C152 Laboratory Exercise 5

Professor: Krste Asanovic
TA: Andrew Waterman
Department of Electrical Engineering & Computer Science
University of California, Berkeley

April 13, 2010

1 Introduction and goals

The goal of this laboratory assignment is to allow you to conduct some simple virtual experiments in the Simics simulation environment. Using Simics models of the multithreaded Sun UltraSPARC T1 processor, you will collect statistics and make some architectural recommendations based on the results.

The lab has two sections, a directed portion and an open–ended portion. Everyone will do the directed portion the same way, and grades will be assigned based on correctness. The open–ended portion will allow you to pursue more creative investigations, and your grade will be based on the effort made to complete the task or the arguments you provide in support of your ideas.

Students are encouraged to discuss solutions to the lab assignments with other students, but must run through the directed portion of the lab by themselves and turn in their own lab report. For the open-ended portion of each lab, students can work individually or in groups of two or three. Any open-ended lab assignment completed as a group should be written up and handed in separately. Students are free to take part in different groups for different lab assignments.

You are only required to do one of the open ended assignments. These assignments are in general starting points or suggestions. Alternatively, you can propose and complete your own open ended project as long as it is sufficiently rigorous. If you feel uncertain about the rigor of a proposal, feel free to consult the TA or professor.

This lab assumes you have completed the earlier laboratory assignments. However, we will re-include all the relevant files from past labs in this lab's distribution bundle for your convenience. Furthermore, we will assume that you remember all the commands used in earlier labs for controlling Simics simulation. If you feel any confusion about these points, feel free to consult the first lab guide or the Simics User Guide.

As a final note, the open–ended portion of this lab may take significant simulation time. It is recommended you start early so you are able to get enough cycles on the servers.

1.1 The target machine

For this lab you will use a simulation of the Sun Microsystems UltraSPARC T1 processor called Niagara-Simple running Solaris. This processor uses the SPARCv9 architecture. It has 8 cores, each of which has 4 hardware thread contexts that appear to the OS as separate cores. This

machine employs fine–grained multithreading, meaning that each core switches between one of the available threads on every cycle. Available configurations of this machine in Simics have 1 (1 core x 1 context), 2 (2 cores x 1 context), or 32 (8 cores x 4 contexts each) thread contexts. Solaris treats each of the hardware thread contexts as a separate CPU.

This configuration has been improved over the one used in Lab 5 by the addition of a new level of private caches. Each of the eight cores now has its own private L1 cache. The four threads on each core share their core's private L1 cache. All L1 caches are write-through and are backed by a shared L2 cache. The private caches are kept coherent via the MESI protocol (bus-based) described in lecture.

1.2 The multithreaded benchmark

The benchmark we will use in this lab is a multithreaded implementation of a simulation of 3D Lattice-Boltzmann magneto-hydrodynamics (in other words, plasma turbulence). Pthreads are used to implement threading. The program is run with the following 8 command line parameters:

```
lbmhd_solaris [X dimension size] [Y dimension size] [Z dimension size]
 [number of threads] [level of compiler optimization (0-2)] [vector length (1-128)]
 [amount of loop unrolling (1-8)] [ILP (1-8)]
```

The first three parameters specify the size of the problem, the 4th specifies the number of threads used to compute the answer, and the final four parameters specify the number and degree of optimizations used during execution.

1.3 Graded Items

You will turn a hard copy of your results to the professor or TA. Please label each section of the results clearly. Make sure you name your open–ended section partners in your individual portion, and that the group results name you as a participating member. The following items need to be turned in for evaluation:

1. Problem 2.2: Niagara cache statistics and performance
2. Problem 3.1 or Problem 3.2 statistics and evaluations

2 Directed Portion

2.1 Setup

In this lab section you will boot up the Niagara machine and create checkpoints. If you saved the 32 thread checkpoints from Lab 5, they can be used here, but you a need a checkpoint where no caches have been added to the system. Whether starting from a previous checkpoint of from scratch, make sure you accomplish all of the following:

1. Start Simics.
   ```
   host$ ./simics
   ```
2. Set the number of CPUs.
   ```
   simics> $num_cpus = 32
   ```

3. Run the configure script and let the machine boot. This will take additional time for the machine with 32 contexts.
   ```
   simics> run-command-file targets/niagara-simple/
   niagara-simple-solaris-common.simics
   ```

