```c
//timing-hooks-saxpy.c
#include <stdio.h>
#include <stdlib.h>
#include <cilk/cilk.h>

#include "memoryweb.h"
#include <emu_c_utils/emu_c_utils.h>

void saxpy(long n, float a, float *x, float *y)
{
  for (long i = 0; i < n; i++)
    y[i] += a * x[i];
}

int main(int argc, char **argv)
{
  long num = atol(argv[1]); // number blocks
  long size = atol(argv[2]); // block size
  float aval = atof(argv[3]); // constant
  float **x = mw_malloc2d(num, size * sizeof(*x));
  float **y = mw_malloc2d(num, size * sizeof(*y));

  for (long j = 0; j < num; j++) {
    for (long i = 0; i < size; i++) {
      x[j][i] = j * size + i; y[j][i] = 0;
    }
  }

  lu_profile_perfcntr(PFC_CLEAR, "CLEAR COUNTERS");
  lu_profile_perfcntr(PFC_START, "START COUNTERS");

  //Set a timing hook for simulation
  hooks_region_begin("example");
  for (long i = 0; i < num; i++) {
    cilk_spawn_at (y[i]) saxpy(size, aval, x[i], y[i]);
  }
  cilk_sync;

  //Measure the simulated time at the end of the region
  double time_ms = hooks_region_end();

  lu_profile_perfcntr(PFC_STOP, "STOP COUNTERS");

  printf("time (ms) = %lf\n", time_ms);
}
```

0.1.1 Profiling with Timing Hooks

We can then compile and profile the code. This will generate a separate folder "saxpy_profile" with HTML output files that you can then investigate in further detail.

Note that this process might take a long time! For this reason it is important to only scope the region of interest in your code that you want to gather statistics for. Simulation should take under 3 minutes.

[3]:
```
%%bash
emu-cc -o timing-hooks-saxpy.mwx $FLAGS timing-hooks-saxpy.c
```

[4]:
```
%%bash
time emusim_profile saxpy_profile -m 24 --total_nodes 2 -- timing-hooks-saxpy.
↪mwx 8 128 5.0
```

```
Generating profile in saxpy_profile/timing-hooks-saxpy
emusim.x  -m 24 --total_nodes 2
timing-hooks-saxpy.mwx 8 128 5.0
Start untimed simulation with local date and time= Thu Sep 22 11:07:12 2022

Timed simulation starting…
{"region_name":"example","core_clk_mhz":175,"use_CORE_CLK_MHZ_envvar":0,"time_ms
":0.45,"ticks":79073}
time (ms) = 0.451846
End untimed simulation with local date and time= Thu Sep 22 11:09:26 2022


Info: /OSCI/SystemC: Simulation stopped by user.
Generating saxpy_profile/timing-hooks-saxpy_total_instructions.png
Generating saxpy_profile/timing-hooks-saxpy_total_migrations.png
Generating saxpy_profile/timing-hooks-saxpy.Thread_Enqueue_Map.png
Generating saxpy_profile/timing-hooks-saxpy.Memory_Read_Map.png
Generating saxpy_profile/timing-hooks-saxpy.Memory_Write_Map.png
Generating saxpy_profile/timing-hooks-saxpy.Atomic_Transaction_Map.png
Generating saxpy_profile/timing-hooks-saxpy.Remote_Transaction_Map.png
Generating saxpy_profile/timing-hooks-saxpy.MSP_Activity.png
Generating saxpy_profile/timing-hooks-saxpy.SRIO_Outgoing_Activity.png
Generating saxpy_profile/timing-hooks-saxpy.SRIO_Incoming_Activity.png
Generating saxpy_profile/timing-hooks-saxpy.Live_Threads.png
saxpy_profile/timing-hooks-saxpy.hpc exists
Find all graphs in: saxpy_profile/timing-hooks-saxpy_22-09-2022_11:11:33
The last hpc call to analyze will be 0
Program called lu_profile_perfcntr with message: STOP COUNTERS
Generating Graphs for [STOP COUNTERS]…
Stopping here after read 0
hpc_file_name_base: timing-hooks-saxpy.hpc
Report written to saxpy_profile/timing-hooks-saxpy-report.html, you may open it
```

```
in your browser now


        SystemC 2.3.3-Accellera --- Sep  7 2022 09:15:59
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
/net/tools/emu/pathfinder-sw/22.09-beta/bin/make_tqd_plots.py:12: UserWarning:
Creating legend with loc="best" can be slow with large amounts of data.
  plt.savefig(name, bbox_inches="tight")
/net/tools/emu/pathfinder-sw/22.09-beta/bin/make_hpc_plots.py:121:
MatplotlibDeprecationWarning: Passing non-integers as three-element position
specification is deprecated since 3.3 and will be removed two minor releases
later.
  plt.subplot(subplotX, subplotY, subplotNum)  # place the graph in the correct
subplot in the figure
/net/tools/emu/pathfinder-sw/22.09-beta/bin/make_hpc_plots.py:177: UserWarning:
Tight layout not applied. tight_layout cannot make axes height small enough to
accommodate all axes decorations
  plt.tight_layout()
/net/tools/emu/pathfinder-sw/22.09-beta/bin/make_hpc_plots.py:177: UserWarning:
Tight layout not applied. The bottom and top margins cannot be made large enough
to accommodate all axes decorations.
  plt.tight_layout()

real    5m6.224s
user    4m55.405s
sys     0m11.449s
```

