How Much Does Memory Layout Impact Performance? A Wide Study

Augusto Born de Oliveira Jean-Christophe Petkovich
Sebastian Fischmeister
University of Waterloo
Waterloo, ON, Canada
{a3oliveira, j2petkov, sfischme}@uwaterloo.ca

Abstract

Memory layout variations are often cited as a threat to the reproducibility of computer performance experiments. Previous research has shown worst-case effects as large as 300% of execution time, among other surprising cases. While these are worrying results, how frequent and widespread are these large effects? To answer this question, this paper presents a wide-scale evaluation of memory layout effects on the performance of a large subset of the SPECCPU2006 benchmark suite on a wide array of diverse machines. We find that on average, these benchmarks are not as susceptible to memory layout effects as the worst-case analysis found in the literature suggests. Finally, we re-execute our own experiment and demonstrate why the reproducibility of an experiment's result *should not depend on its* data *being perfectly reproducible.*

1. Introduction

Several recent publications in the computer performance literature have reported on the dangers of discounting the effect of small memory layout variations on empirical performance results [1, 3, 4, 5, 13, 16]. Even small, seemingly innocuous, adjustments to a program's memory layout — e.g., the size of the POSIX environment — can have a profound effect. In some cases, the extreme worst case memory layout effects on performance can reach as high as 57% in the case of link-order, and 300% in the case of changes to the POSIX environment [4, 16]. We have independently confirmed [5] the impact of memory layout on performance observed by Mytkowicz *et al.* [16] using analogous experiment conditions, and others have observed similar effects in the wild [3, 13]. However, at the other extreme, a number of researchers report that performance effects as small as 2% occur in some benchmarks as a result of link-order manipulations [1, 16].

Changes to the memory layout of an application can manifest performance effects in several different ways. The size of the POSIX environment of a process influences its memory layout. Linux places environments at the beginning of the address space, the larger the environment, the further along the address space the code section and stack allocated data reside. As a direct result of this offset, cache and page misses may increase or decrease, influencing performance. Linux's Address Space Layout Randomization (ASLR) feature may also cause these effects to occur as a result of the process itself being mapped to different addresses.

While these factors are well known and the worst reported effects are worrisome, it is important for researchers to know if such large performance variations are guaranteed to affect their benchmarks, or if they are rare or specific to certain platforms. In this paper, we measure the performance effect of varying the memory layout of a wide range of applications and platforms, to determine how frequent and how large these effects are on average. We use two methods of varying the memory layout of an application, the POSIX environment size, and Linux's ASLR feature, on 11 different benchmarks in the SPECCPU2006 benchmark suite, on 17 unique hardware configurations. With replicates, this produces 2,244 individual trials for analysis. This provides us with a much broader sense of the impact on program performance of memory layout variations.

First, we investigate if performance effects are reproducible between benchmarks and machines, and second, we try to reproduce these effects on the same hardware later in time. We find both that memory has a significant effect on some benchmarks, but that the effect is small when present.

2. Related Work

Several researchers have reported observing statistically significant performance differences from memory layout manipulations. With observations of effects from layout related nuisance factors such as link-order, POSIX environment size, and ASLR.

Kalibera *et al.* [13] demonstrate that the random initial state of 6 different benchmarks had a significant impact on each benchmark's results. Kalibera concluded that benchmarks were likely to be influenced by their random initial state. Four of the benchmarks tested were Fast Fourier Transforms, it is possible that there is a greater incidence of sensitivity to memory layout in computation-heavy benchmarks.

Chen *et al.* [1] recently reported observing small, within 2%, variations in performance from different link-orders and POSIX environment sizes in their study of iterative optimization, a process highly dependent on reliable, reproducible and consistent results.

Curtsinger *et al.* [3, 4] developed a tool for randomizing the memory layout of code, stack and data memory regions of a program at runtime, allowing a researcher to control the effects of memory layout. The tool forces memory layout effects to approximate a Gaussian distribution, permitting the

use of traditional statistical analysis techniques. Curtsinger also reports performance regressions of up to 57% as a result of manipulating link-order.

