CMP SCI 635: Modern Computer Architecture

Professor Charles Weems

Producing Wrong Data Without Doing Anything Obviously Wrong!

Name: Kunjal Panchal

Student ID: 32126469

Date: 05th Sept, 2019

Paper: 1 – Methodology [Diwan09]

Strengths:

1. Researchers used the benchmark multiple times, back-to-back. And took care of the OS activities that might persists for all the runs of a configuration. They gave satisfactory reasons why it might not be the case ("First, in several cases they did run additional runs of a configuration to explore some phenomenon in more detail and got reproducible results; if OS activity was causing bias this would not have been the case. Second, even for shortest running benchmark the OS activity would have to persist for over 6.5 seconds to bias our results and much longer (minutes) for the longer running benchmarks; this is unlikely.") Even though, it's a highly theoretical assumption.

2. The micro-architecture, the memory sizes, the cache sizes, TLB sizes were varied enough to make the produced results considerably generalized.

3. The authors actually figure out which environment variables might have the biggest contribution for the measurement bias. Which will be huge in stabilizing or limiting the bias caused by Unix environment size changes to the point where one can neglect it. And gives an explanation of the connection between load-store operations and alignments. But that hypothesis also couldn't be proven computationally as PEBS (precise event-based sampling) doesn't support load-store overlap event.

4. Measurement bias is commonplace because they have observed it for all of three microprocessors (one of them simulated), and using both the Intel and the GNU C compilers. And the results have been more or less similar. And it's a problem in other sciences too. That's a really strong observation. But also, I would like to know if the later or current research made any consideration for this bias.

Weaknesses:

1. The straddling of the violin plots near 1.0 really makes one think about the comparison between O2 and O3 optimizations with their inherent characteristics, which are not considered of importance for this paper.

If it seems weird that O2 could be faster than O3, an intuition for why this is somewhat common is that O3 does a lot of optimizations that increase code side (e.g., more aggressive loop unrolling), which almost always increases performance in microbenchmarks, but can sometimes decrease performance in workloads with a larger code footprint.

2. Compiler optimizations can do a decent job of making the software you wrote run faster, but it can't usually make up for you choosing a slower algorithm. It's not going to magically make iterating through a linked list fast enough to compete with a hash table. And this was tasted on SPEC CPU2006 C programs, why not use some other CPU benchmarking suites too. The author himself said that the SPEC 2006 suite is not diverse enough. And as the paper stated, it might be biased itself.

3. Even when we know there might be measurement biases, we still can't prevent it because we do not always know all the hardware specification and it's unpredictable in general. Measurement bias for one machine will never predict measurement bias for another one.

Author admits: "More generally, we find that inadequate information from hardware manufacturers and from the hardware severely cripples our ability to (i) understand the performance of a system and to (ii) fully exploit the capabilities of the hardware."

So, we can only hypothesise about how much measurement bias actually prevails and how varied or drastic its effect can be.

4. I would have liked to see how measurement bias affects computationally intensive programs. Rather than just focusing on memory intensive tasks.

5. They conducted this experiment in a highly ideal environment as well. Eg. removing all the inessential environment variables, not running any background or simultaneous processes, minimally-loaded machines etc.

6. Data wasn't collected for Pentium4 Floating Point benchmarks.

7. *perlbench* has given very different results than others, for example, for O3 speedup while changing environment size, how valid is it to focus on one benchmark over all others, maybe *perlbench* is the real anomaly here. It is the only one copying contents of environment to the heap too.

8. For evaluating diversity of different benchmarks and measuring the speedup, the authors are taking average of the observation. Wouldn't taking median make more sense?

9. Casual Analysis method feels like it's matching correlated events until we get a satisfying conclusion. It's trial and error till we want something to happen our way, that's also a bias.

10. Minor non-scientific point: While wanting to know more about microprocessors from the manufacturers, it's not feasible from a business point of view.

Questions/Assertions:

1. About the underlying architecture of the hardware; Pentium4 and core 2 both are RISC within CISC. And if so, will pure CISC or RISC processors would have given into measurement biases? Also, which architecture m5 O3CPU follows?

2. Can these results be, in any way, only specific to C language? Does the programming language's inherent characteristics be the reason we see measurement bias? Would the results be same if we try this with other language? Because C directly interacts with lower level architecture, while Java won't, as it has its own virtual machine. Will changing the environment size and link order (well, there's no linking in Java) still work? What parameters we use for tweaking when these two won't work. One parameter in particular, the Unix environment size, affects starting address of C stack, that might not be the case with other programming languages.

3. Are O2 and O3 compatible with each other? Can they really be compared? But the paper ignores the comparison between O2 and O3. And why is it between O2 and O3 particularly, why not Os, Og, Ofast?

4. Here it was given that changing Unix environment size changed the alignment of local variables. But can't there be some hardware structures where alignments are fixed, eg local variables only stored at even bytes of places. Then how would it affect the fact that measurement bias is due to environment getting loaded first, because then it won't matter where stack is, as the alignment will never change. And we don't know anything about another operating system than Unix. Also, can the results change if used other tool than PAPI, if not, there might be some inherent fault into structure of PAPI.

5. What things can we attribute program performance other than to memory layout.

6. How would multi-threaded applications make difference when memory layout is changed?

7. Can the fluctuations in speedups with different benchmarks with different linking order be result of size of .o files being in random order? Can it be that linking order depends and performs best when sizes are in increasing or decreasing order, rather than being random. {Although there's written that best link orders aren't same for different CPUs, so it's probably not dependent on size}

8. For linking order change, we can see that the median height of violin plots for integer benchmarks are significantly greater than of floating-point benchmarks. And it is mentioned that for pentium4, it's even more dramatic. Why is it so? You would think speedup with integers would be more stable than for floating points.

9. We can assert that Unix environment variable size changes, except for 2-3 benchmarks, produces tight violin plot, meaning that there must be not much of a measurement bias affecting it, barring the *perlbench* benchmark.

10. Meta point: My impression is that most papers pick "benchmarks" that show the most favourable results.