4. Load the host machine's file system (`mount /host`) and copy in the file `lbmhd_solaris`.

5. Create a checkpoint for this machine configuration.

6. Change directories to the location where you extracted your lab5 materials.
   ```
   simics> cd lab5/
   ```

7. Run the script `add-breakpoint.simics`. This will add a breakpoint at the appropriate point in the execution of LBMHD.
   ```
   simics> run-command-file add-breakpoint.simics
   ```

8. Begin execution of LBMHD for the following set of command line parameters:
   ```
   target# lbmhd_solaris 32 32 32 32 1 8 2 1
   ```

9. Simics should breakpoint after the LBMHD process reaches the "Benchmarking..." phase. Create a new checkpoint at this time. Setup is complete. Exit Simics.

2.2 Running multithreaded code on a Sun Niagara T1 processor

In this section, you will run the multithreaded workload and study the performance of the private caches and the MESI protocol.

1. Start Simics in `stall` mode, continuing from the configuration you saved in the last setting. `cd` to the directory where your Lab 5 material was extracted and run a script to connect the 2 level cache hierarchy. This script adds eight private L1 caches (`l1cache[0-7]`), one shared L2 cache (`l2cache`) and one memory unit (`staller`).
   ```
   simics> cd lab5/
   simics> run-command-file add-niagara-2level-cache.simics
   ```

2. Run the simulation for 1,000,000 instructions to warm the caches.
   ```
   simics> c 1000000
   ```

3. Once the caches are warm, reset their statistics so that we can collect accurate data from them. `simics> l2cache.reset-statistics`
   ```
   simics>  l1cache0.reset-statistics
   simics>  l1cache1.reset-statistics
   etc...
   ```

4. Run the simulation for at least 10,000,000 instructions to collect data.
   ```
   simics> c 10000000
   ```

5. Examine the statistics for the L1 and L2 caches. Note that the `statistics` command now reports some MESI statistics. Record the data read, data write, and instruction fetch hit rates for the L2 cache, and the average data read, data write and instruction fetch rates and average MESI statistics for the L1 caches.

6. The data you just collected was for 8KB L1 caches and a 512 KB L2. Exit Simics and edit the `configure-caches.py` file to change the L1 cache sizes to 64 KB.

7. Restart from checkpoint you created for this parameterization of LBMHD, and repeat the experiment.

How did the cache miss rates change? How did the MESI statistics change? Explain your observations.

3 Open-ended Portion

3.1 Design space exploration of the shared memory hierarchy

You are the architect in charge of designing the memory hierarchy for a chip multiprocessor suspiciously similar to the UltraSPARC T1. You are being given a budget of 1MB of on-chip cache, which must be allocated in some manner between the private L1s and shared L2.

Edit the `configure-caches.py` file to change the size of the various caches. You can change the number of cache lines, size of each line, and associativity of each cache, but do not change its write policy (only the lowest level cache can be write-back). Make sure the total size of all your caches is less than 1 MB.

The baseline layout for the memory hierarchy is as described in the previous sections (8 I/D combined private L1s, 1 shared L2). You may stick with this hierarchy, or if you are feeling creative you can create additional levels or distinctions. Note that the `snoopers` and `higher_level_caches` parameters are used to enable MESI coherence. See the user guide for more details or email for help if you are having trouble creating a valid configuration.

The simplified cache access time rule is as follows: A cache of 1KB or smaller has a 1 cycle latency. Every doubling in size results in an increase of 1 cycle, up to a flat, shared 1MB cache which has a latency of 10 cycles. Round up. Make sure that as you change sizes you adjust your latencies appropriately.

Evaluate by running instances of LBMHD and recording the cache statistics. Make sure to state what parameters you are using. Remember that you have to run in `stall` mode for the caches to work.

Report on your chosen configuration and provide a convincing argument as to why it is the best. Report also on any configurations you tested that performed poorly.

3.2 Propose a project

Propose a project of your own. You should have a feel for the kinds of projects that might be appropriate given the work you have done this term. If you have knowledge of parallel programming libraries or languages a project related to optimizing a program for a specific machine or architecture would certainly be acceptable.