We have reproduced the results found in [16] in our own reproduction case study [5] using very similar experimental conditions. We confirmed that not only did memory layout have a significant impact on performance, but that the optimal link-order was machine dependent.

Researchers have called for more reproducibility and statistical rigor in empirical computer science [2, 6, 7, 8, 17, 19], and recently groups such as the Evaluate Collaboratory [10] have come together to further pursue this goal. Complaints include low numbers of factors, lack of measurements of dispersion, improper or missing statistical analysis. In addition, the reporting practices in empirical computer performance research — key to reproduction of empirical results — has recently come into question. Tichy *et al.* [20] find substantial flaws in experiment design and execution in their survey of computer performance publications. Vitek and Kalibera [21] report that 39 of 42 papers published in PLDI'11 did not report a measure of dispersion with their data. Desprez *et al.* draw similar conclusions from their paper survey [9]. Our own literature survey [5] of over 200 recent publications from well reputed performance conferences found that less than 25% of the papers that included empirical performance data included measurements of dispersion, less than 2% performed any statistical analysis of their results, and less than 60% included any performance comparison for contextualizing results.

Despite these warnings and calls to action, we have shown [5] that even today only the vanishing minority of empirical computer science publications contain complete enough information to reproduce experimental results, or include a measure of dispersion, a requirement for statistically sound results in the presence of variability.

3. Memory Layout Experiments

To determine how frequent and how large memory layout effects are in the wild, we measure the effect of different memory layouts on the execution time of a wide range of benchmarks, on a wide range of machines. The benchmarks chosen are a subset of the SPECCPU2006 benchmark suite, shown in Table 1, selected for their widespread use in computer science [11]. We are interested in both increases and decreases of execution time, SPECCPU2006's native metric, since the effects of memory layout are not necessarily all detrimental. Table 2 shows the machines used. This large sample of heterogeneous machines and benchmarks will provide a larger sample size than previous research, painting a more comprehensive picture of these effects.

We force benchmarks to execute under different memory layouts by varying two factors: POSIX environment padding and Linux's ASLR feature. The environment size of a process affects its memory layout because Linux places environments at the beginning of the address space. The larger the environment, the farther out in the address space the code section will be, and depending on the increments of this offset, cache or paging-related performance effects may occur. Similarly, Linux's ASLR feature may cause these effects to appear due to the sections of a process being mapped at different addresses.

Benchmark	Area
Astar	Path-finding
Calculix	Structural Mechanics
Bzip2	Data Compression
GCC	Compiler
GobMK	A.I.
H264	Video Comp.
Lbm	Fluid Dynamics
Milc	Physics
Povray	Ray-tracing
Tonto	Chemistry
Wrf	Weather
Zeus	Physics

Table 1: Benchmark Set

ID	Processor	RAM
73	VIA Nano X2 1.6GHz	1.5GB
75	Pentium 4 CPU 3.20GHz	1GB
80	Core i7-2600K 3.40GHz	8GB
81	Opteron 8378 2.4GHz	32GB
88	Pentium M 1.70GHz	1GB
90	Pentium 4 3.20GHz	1GB
91	Pentium 4 2.40GHz	1GB
93	Pentium 4 3.40GHz	1GB
96	Pentium 4 3.20GHz	2GB
97	Pentium 4 3.00GHz	900MB
98	Pentium 4 3.00GHz	900MB
99	Pentium 4 2.80GHz	900MB
105	Pentium 4 3.20GHz	900MB
106	Pentium 4 3.20GHz	500MB
128	Pentium D 3.00GHz	2GB
130	Pentium 4 3.20GHz	900MB
135	Athlon 64 3500+	2GB

Table 2: Machine Set

The experiment follows a two-level full-factorial design [15], where both factors have two levels (or settings) and are explored concurrently (i.e., all possible combinations of factors are explored). The environment paddings used are zero bytes or 10,928 bytes, and ASLR is set to "on" or "off". Each factor combination is executed three times on each machine to allow for a measure of variance, leading to a total of twelve ($2 \times 2 \times 3$) jobs per machine, per benchmark, totaling 2,244 individual jobs. To collect this large amount of data, we use DataMill, a performance evaluation infrastructure further discussed in Section 5.

