EC700 – Advanced Computing Systems and Architecture – Spring 2018

Mini Project #1

Out: February 6, Tuesday. Due: **Wednesday, February 21, 11:59pm**.

You will be using Intel's **Pin** in this mini project and running SPLASH-3 benchmarks.

You can use eng-grid and your personal folder under the EC700 course folder for the project.

Task 0.

- Download and run Pin 3.5 with some of the examples given in the user manual. Useful links:
    - User manual: https://software.intel.com/sites/landingpage/pintool/docs/97503/Pin/html/
    - Tutorial: https://software.intel.com/sites/default/files/managed/62/f4/cgo2013.pdf
- Download and run the following applications from *SPLASH-3: radiosity, FMM, barnes*
    - https://github.com/SakalisC/Splash-3
    - You can use parsec-simlarge inputs initially.

Task 1. Counting instructions, memory references, and more (30 points)

- Write an instrumentation tool using Pin to count the total number of instructions executed and also to provide an instruction breakdown (e.g., which type of instruction was executed how many times).
- Run your SPLASH-3 benchmarks with 1, 2, 4, 8, 16 threads and record the instruction count and breakdown, as well the running time of the application (and other relevant statistics that you may find interesting).
- How is the performance changing across benchmarks? How about across different thread counts? How is the scalability? Does the instruction mix play a role in determining the performance trends, why/why not?
- Experiment with larger input sets as needed. Document your experimental choices.

Task 2. Tracing memory (35 points)

- Write an instrumentation tool using Pin to identify different heap objects that are allocated in an application. You can use the return address of memory allocation functions (such as malloc—there are others too) to distinguish different heap objects that are being allocated.
- Add the necessary instrumentation to record various heap objects, their identifiers (such as the return address), the size of each object allocated, and how many times each object is accessed.
- How does the heap usage of the applications you are experimenting with differ? How much diversity is there? Why would this matter from a memory management perspective (i.e., now that you know the memory access characteristics with some more detail, what would you do to improve the memory performance of a given application)?

Task 3. Instrumenting functions and more (35 points)

- Write an instrumentation tool to identify the return address of each "call" instruction (note the CALL instruction in x86).
- Implement a simple stack to push the return address of each call instruction once the program executes a call instruction. When the program returns (i.e., for "ret" instruction), check if the return address is the same as the address on top of the stack.
- What happens if the return addresses (of top of your stack and of the current ret instruction) do not match? Elaborate.

- Can you create a sample code where the return addresses do not match? Explain why this mismatch happens in your code.

Extra Credit (10 points). Creating your own simple performance simulator

- Use Pin as a base to design a simple performance simulator for a simple x86 processor with a 1GHz clock. For example, you may assume arithmetic instructions take a cycle, branches take 2 cycles, and jumps take a cycle. Assume there is no cache, and all memory accesses take 50 cycles. Document all your assumptions.
- Then, advance your memory model to model a heterogeneous memory case where larger/more frequently-accessed heap objects can be placed in a specialized scratchpad memory with shorter access latency.
- Report your performance results for all your benchmarks. How does the scratchpad improve performance? Can you explain the differences across difference benchmarks (or different thread counts)?

What to submit:

A <u>2 to 4 page</u> report in pdf (minimum font size 10, single space, standard 1-inch margins).

In your report, you should answer the questions for each task by demonstrating evidence (e.g., results in the form of tables and figures) and provide sufficient and clear technical discussion. Your report should include (at least) the following sections:

- Description of experimental methodology:
  - Explain your experimental choices. Can someone else repeat your experiments and achieve the same results with the information you are providing?
- Task 1:
  - Answer the given questions. Make sure to provide evidence from your simulation results.
- Task 2:
  - Answer the given questions, provide evidence.
- Extra credit (optional):
  - Provide experimental choices, results, and reasoning.

All figures and tables should be clearly labeled (make sure to have axes labels, captions, etc.) and should be referred to in text (e.g., Fig. 3 explains …). Make sure you explain your results and provide sufficient technical discussion.

Any external resource should be cited properly; otherwise, using un-cited work that is not your own original work may be considered as plagiarism.

Your report could be <u>closer to 2 pages rather than 4</u>. Extra space is given to you so that you can fit in your figures, tables, etc. more easily. Length is not a grading metric (as long as you are within the given page limit), but the <u>quality of presentation is</u>. Unclear figures, unlabeled axes, results without discussion, responses without evidence, etc. are all potential weaknesses you should avoid.

How to submit:

Email your report (pdf) to acoskun@bu.edu.

The filename should be: yourBUusername_miniProject1.pdf. The subject line of the email should be: EC700 miniProject1.