Experimental evidence of chaotic dynamics in computer hardware

Todd Mytkowicz, Elizabeth Bradley, and Amer Diwan

University of Colorado at Boulder
Technical Report CU-CS-1031-07
June 2007

Deptartment of Computer Science
Campus Box 430
University of Colorado
Boulder, Colorado 80309-0430

In review, Physical Review Letters

Experimental evidence of chaotic dynamics in computer hardware

Todd Mytkowicz,* Elizabeth Bradley, and Amer Diwan
Department of Computer Science
University of Colorado
Boulder, Colorado

In this Letter, we demonstrate the first experimental evidence of chaotic dynamics in real computer hardware. We use delay-coordinate embedding to reconstruct the state-space dynamics of a modern microprocessor while it executes a standard benchmark program. Analyzing these dynamics, we establish multiple corroborating measures of low-dimensional chaos. Our results indicate that traditional nonlinear analysis techniques are an elegant and effective solution to understanding the dynamics of computer performance.

The growing complexity of computer architecture has overwhelmed the analysis techniques used in that field. Traditional models of computer performance, for instance, essentially ignore the temporal details of behavior. Instead, hardware architects generally use aggregate measures like FLOPS (floating point operations per second, calculated over an entire program run) as measures of performance. In the past, when computers were simpler devices, measures like this may have been adequate for understanding performance. With modern hardware and software becoming so complex, however, even simple design changes can cause unintended and counterintuitive effects. Aggregate measures, such as FLOPS, are not detailed enough to elucidate the dynamics that underlie these effects. At the same time, the act of making high fidelity measurements can perturb the dynamics of the very system that we want to understand. These are exactly the kinds of challenges faced in physics problems, and the underlying premise of this Letter is that the ideas and tools developed by the physics community are an effective way to approach the analysis of modern computer systems.

Though not a common view in the microprocessor literature[16], a computer is clearly a high-dimensional nonlinear dynamical system. Viewed in this light, microprocessor registers, main memory contents and even regional microprocessor temperature are state variables of the system. The logic hardwired into the microprocessor, combined with the software executing on that hardware, defines the system dynamics. Under the influence of these dynamics, the computer's state moves on a trajectory through this high-dimensional space as the clock cycles progress and the program executes.

Taking this view of computers as dynamical systems, we use traditional nonlinear time-series techniques to analyze them. We measure how many instructions are executed per clock cycle from a standard benchmark program. We use a nonlinear noise reduction technique to filter our data, use delay-coordinate embedding to reconstruct the state-space dynamics and then compute several standard invariants on this reconstructed dynamics. The results are the first experimental evidence of chaos on real computer hardware.

Measuring the behavior of a running computer without perturbing its behavior is surprisingly difficult. Ideally, one would inspect the contents of every microprocessor register and main memory location on a cycle-by-cycle basis, thereby directly measuring a large subset of the system's state variables. This is completely impractical, of course. The hardware does not expose all of these variables; even if it did, this level of measurement intervention would perturb the very program we were trying to understand. For example, simply dumping the main memory of a 1GB machine running at 2.4 gigahertz once every 100,000 cycles would produce produce a terabyte of data every 40 milliseconds. This process would not only dominate the computer's dynamics, but also overwhelm any data analysis tool.

For these reasons, performance analysts measure only a few internal variables using a hardware performance monitoring facility (HPM). HPMs enable analysts to measure many aspects of a computer's performance at a reasonable cost. The HPM facilities typically contain two to eight dedicated registers, each of which can count a different, user-programmed event. Using these registers, one can capture the total number of instructions executed per cycle (IPC), for instance, or the total number of data references that miss in high level caches (i.e. L1 or L2). These are the most widely used metrics in the computer performance analysis literature. IPC, in particular, is a good way to compare the performance of modern microprocessors, most of which can execute more than one instruction per clock cycle.

The HPM facility maintains a running count of events. To capture that information and save it for later use, we wrote a monitoring tool that periodically stops a running program, reads the current values in the HPMs, and stores them to disk. Because the HPM registers are in hardware, the counting of events does not perturb the running program. Storing these counters in memory, or on disk, *can* affect shared hardware structures, such as the memory caches and thus introduce noise into our measures. To avoid this insofar as possible, our tool only monitors hardware events when the target program is running, and not when the operating system (or the monitoring tool itself) have control of the microprocessor.

