Architecturally-independent and time-based characterization of SPEC CPU 2017

Muhammad Hassan, Chang Hyun Park, David Black-Schaffer
Department of Information Technology
Uppsala University
Uppsala, Sweden
(hassan.muhammad, chang.hyun.park, david.black-schaffer)@it.uu.se

***Abstract*—Characterizing the memory behaviour of SPEC CPU benchmarks is critical to analyze bottlenecks in the execution. Unfortunately, most prior characterizations are tied to a particular system (e.g., via performance counters, fixed configurations) and missing important time-based behaviour (e.g., average over execution). While performance counters are accurate for that particular system, the results are less accurate for different micro-architectures and configurations. Most importantly, aggregate statistics (e.g., average over full execution) miss important time-based information which reveal transient phases that have significant impact on the execution. This work focuses on micro-architecturally independent, time-based characterization and analysis of the memory system behavior of SPEC CPU 2017. By collecting micro-architecturally independent and time-based information, we provide *reusable* data for various memory configurations.**

I. Introduction

SPEC CPU benchmarks ([1], [2]) are standard for computer system performance evaluation. Characterizing their memory system behaviour is critical for understanding the bottlenecks present in current memory systems and direct future research. However, prior works on workload characterization ([3]–[8]) are either coupled to a particular system (via hardware performance counters) and miss important time-based information (e.g., via aggregate statistics over the full execution). Workload characterization using hardware performance counters gives the results for a specific system with fixed configurations. While such results are accurate for that particular system and easy to collect, they are not accurate for other micro-architectures (e.g. Intel vs AMD) and not reusable for other configurations (different cache sizes or prefetchers). In addition to architectural-dependence, previous works have looked at aggregate statistics (average over full execution) which miss important time (or phase) based information that is critical in understanding bottlenecks of the execution. For example, gcc 2017 at 1GB cache without prefetching has almost 9% of execution (113B instructions) having an MPKI of $>$20 (Figure 2b ④), which contributes to almost 80% of the total cache misses, however the average MPKI of 4.5 reveals no such insight.

In this work, we provide micro-architecturally independent, time-based memory behaviour of SPEC CPU 2017 benchmarks with 16 different cache sizes, with and without prefetching. We first analyze the aggregate behaviour (average over the full execution) to understand how sensitive the applications are to different memory configurations and the changes from SPEC CPU 2006 to 2017. We then analyze applications time-based behaviour as a function of cache size which shows distinct phases of the execution. To simplify time-based analysis, we propose aggregating the data into *MPKI Bins*, which shows what percentage of the execution experience a certain MPKI as a function of cache size. MPKI bins enable compact yet insightful representation by revealing transient but memory intensive phases in the execution.

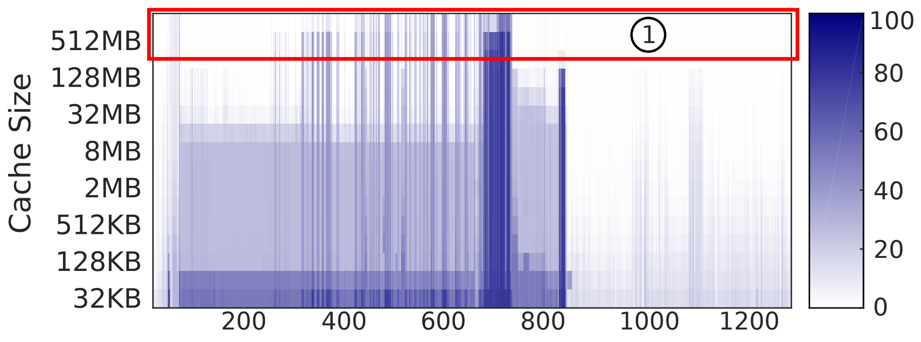
II. Methodology

We implemented a data cache simulator with a stride prefecther which is fed by Intel's Pin [9]. The prefetcher tracks misses based on the Program Counter (PC) and fetches the next 4 cache lines in the stride pattern. The simulator captures statistics every 100M instruction window. Benchmarks longer than 5T instructions were truncated. We collected data for 16 different cache sizes (32KB to 1GB, 8 way associativity), with and without prefetching.

