CMP SCI 635: Modern Computer Architecture

Professor Charles Weems

Evaluating Non-deterministic Multi-threaded Commercial Workloads

Name: Kunjal Panchal

Student ID: 32126469

Date: 10th Sept, 2019

Paper: 2 – Methodology [Hill02]

Strengths:

1. Removed cold-start transients and end transients. System is warmed up before we start evaluating its throughput for commercial workloads.

2. Use of real multiprocessor machine to develop workloads. That way the researchers actually took into account the multiprocessor speedup, microarchitectural statistics. And this, is obviously, faster than simulation for rapid performance testing. They actually addressed the issues one encounters while dealing with commercial applications under multiprocessor environment e.g. speed ups , scale ups and load imbalance. They observed the results on per processor basis by choosing Sun E6000 16 UltrasparcII Processors. And those disk images are used in simulation, which should be sufficient enough to get rid of all the ideal, unencounterable-in-actual-commercial-executables conditions.

3. Variety of programming environment and benchmarks. They incorporated the types of computing environment which are actually used commercially. Instead of sticking to just some big-scale C benchmark which might not depict what actually takes place while we are using a Java based database or web servers. Also, variety of pseudo-random requests to the servers are a plus when we are modelling client-side behaviour.

4. "*To mitigate the effects introduced by non-determinism, we run multiple simulations for each particular hardware configuration, and use their mean simulated execution time as our performance metric. We illustrate below how this methodology can be used to effectively separate systematic improvements from random effects caused by non-determinism.*" Shows that the conclusions of this paper aren't the result of observing only one run on only one configuration and from Figure 3, we can say that this method, indeed, improved the random effects generated by non-determinism.

Weaknesses:

1. Snapshots, checkpoints, inherently being a simulator, all the benchmarking performance analysing tools ultimately depend on real hardware and might add some overhead, which wasn't taken into account. Small as it might be, for large workloads, that might skew the observations.

2. Didn't address actual I/O wait time processors might encounter while dealing with commercial applications which are heavily based on user interaction. That could have added more non-determinism. Sometimes the wait for an I/O activity can be the real bottleneck for full multiprocessor utilization. (e.g. Servers might have a peak traffic time, rest of the time getting utilized <10%). The researchers have set client think time to be zero in all benchmarks, that's not practical.

3. As an addendum to the previous {Weakness#2} point, Table 1 shows that for workloads SPECjbb and Barnes-Hut, time spent in Kernel is 3% and 1% respectively, so assuming that they are not CPU-intensive, the researchers have greatly downplayed the role of Idle or I/O mode.

4. Lack of variety of time-sampling interval ("*...results of twenty separate runs, starting from the same checkpoint, measured every 200 transactions...*"). I would have liked to see results from different transactional checkpoints and interval being different; if there's a way to average out the results then.

Questions/Assertions:

1. What portion of the workload consists of the extra instructions added to point out end of each transaction? [Might add significant overhead.]

2. Wouldn't having fairly equal or almost equal amount of workload for each benchmark make more sense as we would get clear idea of throughput (e.g. *new order* will certainly have more instructions to execute, but couldn't we group multiple *order status* requests together or with some other less weighed instructions, if they are in fairly large amount, to balance the workload for each transaction?)

3. Author gives an explanation of why they avoided commercial workloads which might be better suited for servers ("*Full-size commercial workloads can have memory requirements of many gigabytes, and secondary storage requirements of several terabytes. These memory and disk requirements often stress execution on native systems, and it is not currently possible to simulate such large systems. Moreover, simulators usually run on host workstations that are much less powerful than server systems.*"). But can't we just analyse a small chunk of this applications after the cold-start transient. It might still be that the main and secondary memory requirements are exceeding but can it be that that difficulty might be overcome with virtual memory?

4. Related to the previous question, can virtual memory addressing do away with having to scale down the workloads? Because scaling down has its own downsides too (e.g. throughput penalty).

5. How would distributed-database servers, which are also a commercial multithreaded non-deterministic application, make impact on workload benchmarks. There will be one more layer of non-determinism because of coherency control and cache coherency on multiple multi-processor environments.

6. Are the pseudo-random requests generated at fixed time interval? Then this raises the same issue as {Weakness #2}.

7. While they did change the cache miss timings for OLTP simulations, would changing checkpoint timings also make dramatic difference in execution time? Would changing warm-up length will make difference? I guess so, because that will change which transactions are in workload.