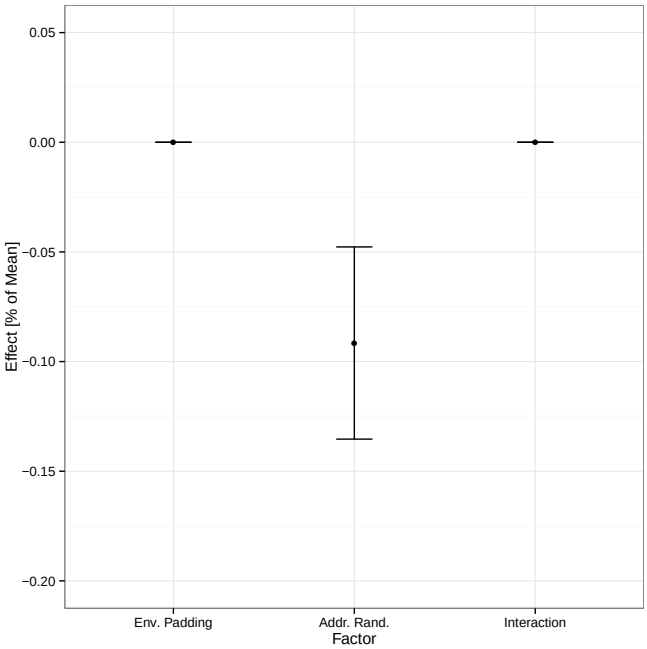

Figure 1: Memory Effects on Bzip2 on Machine 128

3.1. Analysis of Variance

To determine the effect of the two memory layout factors on the execution time of the benchmarks, we use analysis of variance (ANOVA). ANOVA is a statistical method that partitions the variability of a data set into its component parts; in our case, the variability caused by the environment padding factor, the variability caused by the ASLR factor, the interaction between these two factors, and unexplained variability caused by error (i.e., uncontrolled sources of variability).

Figure 1 shows the effect size for the two factors and their interaction on Bzip2 when executed on machine 128. The x-axis shows the source of variability, and the y-axis shows the size of the effect as a percentage of the mean performance, with error bars showing one standard error around the effect estimate. For example, an effect of +10% means that, on this machine, this benchmark's execution time will increase by 10% of the average when this factor is "turned on". In the case of Figure 1, the effect of both factors is less than one tenth of one percent of the mean, even though address randomization effect is statistically significant with $P < 0.01$ (the environment padding effect and the interaction between the two effects are not statistically significant).

An effect must be statistically significant to be distinguishable from noise. More pragmatically, an effect must also be large enough to be of importance; for example, a statistically significant effect of 0.1% of the mean is not large enough to impact the majority of applications. The remainder of this section elaborates on this distinction.

3.2. Results

Figure 2 shows the relative effect sizes for each factor and their interaction for each benchmark, with data from all machines grouped in a single box plot. The three horizontal lines in each box represent the upper quartile, the median, and the lower quartile. The lower and upper whiskers extend to the last value within 1.5 times the interquartile range of the lower and upper quartile, respectively. Dots represent outliers that are outside this range. If a benchmark had a systematic sensitivity to either factor or their interaction, one of these plots would show a significant deviation from zero. As the plots show, however, the most significant deviation is still well under 3%.

Figure 3 shows the relative effect sizes for each factor and their interaction for each machine, with data from all benchmarks grouped in a single box plot. Similarly to Figure 2, this plot would reveal a machine's systematic sensitivity to either factor or their interaction, but, again, no such significant sensitivity exists.

