Programming Assignment 2 (Cache Attack Mitigation Techniques)
CS 665: Secure Memory Systems, Fall 2018
Computer Science and Engineering
Indian Institute of Technology Kanpur
Due Date: 22nd October, 2018, 11:55 PM

In this assignment, you will implement a LLC replacement policy (as discussed in the class) that prevents cross-core eviction attacks such as evict+reload and prime+probe.

You have to use a trace based cache simulator called ChampSim:

https://github.com/ChampSim/ChampSim

Read the README carefully and make your hands dirty with the ChampSim.

You have to add your replacement policy inside the replacement folder.

This section will provide the task list which needs to be accomplished by you for this assignment. The task list for this assignment is as follows:

A. **[You must be kidding: 2 points]** ChampSim does not implement an inclusive LLC. So, your first goal is to make the LLC inclusive. Once the LLC is inclusive, demystify the statement: "the back-invalidation hits at the private caches is small" with LRU replacement policy. Note that LRU replacement policy is already implemented in ChampSim (https://github.com/ChampSim/ChampSim/blob/master/replacement/lru.llc_repl)

B. **[Time for the mitigation: 8 points]** Implement the LLC replacement policy that prevents cross-core evictions that result in back-invalidation hits assuming LLC implements the LRU policy. You can run a simple 2-core setup and you have to show that your policy prevents the basis of any cross-core eviction based attacks.

C. **[Time to move on: 5 points]** In **Task B,** the replacement policy is restricted to LRU only. Change your code so that it can be applied to any replacement policy (knock the door of Biswa for this). Check the performance loss with your change.

D. [**Show me something new: 5 points**] Enhance your policy so that performance loss will be minimal.

For B, C, and D, Biswa will test it on hidden trace files. So do not assume anything about the program behavior.

As ChampSim works on memory trace of a program, so you need to generate a trace for your program for ChampSim. The steps for creating trace file using the Pin Tool champsim_tracer is as follows:

1. Download the pint tool from the link given below: https://software.intel.com/sites/landingpage/pintool/downloads/pin-3.2-81205-gcc-linux.tar.gz
2. Extract it
3. Set an environment variable PIN_ROOT with the path of PIN tool.
   export PIN_ROOT=path/to/pin-3.2-81205-gcc-linux
4. Add the path of pin-3.2-81205-gcc-linux in the PATH environment variable **eport PATH=$PATH:path/to/pin-3.2-81205-gcc-linux** (add this line at the end of .bashrc file)
5. **cd path/to/ChampSim/tracer**
6. ./make_tracer.sh

By follwing these steps, your **champsim_tracer.so** library is ready at the location **path/to/ChampSim/tracer/obj-intel64/champsim_tracer.so**

Now you can use this tracer to generate trace file for your program (say **testProgram**) as follows:

pin -t path/to/ChampSim/tracer/obj-intel64/champsim_tracer.so -- path/to/ testProgram

By applying the above command, a trace file will be created with name champsim.trace in your current working directory. If you get an error something like as shown in the box below:

```
A: Source/pin/injector_nonmac/auxvector.cpp: CopyAux: 291:
unexpected AUX VEC type 26

NO STACK TRACE AVAILABLE
Detach Service Count: 1
Pin 3.2
Copyright (c) 2003-2016, Intel Corporation. All rights reserved.
@CHARM-VERSION: $Rev: 81201 $
C: Injector exited with signal 6
E: Wait for injector failed: No child processes
Segmentation fault (core dumped)
```

Then include -ifeellucky at the time of trace generation, the syntax is as follows:

pin -ifeellucky -t path/to/ChampSim/tracer/obj-intel64/champsim_tracer.so -- path/to/ testProgram

By following above mentioned steps you can generate a trace file for you program.

[Note: All the tasks have to be be done on Linux Operating System. We will accept your submission via Canvas only, and other submission mode is strictly prohibited such as submitting via email or piazza. While using Piazza/email, use PA1 in the header]

Deliverables:

1. Source code for A and B, C, and D. Document your code properly. Share your results and experiences in a pdf: why something worked/did not work. Any insights, any experiments that you did to validate your assumption/confusion. Make sure to document your code well. Also, In the pdf, provide division of labor: Who did what, Say Biswa1 did A, Biswa2 did B and 25% of C.
2. Feel free to discuss about the assignment among your friends and with Biswa. However, at the end of the day, we want you to do everything on your own.