View Simulation Timing Hook and Profile Output This should have generated the file `saxpy_profile/timing-hooks-saxpy-report.html`. Use the Jupyter file browser to navigate to `saxpy_profile` and open the report in your browser.

0.2 Measuring Performance on the Pathfinder Hardware

We've alluded to the usage of performance counters in many of our previous code examples. The commands below are all counter-related and are meant to be useful for both simulation and hardware execution.

```
//Clear counters, then start counting
lu_profile_perfcntr(PFC_CLEAR, "CLEAR COUNTERS");
lu_profile_perfcntr(PFC_START, "START COUNTERS");
...
//Stop counters

  lu_profile_perfcntr(PFC_STOP, "STOP COUNTERS");
```

To investigate this further, we've included a new code example to test with that sums across an array.

[5]:
```
Code('array-sum-hw-profiling.c')
```

[5]:
```
//array-sum-hw-profiling.c
#include <stdio.h>
#include <stdlib.h>
#include <cilk/cilk.h>

#include "memoryweb.h"
#include <emu_c_utils/emu_c_utils.h>

long sum(long **array, long epn) {

  long sum = 0;

  for (long i=0; i<NUM_NODES(); i++)
    for (long j=0; j<epn; j++)
      sum += array[i][j];

  return sum;
}


int main(int argc, char **argv)
{
  //allocate_arrays(); // Allocate arrays using malloc2d
  //initialize_arrays(); // Initialize arrays

  long size = atol(argv[1]); // block size

  //allocate arrays
  long epn = size/NUM_NODES(); // elements per node
  long** A = (long **) mw_malloc2d(NUM_NODES(), epn * sizeof(long));
  long** B = (long **) mw_malloc2d(NUM_NODES(), epn * sizeof(long));
  long sumA;
  long sumB;

  // Initialize array values
  for (long i=0; i<NUM_NODES(); i++){
    for (long j=0; j<epn; j++) {
        A[i][j] = 1;
        B[i][j] = 2;
    }
  }

  lu_profile_perfcntr(PFC_CLEAR, "CLEAR COUNTERS");
  lu_profile_perfcntr(PFC_READ, "READ COUNTERS AFTER CLEAR");
  lu_profile_perfcntr(PFC_START, "START COUNTERS");
```

```
  sumA = cilk_spawn sum(A, epn); // Spawn child thread for A
  sumB = sum(B, epn); // Sum values of B in parent thread
  cilk_sync; // Wait for spawned thread to complete
  printf("sumA = %ld, sumB = %ld\n", sumA, sumB);

  //Stop the performance counters
  lu_profile_perfcntr(PFC_STOP, "STOP COUNTERS AT END");

  return 0;
}
```

You can run this example with the simulation profiler again. Notice that it will generate some counter output in the profile folder.