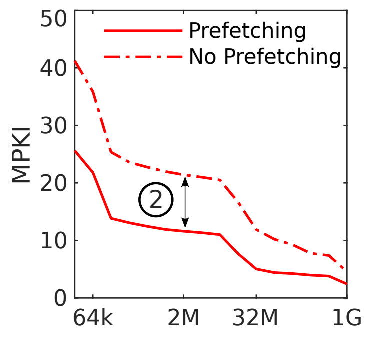
III. Results

Aggregate Results: Aggregate results provide the average behavior of the full application execution e.g. average MPKI for a single cache level with or without prefetching. These results show the behaviour of the applications to various memory configurations (cache size and prefetcher). We compared the cache and prefetcher sensitivity of SPEC 2006 and 2017 suites. The analysis shows larger cache working set size in the SPEC 2017 benchmarks (gcc, lbm, mcf, bwaves, cactuBSSN). But it also reveals cases where the SPEC 2017 benchmarks have improved cache and prefetcher behaviour (lower MPKI at the same size cache or more reduction in misses with prefetching) moving from 2006 to 2017 (wrf, xalancbmk, omnetpp, xz, x264, leela).

Time Based Results Though the aggregate results show that gcc with 1GB cache and prefetching has an average MPKI of 2.4, Figure 1a ① shows that there are some phases where the MPKI can be higher than 20. But that information is averaged

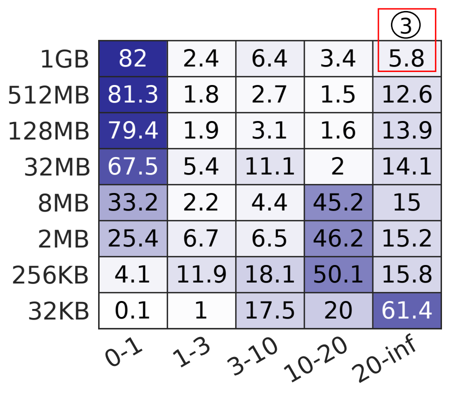


(a) MPKI heatmap of the execution (with prefetching) over time as a function of cache size (labels are in billions of instructions)



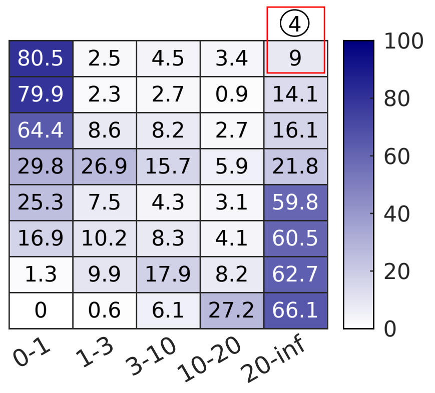
(b) Aggregate MPKI as a function of cache size

Fig. 1: Time-based analysis of gcc 2017 vs. an aggregate analysis. While both show the impact of varying the cache size, the aggregate analysis hides the fact that there are phases of very intense activity (MPKIs over 50) across nearly all cache sizes.



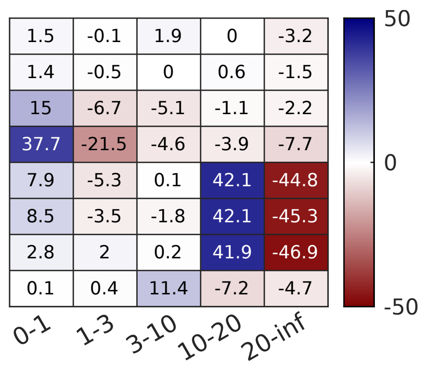
	0-1	1-3	3-10	10-20	20-inf
1GB	82	2.4	6.4	3.4	5.8
512MB	81.3	1.8	2.7	1.5	12.6
128MB	79.4	1.9	3.1	1.6	13.9
32MB	67.5	5.4	11.1	2	14.1
8MB	33.2	2.2	4.4	45.2	15
2MB	25.4	6.7	6.5	46.2	15.2
256KB	4.1	11.9	18.1	50.1	15.8
32KB	0.1	1	17.5	20	61.4

(a) With prefetching


④
100
80
60
40
20
0

	0-1	1-3	3-10	10-20	20-inf
1GB	80.5	2.5	4.5	3.4	9
512MB	79.9	2.3	2.7	0.9	14.1
128MB	64.4	8.6	8.2	2.7	16.1
32MB	29.8	26.9	15.7	5.9	21.8
8MB	25.3	7.5	4.3	3.1	59.8
2MB	16.9	10.2	8.3	4.1	60.5
256KB	1.3	9.9	17.9	8.2	62.7
32KB	0	0.6	6.1	27.2	66.1

(b) Without prefetching



	0-1	1-3	3-10	10-20	20-inf
1GB	1.5	-0.1	1.9	0	-3.2
512MB	1.4	-0.5	0	0.6	-1.5
128MB	15	-6.7	-5.1	-1.1	-2.2
32MB	37.7	-21.5	-4.6	-3.9	-7.7
8MB	7.9	-5.3	0.1	42.1	-44.8
2MB	8.5	-3.5	-1.8	42.1	-45.3
256KB	2.8	2	0.2	41.9	-46.9
32KB	0.1	0.4	11.4	-7.2	-4.7

(c) Difference of prefetching with no prefetching

Fig. 2: MPKI bins for gcc 2017: Percent of execution experiencing different MPKIs with (a) and without (b) prefetching, and the impact (change) of enabling prefetching (c).

out in aggregate statistics. Time-based results not only show the intensity of memory bottlenecks but also when they occur and how frequently.

