Lab #2: Hardware Counter Guided Transformation

Lab 2 is due Monday, Oct. 16. Your lab report and source code must be submitted by **1:50 PM** before class. The late policy applies to this lab project.

This lab is to be done individually. Get started early! Also, remember to **put your name** on the lab report!

THE PAPI HARDWARE COUNTER LIBRARY: 1

PROBLEM 1: TRANSFORM CODE GUIDED BY HARDWARE COUNTERS 1

1.1 REDESIGN THE STRUCT "MEM" 1

1.2 CO-OPTIMIZE THE STRUCT "MEM" AND THE FUNCTION "FUNC" 1

PROBLEM 2: NODE PROFILE, EDGE PROFILE, AND PATH PROFILE 2

2.1 PROFILING: 2

2.2: PRACTICE USING HARDWARE COUNTERS AND OPTIMIZE FOR MEMORY HIERARCHY 2

HOW TO SUBMIT: 2

The PAPI Hardware Counter Library:

You can solve the Problems on any machine with a C compiler and support for the PAPI library. The machine cpeg655.ece.udel.edu satisfies the requirements. If you decide to use another system, please describe the CPU configuration, the OS and the compiler on that system in your lab report.

The PAPI library has been installed under the directory /usr. The library binaries are in /usr/lib, and the library header files are in /usr/include. When you need to compile your program with PAPI, you can use the command line: "gcc -I/usr/include your_program_file -L/usr/lib -lpapi".

Problem 1: Transform Code Guided By Hardware Counters

1.1 Redesign the struct "Mem" in mem.c to ***minimize*** the L2 cache miss. You may use struct-of-array, array-of-struct, or any combination. You CANNOT change the structure of the code in the function "func", NOR change the order of operations. You can only revise the references to the variables depending on your design of the struct. Explain why your new design will minimize the L2 cache miss.

1.2 Co-optimize the struct "Mem" and the function "func" to minimize the L2 cache miss. The only constraint is that your new code should implement the same workload, i.e., the same operations in the original code, and use the same amount of data. You can change the order of operations as well as the struct “Mem” in any way you want. Explain why your new design will minimize the L2 cache miss.

Problem 2: Node Profile, Edge Profile, and Path Profile

2.1 Profiling: Instrument the function "func" of program prof.c to gather the node profile, the edge profile and the path profile of the function. Treat the “for” loops in the function “func” as nodes, i.e., you don’t need to go in the for loops to generate paths. The "func" function has a number of arguments. The corresponding main function provides two possible input sets for those arguments. You should do the profiling for both input sets. You need to submit 3 versions for the program, each version measuring one type of profile. Draw the CFG of the "func" function and include it in the report.

2.2: Practice Using Hardware Counters and Optimize for Memory Hierarchy

In this problem you are required to using PAPI hardware counter library to measure the memory hierarchy performance of ***all the paths*** in the function “func”. In other words, you should report the L2 cache misses for each path in the function “func”. You don’t need to measure or report the counters for other parts of the program. Furthermore, you should optimize/de-optimize the function under two input scenarios for memory hierarchy and again use PAPI to verify their memory hierarchy performance.

(1) Use the PAPI library to measure the L2 cache miss of the function "func" of the program. The "main" function has already set up the initialization of the PAPI library. You only need to provide the event names in the line that is labeled with "Please add your event here." Submit your code and measurements in your report. You should measure for both input sets.

(2) Transform the func function so that for ***two input sets , the most frequently executed path*** in the "func" function can achieve ***minimum*** of L2 cache miss. The "func" is basically a sequence of memory accesses. You can change the order of the memory accesses, but you cannot add or remove memory accesses to the sequence. You may also change the data struct declaration for the transformation. Note that you are required to only work on the most frequently executed path, implying that you may sacrifice other paths to achieve the goal. The most important grading criteria is WHY you do what your do. Submit your code and measurements, together with your explanation of the transformations, i.e., why they work. (Hint: Optimize/de-optimize by avoiding/creating cache conflicts.)

How to Submit:

Copy your lab report, which is a .pdf, a .doc, or a .html file, and all your source code into an empty directory. Assuming the directory is "submission", make a tar ball of the directory using the following command:

tar czvf [your_first_name]_[your_last_name]_lab2.tar.gz submission.

Replace [your_first_name] and [your_last_name] with your first name and your last name.

Submit the tar ball. The submission time will be used as the time-stamp of your submission.