The previous figures show that, from a bird's eye view, our memory-layout-related effects appear negligible. Figure 4 shows the absolute worst case observed in our experiments, Astar on machine 81. The x-axis shows the factor name, and the y-axis shows the *absolute* magnitude of the effect. On this machine-benchmark pair, the ASLR factor had an effect estimate of -22.06s, while the mean execution time of that benchmark on that machine was 787.3s (a relative effect of -2.8%). Note that the error bars straddle the zero axis, and therefore these effects are not statistically significant. Most importantly, even if we were to ignore statistical significance (running the risk of treating noise as a real effect), this effect is not *practically* significant, and therefore, not even our worst single observed case is cause for concern.

Given the breadth of machines and benchmarks explored here, we feel confident in saying that memory effects, as controlled by the two factors we investigated, are not significant enough to warrant widespread concern. Exceptional cases, where these effects are more pronounced, or where a sub-3% performance difference is crucial may occur, but for the great majority of computer performance researchers these effects appear negligible.

4. Reproduction

While the goal of statistical methods such as ANOVA is to ensure reproducible results, we have repeated the set of experiments described in Section 3.2 one week later to evaluate how reliable those measurements were, and how easily the results could be reproduced at a later date.

Table 3 presents a comparison between the original experiment and the reproduction experiment. The numbers presented for mean and maximum effects are based on the absolute values of the effects, such that negative effects do not cancel out positive ones. Due to hardware malfunctions, only 13 of the original 17 machines were available for the reproduction,