MPKI Bins Though Figure 1a ① shows the phases which are memory intensive at 1GB cache, it does not quantify that almost 5.8% of the execution (almost 73B instructions) is in those memory bound phases. The cache misses that occur in these 73B instructions of high memory intensity make up almost 57% of the total cache misses. To extract further insight from time-based data we propose aggregating the data by behaviour into MPKI Bins. MPKI Bins shows us what percentage of the instructions experienced what range of MPKI for each cache size (Figure 2). Figure 2a and 2b shows MPKI bins distribution of gcc with and without prefetching respectively, for various cache sizes. Figure 2a ③ shows that at 1GB with prefetching, 5.8% of the execution is in the high MPKI region.

We can use Figure 2 to understand the effects of prefetching on gcc 2017. Consider a 2MB cache where prefetching cause a drop of 10 in MPKI (Figure 1b ②). Figure 2a shows that at 2MB only 15% of the execution has an MPKI >20 while without prefetching (Figure 2b), almost 60% of the execution has an MPKI>20, which results in a difference of 10 in MPKI. Figure 2c shows the difference in MPKI bins with and without prefetching. Figure 2c shows quantitatively that 45% of the execution went from having an MPKI of >20 to 10-20 with prefetching enabled. The interesting thing to note is that MPKI bins not only show how prefetching effects at a single cache size (horizontally) but also across different cache sizes (vertically).

IV. Conclusion

In this work we presented micro-architecturally independent, time-based analysis and characterization of the SPEC 2017 benchmarks with various memory configurations. This work enables reusable characterization as it is not tied to a fixed micro-architecture and configuration. We analyzed memory system behaviour over a range of cache sizes with and without prefetching. To simplify time-based analysis, we proposed aggregating the data by behaviour into MPKI bins which enables compact yet insightful representation of the time based data. With MPKI bins, we revealed that transient phases can have significant impact on cache performance of the overall execution.

References

[1] *SPEC CPU2017*, https://www.spec.org/cpu2017.

[2] *SPEC CPU2006*, https://www.spec.org/cpu2006.

[3] R. Panda, S. Song, J. Dean, and L. K. John, “Wait of a decade: Did SPEC CPU 2017 broaden the performance horizon?” in *IEEE International Symposium on High Performance Computer Architecture, HPCA 2018, Vienna, Austria, February 24-28, 2018*, 2018, pp. 271–282.

[4] A. Limaye and T. Adegbija, “A workload characterization of the SPEC CPU2017 benchmark suite,” in *IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS 2018, Belfast, United Kingdom, April 2-4, 2018*, 2018, pp. 149–158.

[5] S. Singh and M. Awasthi, “Memory centric characterization and analysis of spec cpu2017 suite,” in *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, ser. ICPE ’19. New York, NY, USA: ACM, 2019, pp. 285–292.

[6] S. Song, Q. Wu, S. Flolid, J. Dean, R. Panda, J. Deng, and L. K. John, “Experiments with spec cpu 2017: Similarity, balance, phase behavior and simpoints,” TR-180515-01, LCA Group, Department of Electrical and Computer Engineering, The University of Texas at Austin, Tech. Rep., May 2018.

[7] R. H. S. R and A. Milenkovic, “SPEC CPU2017: performance, event, and energy characterization on the core i7-8700k,” in *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering, ICPE 2019, Mumbai, India, April 7-11, 2019.*, 2019, pp. 111–118.

[8] A. Navarro-Torres, J. Alastruey-Benedé, P. Ibáñez-Marín, and V. Viñals-Yúfera, “Memory hierarchy characterization of spec cpu2006 and spec cpu2017 on the intel xeon skylake-sp,” *PLOS ONE*, vol. 14, no. 8, pp. 1–24, August 2019.

[9] *Pin - a dynamic binary instrumentation tool*, https://software.intel.com/en-us/articles/pin-a-dynamic-binary-instrumentation-tool, 2018.