C Changes in Version 1.3. – April 16th 2012

To switch from version 1.0 or 1.1 or 1.2 to version 1.3, download the new version and copy a previously written scheduler.c and scheduler.h to the src/ directory in the new version.

The version 1.3 distribution now also includes a subset of the workloads that will be used for the final competition. Here are the noteworthy points about the workloads:

- The following ten benchmark traces are included in the distribution (13 files):
  1. *black:* A single-thread run from PARSEC's blackscholes.
  2. *face:* A single-thread run from PARSEC's facesim.
  3. *ferret:* A single-thread run from PARSEC's ferret.
  4. *fluid:* A single-thread run from PARSEC's fluidanimate.
  5. *freq:* A single-thread run from PARSEC's freqmine.
  6. *stream:* A single-thread run from PARSEC's streamcluster.
  7. *swapt:* A single-thread run from PARSEC's swaptions.
  8. *comm1:* A trace from a server-class transaction-processing workload.
  9. *comm2:* A trace from a server-class transaction-processing workload.
  10. *MT*-canneal:* A four-thread run from PARSEC's canneal, organized in four files, MT0-canneal to MT3-canneal.

- Benchmarks black, face, ferret, freq, stream have about 500 million instructions. These instructions were selected from 5 billion instruction traces with a methodology similar to that of Simpoint[1].

- Benchmarks fluid and swapt are also defined with the Simpoint-like methodology described for the other PARSEC benchmarks. The only difference is that the traces include 750 million instructions so they have execution times similar to the other benchmarks.

- Benchmarks comm1 and comm2 are roughly 500 million instruction windows that are representative of commercial transaction-processing workloads.

[1]SimPoint [28] is used to generate traces which are representative of the benchmarks. SimPoint uses Basic Block Vectors(BBVs) [28] to recognise intervals of the execution which can be used to replicate the behavior of the benchmark. It assigns weights to each interval, which can be applied to the results obtained from each interval.

In our model, each benchmark is simulated for 5 billion instructions with interval sizes of 5 million instructions. A number of metrics are collected from each interval, including number of LLC misses, number of reads, number of writes, number of floating-point, integer, and branch instructions. The collection of these metrics forms the BBV that is used with SimPoint. The BBVs are classified into 10 clusters using SimPoint. The number of intervals selected from each cluster depends on the weight of each cluster. The final trace of 500M instructions is the combination of 100 intervals, taken from the large trace.

- The 4-thread canneal traces represent the first 500 million instructions executed by each thread once the region of interest is started. While all of the other traces were collected with 512 KB private LLCs for each single-thread program, the 4-thread canneal traces assumed a 2 MB shared LLC for the four cores. A write in a trace file represents the dirty block evicted by a block that is being fetched by that thread.

- The 10 traces are used to form 10 different workloads that will be used for the competition. All 10 workloads are run with 4channel.cfg, and the first 8 are also run with 1channel.cfg. The numbers from these 18 simulations will be used to compute the metrics that must be reported in the papers being submitted to the competition. The runsim file in the usimm directory lists all 18 simulations. The competition website will also have pointers to scripts, excel files, and latex table templates that can be used to compute and report the final metrics. The final competition results will be based on these 18 experiments and a few more; the additional experiments and workloads will be announced after the submission deadline. The 10 workloads are:

    1. comm2
    2. comm1 comm1
    3. comm1 comm1 comm2 comm2
    4. MT0-canneal MT1-canneal MT2-canneal MT3-canneal
    5. fluid swapt comm2 comm2
    6. face face ferret ferret
    7. black black freq freq
    8. stream stream stream stream
    9. fluid fluid swapt swapt comm2 comm2 ferret ferret
    10. fluid fluid swapt swapt comm2 comm2 ferret ferret black black freq freq comm1 comm1 stream stream

- The Fairness metric for the competition is being modified to be the following. Fairness is being defined as the maximum slowdown for any thread in the workload, relative to a single-program execution of that thread with an FCFS scheduler (a high number is bad). The final PFP metric will multiply the {average of maximum slowdowns across all experiments} and the {sum of execution times of all programs in those experiments}. For the PFP metric, only 14 of the 18 experiments will be used (the single-program comm2 workload and the multi-threaded canneal workload will not be used to evaluate fairness).

In going from version 1.2 to version 1.3, the code in files memory_controller.c, memory_controller.h and main.c have changed. USIMM Version 1.3 incorporates the following changes over version 1.2:

- Bug fix: The is_T_FAW_met is modified to correctly enforce the t_FAW condition. Earlier in a t_FAW rolling window, it would be possible for the scheduler to erroneously issue a maximum of five activations, (assuming the t_RRD timing condition was met). Now, the scheduler can issue a maximum of 4 activate commands in the t_FAW window.
- Bug fix: Changed the variable cas_issued_current_cycle to keep track of COL_RD or COL_WR commands issued to each bank. Earlier, the variable only kept track of whether a COL_RD or COL_WR had been issued in the current simulation cycle to a channel before issuing an autoprecharge. Also, the variable is now reset when an autoprecharge command is issued. This has no impact on correct implementations of the autoprecharge functionality. The change prevents schedulers from incorrectly issuing multiple auto-precharges to a channel in the same cycle and also prevents an autoprecharge to be sent to a bank that did not have a COL_RD or COL_WR issued to it that very cycle.
- Changes to statistics: New variables, stats_reads_merged and stats_writes_merged, counting the number of merged reads and writes respectively, have been exposed to the scheduler. The variables fetched and committed (which, respectively, contain the fetched and committed instruction counts for each simulated core) have been migrated from the file main.c to memory_controller.h to allow the scheduling algorithm to use this information. The simulator also now prints the sum of execution times on each core and the EDP metric for the simulation.