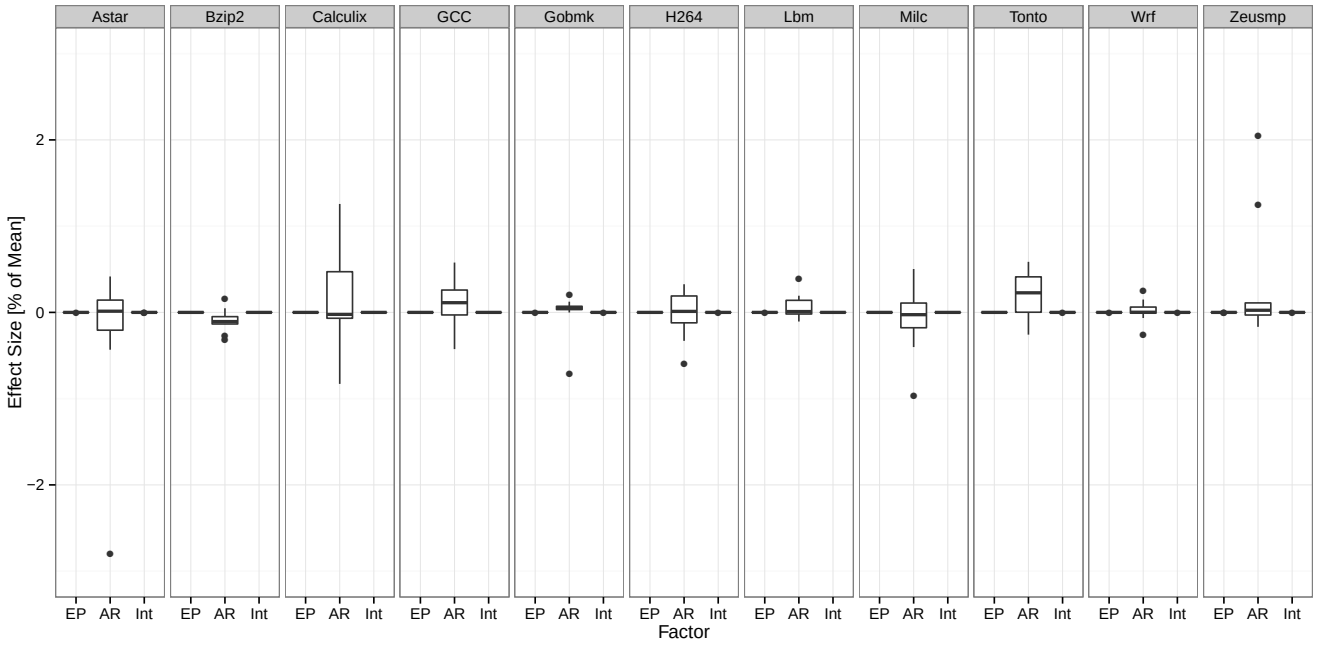

Figure 2: Memory Effects by Benchmark

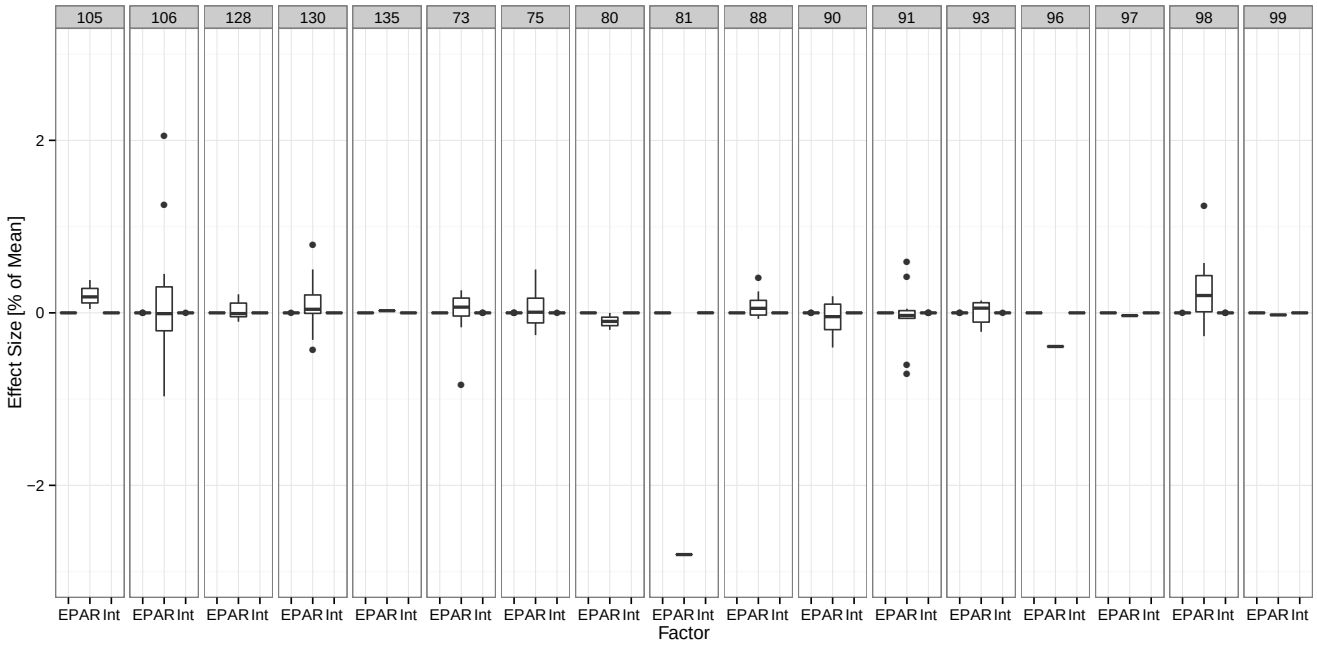

Figure 3: Memory Effects by Machine

and only 9 of the original 11 applications ran to completion for the full duration of the experiment. As seen on the table, the mean estimates of the reproduction were similar to the original experiment (that is to say, effects were negligible), but the single worst case effect of the address randomization factor grew significantly. However, much like the 2.8% worst case effect of the original experiment, the 5.94% effect of the reproduction experiment is also statistically insignificant, and therefore indistinguishable from noise. The equivalent plots to Figures 2 and 3 are suppressed due to space constraints, but are largely indistinguishable from the original plots.