[6]:
```
%%bash
time emusim_profile array_sum_profile -m 24 --total_nodes 2 --␣
 ↪array-sum-hw-profiling.mwx 8 128 5.0
```

```
Generating profile in array_sum_profile/array-sum-hw-profiling
emusim.x  -m 24 --total_nodes 2
array-sum-hw-profiling.mwx 8 128 5.0
Start untimed simulation with local date and time= Thu Sep 22 11:12:19 2022

[WARN]: In /home/tdysart/Toolchains/22.R2-Sept7/llvm-
cilk/mwsim/src/timsim.cpp:sc_main:590 Calling READCNTRS when not profiling!
Timed simulation starting…
sumA = 8, sumB = 16
End untimed simulation with local date and time= Thu Sep 22 11:12:33 2022


Info: /OSCI/SystemC: Simulation stopped by user.
Generating array_sum_profile/array-sum-hw-profiling_total_instructions.png
Generating array_sum_profile/array-sum-hw-profiling_total_migrations.png
Generating array_sum_profile/array-sum-hw-profiling.Thread_Enqueue_Map.png
Generating array_sum_profile/array-sum-hw-profiling.Memory_Read_Map.png
Generating array_sum_profile/array-sum-hw-profiling.Memory_Write_Map.png
Generating array_sum_profile/array-sum-hw-profiling.Atomic_Transaction_Map.png
Generating array_sum_profile/array-sum-hw-profiling.Remote_Transaction_Map.png
Generating array_sum_profile/array-sum-hw-profiling.MSP_Activity.png
Generating array_sum_profile/array-sum-hw-profiling.SRIO_Outgoing_Activity.png
Generating array_sum_profile/array-sum-hw-profiling.SRIO_Incoming_Activity.png
Generating array_sum_profile/array-sum-hw-profiling.Live_Threads.png
array_sum_profile/array-sum-hw-profiling.hpc exists
Find all graphs in: array_sum_profile/array-sum-hw-profiling_22-09-2022_11:13:00
The last hpc call to analyze will be 1
Program called lu_profile_perfcntr with message: READ COUNTERS AFTER CLEAR
Generating Graphs for [READ COUNTERS AFTER CLEAR]…
```

```
Program called lu_profile_perfcntr with message: STOP COUNTERS AT END
Generating Graphs for [STOP COUNTERS AT END]…
Stopping here after read 1
hpc_file_name_base: array-sum-hw-profiling.hpc
Report written to array_sum_profile/array-sum-hw-profiling-report.html, you may
open it in your browser now


        SystemC 2.3.3-Accellera --- Sep  7 2022 09:15:59
        Copyright (c) 1996-2018 by all Contributors,
        ALL RIGHTS RESERVED
/net/tools/emu/pathfinder-sw/22.09-beta/bin/make_hpc_plots.py:121:
MatplotlibDeprecationWarning: Passing non-integers as three-element position
specification is deprecated since 3.3 and will be removed two minor releases
later.
  plt.subplot(subplotX, subplotY, subplotNum)  # place the graph in the correct
subplot in the figure
/net/tools/emu/pathfinder-sw/22.09-beta/bin/make_hpc_plots.py:177: UserWarning:
Tight layout not applied. The bottom and top margins cannot be made large enough
to accommodate all axes decorations.
  plt.tight_layout()
/net/tools/emu/pathfinder-sw/22.09-beta/bin/make_hpc_plots.py:177: UserWarning:
Tight layout not applied. tight_layout cannot make axes height small enough to
accommodate all axes decorations
  plt.tight_layout()

real    1m51.451s
user    1m43.897s
sys     0m15.229s
```

Testing on the Pathfinder HW Then you can use sbatch to launch a single node job on the Pathfinder system.

[7]:
```
%%bash
#Clean up any older Slurm output files
rm -f *.out

#Run a single node
sbatch sbatch-array-sum-sn.sh

#If the job runs successfully, the output file should print out "SAXPY complete!
 ↪"
sleep 5
#Show the content of the latest output file
printf "\nOutput from the run:\n"
less slurm-*.out
```

```
Submitted batch job 134217833 on cluster pathfinder

Output from the run:
sumA = 8, sumB = 16
```

0.2.1 Postcript

Once we've finished our testing, we can clean up some of the logfiles that we used for this example with `make clean`. Uncomment the following line to clean this directory.

[8]:
```
#Uncomment this line to clean up your environment
#!make clean
```