Although precisely the same conditions were not recreated as a result of a power surge that led to hardware failures, we were able to reproduce the *result* from the first experiment. This is due to the built-in reproduction of results on a wide array of machines and benchmarks. This broader strategy of

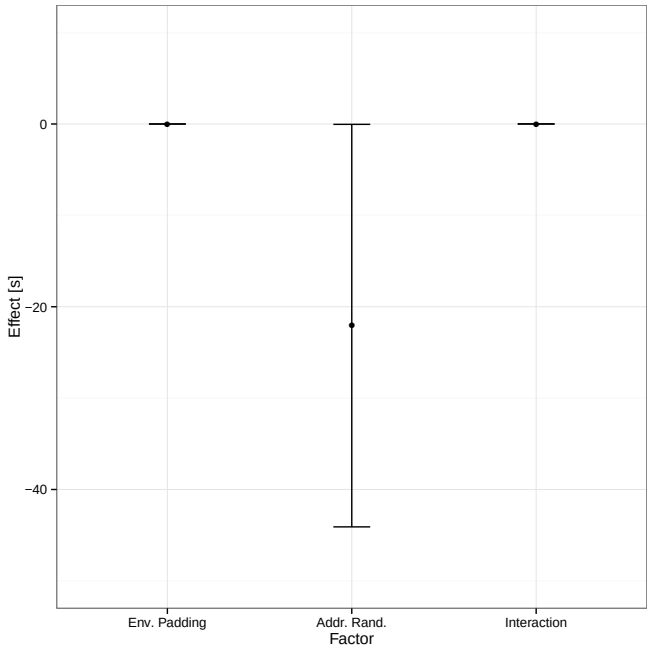

Figure 4: Worst Case Addr. Rand. Effect on Astar on Machine 81

	Original	Reproduction
Mean Env. Pad.	0.000032%	0.000045%
Max. Env. Pad.	0.00035%	0.00031%
Mean Addr. Rand.	0.23%	0.42%
Max. Addr. Rand.	2.8%	5.94%
Machines	17	13
Applications	11	9

Table 3: Reproduction Summary

reproduction is more desirable than simply re-running experiments on a single platform, since results derived from it are more likely to be reproducible by other researchers on other platforms.

5. Discussion and Lessons Learned

Generality of claims vs individual results Our work confirms findings and arguments reported by others regarding the generality of claims. The large set of diverse nuisance factors that influence computer performance experimentation forces us to be extremely careful when reporting that our results hold in the general case [12, 13, 14, 15, 16]. Even in well designed experiments that use multiple machines, and many experimental conditions such as in [3, 13, 16], results and effects can be observed that pertain only to the set of conditions tested.

Reproduction of data vs reproduction of results Given the fast pace of computer technology, reproducing data on a single platform only goes so far. By reproducing an effect on various platforms and under various operating conditions, researchers improve the generality of their results. We cannot justify using small numbers of experimental conditions for comparing performance. The fact that our results greatly differ from those found by others in the literature only further confirms that we must use more sets of experimental conditions.

In [16], Mytkowicz *et al.* find not only that link-order and POSIX environment size have a significant effect on performance, but that they can cause very dramatic regressions or speedups in performance (57% and 300% worst cases respectively). The paper concludes saying that we must be careful not to run experiments against only a small set of conditions, and that we need to be cautious of the impact of small variations in our experimental environment, as they can be deceptively influential.

Although our results point towards much lower worst case memory layout effects — 2.8% and 5.94% worst case effects for the original and reproduction experiments respectively — this does not contradict data found by other researchers, nor does it contradict their results. Instead, our data improves the generality of our understanding of memory layout performance effects. We observe smaller effects in different benchmarks, and on different hardware configurations. Our set of conditions is different from those of others in the literature, so we can expect our data to differ as well.

The process of reproducing the *data* found in a study allows researchers to ensure the experiment is calibrated appropriately, and to work out issues in the experimental setup, as in [5]. Once the experiment achieves the results previously observed in the original study, performing sanity checks by varying factors that purportedly should not affect the results, as we did in this paper, is a very useful and enlightening process.

Reliability of diverse systems In our attempt to reproduce our results at a later date, we found that several of the machines used in the previous experiments had ceased to function as the result of a power surge. Experiences like this highlight the importance of reproducible *results* of a paper as opposed to reproducible *data*. Although we were unable to replicate the precise conditions of the experiment due to our machine losses, we *were* able to reproduce the *results*.

It is because we following advice and guidelines similar to those in [16], to using a diverse set of experimental conditions for experimentation, that we were able to successfully reproduce our results.

Determining sensitivity to memory layout effects Despite the fact that our results point to a lack of sensitivity to memory layout performance effects in the general case, the fact remains that a given experiment could be susceptible to them. How can we determine whether or not memory layout effects will be a concern for a given benchmark-machine pair? At current, there is no simple answer other than to perform exploratory experimentation.

A common idiom in clinical drug trials — another science that must deal with many interacting nuisance factors — is to perform so-called pilot trials, where a drug is tested in a

limited size and scope to roughly estimate the drug's effectiveness and sensitivity to factors. The results of the pilot trial, along with expert knowledge, are used to guide the design and prioritization of factor exploration of the more in-depth experiment, the pivotal trial.

From a small set of randomized conditions, as in a pilot trial, a computer science researcher may be able to identify sensitivity to well-known nuisance factors and prioritize the factors considered in their final experiments accordingly.

Mass experimentation with diverse hardware All of the experimental trials were conducted using DataMill [18, 5], a community-driven infrastructure for computer performance evaluation.

DataMill offers a wide variety of diverse machines for experiment execution in a clean-room environment, ensuring repeatability and consistent results across machines and simplified trial replication. The pertinent details of the hardware and software used during the course of the experiment are also automatically recorded and reported by the infrastructure to contextualize results. This combination of features placed DataMill in a good position to explore the effects of memory layout in a much broader set of experiment conditions than previously realized.

6. Conclusion

This paper presented a wide-scale evaluation of memory effects on benchmark performance, using a large subset of SPEC-CPU2006 on a wide array of machines. The results show that, on average, these benchmarks are not as susceptible to memory layout effects as the worst-case analysis found in the literature suggests, and, therefore, these effects should not be a priority when controlling threats to experiment reproducibility.

Our results also clearly show that statistical significance, while important for reproducibility, is secondary to "practical significance". While many benchmark-machine pairs showed statistically significant memory-layout-related effects, these were almost entirely negligible for the average researcher.

Finally, when attempting to reproduce our own results at a later date, we found it difficult to execute the exact same jobs on the exact same machines at such a large scale due to hardware failures. This makes it all the more important that the *results* of a paper are reproducible even though the *data* may not be. A conclusion drawn from a set of machines will not depend on data from any single one of them, and is more likely to apply in the general case.

References

[1] Yang Chen, Yuanjie Huang, Lieven Eeckhout, Grigori Fursin, Liang Peng, Olivier Temam, and Chengyong Wu, "Evaluating iterative optimization across 1000 datasets," in *Proceedings of the 2010 ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '10. New York, NY, USA: ACM, 2010, p. 448–459. Available: http://doi.acm.org.proxy.lib.uwaterloo.ca/10.1145/1806596.1806647

[2] D. E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young, "Computing as a Discipline," *ACM Communications*, vol. 32, no. 1, pp. 9–23, Jan. 1989. Available: http://doi.acm.org/10.1145/63238.63239

[3] Charlie Curtsinger and Emery Berger, "Stabilizer: Enabling statistically rigorous performance evaluation," University of Massachusetts, Tech. Rep., 2012.

[4] Charlie Curtsinger and Emery D. Berger, "STABILIZER: statistically sound performance evaluation," in *Proceedings of the Eighteenth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '13. New York, NY, USA: ACM, 2013, p. 219–228. Available: http://doi.acm.org.proxy.lib.uwaterloo.ca/10.1145/2451116.2451141

[5] Augusto Born de Oliveira, Jean-Christophe Petkovich, Thomas Reidemeister, and Sebastian Fischmeister, "DataMill: rigorous performance evaluation made easy," in *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '13. New York, NY, USA: ACM, 2013, p. 137–148. Available: http://doi.acm.org.proxy.lib.uwaterloo.ca/10.1145/2479871.2479892

[6] Peter J. Denning, "ACM President's Letter: What is Experimental Computer Science?" *ACM Communications*, vol. 23, no. 10, pp. 543–544, Oct. 1980. Available: http://doi.acm.org/10.1145/359015.359016

[7] Peter J. Denning, "ACM President's Letter: Performance Analysis: Experimental Computer Science as its Best," *ACM Communications*, vol. 24, no. 11, pp. 725–727, Nov. 1981. Available: http://doi.acm.org/10.1145/358790.358791

[8] Peter J. Denning, "Is Computer Science Science?" *ACM Communications*, vol. 48, no. 4, pp. 27–31, Apr. 2005. Available: http://doi.acm.org/10.1145/1053291.1053309

[9] F. Desprez, G. Fox, E. Jeannot, K. Keahey, M. Kozuch, D. Margery, P. Neyron, L. Nussbaum, C. Perez, O. Richard, W. Smith, G. von Laszewski, and J. Voeckler, "Supporting Experimental Computer Science," Argonne National Laboratory Technical Memo, Tech. Rep., 2012.

[10] Evaluate Collaboratory, "Experimental Evaluation of Software and Systems in Computer Science," http://evaluate.inf.usi.ch, accessed Jan. 4th, 2014.

[11] John L. Henning, "Spec cpu2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep. 2006. Available: http://doi.acm.org/10.1145/1186736.1186737

[12] Raj Jain, *The Art of Computer Systems Performance Analysis*, ser. Wiley professional computing. Wiley, 1991.

[13] Tomas Kalibera, Lubomir Bulej, and Petr Tuma, "Benchmark precision and random initial state," in *Proceedings of the 2005 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2005)*, 2005, p. 484–490. Available: http://d3s.mff.cuni.cz/publications/download/KaliberaBulejTuma-BenchmarkPrecision.pdf

[14] D.J. Lilja, *Measuring Computer Performance: A Practitioner's Guide*. Cambridge University Press, 2005. Available: http://books.google.ca/books?id=R8RLniX5DNQC

[15] D.C. Montgomery, *Design and Analysis of Experiments*. John Wiley & Sons, 2008.

[16] Todd Mytkowicz, Amer Diwan, Matthias Hauswirth, and Peter F. Sweeney, "Producing Wrong Data Without Doing Anything Obviously Wrong!" *SIGPLAN Notes*, vol. 44, no. 3, pp. 265–276, Mar. 2009. Available: http://doi.acm.org/10.1145/1508284.1508275

[17] Larry Peterson and Vivek S. Pai, "Experience-Driven Experimental Systems Research," *ACM Communications*, vol. 50, no. 11, pp. 38–44, 2007. Available: http://doi.acm.org/10.1145/1297797.1297820

[18] The DataMill Team, "DataMill: Rigorous performance evaluation made easy," https://datamill.uwaterloo.ca/ visited 2014-01-04.

[19] Walter F. Tichy, "Should Computer Scientists Experiment More?" *IEEE Computer*, vol. 31, no. 5, pp. 32–40, 1998.

[20] Walter F. Tichy, Paul Lukowicz, Lutz Prechelt, and Ernst A. Heinz, "Experimental Evaluation in Computer Science: A Quantitative Study," *Systems Software*, vol. 28, pp. 9–18, 1995.

[21] Jan Vitek and Tomas Kalibera, "Repeatability, Reproducibility, and Rigor in Systems Research," in *Proceedings of The Ninth ACM International Conference on Embedded Software*, ser. EMSOFT '11. New York, NY, USA: ACM, 2011, pp. 33–38. Available: http://doi.acm.org/10.1145/2038642.2038650