Using this custom measurement infrastructure, we recorded the performance of a single program from the SPEC2000 CPU suite[1]. This suite is a standardized collection of benchmark programs, assembled from real-world applications by a group composed of both academic and industry officials. There are many benchmarks used in the computer architecture field, however the SPEC2000 CPU suite is the most broadly used

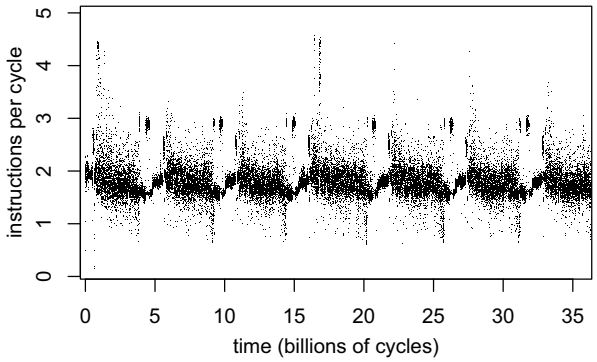

FIG. 1: The first 35 billion (of 250) cycles of IPC (filtered) versus time for the POWER4+ microprocessor running the `bzip2` benchmark program. The pattern seen here essentially repeats for the extent of the entire run.

and our use of it facilitates comparison with prior work.

Given measurements of a single state variable—IPC every 100K cycles from a POWER4+ microprocessor, for instance—one can use delay-coordinate embedding[2] to reconstruct the dynamics of the system. The Takens-Whitney-Mane theorems guarantee that such an embedding, if done correctly, is topologically equivalent to the underlying dynamics. Because dynamical invariants like the largest Lyapunov exponent are invariant under diffeomorphism, one can calculate the invariants of the embedded dynamics and extrapolate the results to the true dynamics. The balance of this Letter describes the results of applying these ideas to modern computer systems. Of course, IPC is most likely not a state variable, but rather a complex nonlinear function of multiple state variables. Even so, we believe it is appropriate to use IPC for delay-coordinate embedding for two reasons. First, many of our dynamical invariants corroborate each other. Second, we repeated all our analysis with a different measure, total L1 data cache misses per cycle, and found almost identical values for all measured dynamical invariants.

We are not the first to use nonlinear time-series techniques on computer performance measures. Berry *et al*.[3] measured a few salient metrics from several of the SPEC2000 benchmarks running on a hardware simulator implemented in code, embedded the data, and then computed dynamical invariants using the TISEAN[4] tool-set. Their results show strong indications of chaotic dynamics for some programs (including `bzip2`, which we use here). The other programs in their study were found to either be from a periodic process or the result of some high-dimensional non-chaotic process.

While a novel application of nonlinear time-series analysis, the methods of Berry *et al.* leave open one essential problem: the match between simulation and reality. In computer architecture, even the most detailed and "validated" hardware simulators have long known to be inaccurate models of the actual machines they are built to emulate[5]. Systems in this field are so complicated that researchers commonly sacrifice simulator accuracy for simulation speed and lack of measurement perturbation. (Indeed, Berry *et al.* cite this as one reason for working with a simulator.) In order to obtain a true picture of the dynamics, we ran our experiments on real computer hardware—not software simulators of that hardware–and dealt with the noise and measurement issues that come along with that endeavor.

A variety of sources of noise can possibly infiltrate the time-series data in our experiment. As mentioned earlier, no measurement methodology can avoid perturbing the program that it monitors. Other programs that are running on the computer, such as the operating system, can affect the data, as can environmental variables like temperature. For these reasons, we filtered the `bzip2` IPC data before embedding it. Linear filters, which assume noise exists in higher frequencies of the Fourier spectrum, are known to be problematic with chaotic data[6]. Instead, we used a nonlinear noise reduction routine due to Grassberger *et al.* that assumes that the data lie on a low-dimensional manifold[7]. This technique has been shown to be fairly robust with regards to parameter choices[8], even when used on noisy, experimental data, as long as one does not over-iterate the algorithm[9]. We are confident that this is not a problem in our application of this technique, as we have only used a single iteration of locally linear projection.

According to Theiler[6], one should exclude temporally correlated points from the estimation of dynamical invariants like λ_1 and D_2. In the calculations reported here, we varied the Theiler window, up to 200 million cycles, and found that this parameter did not affect our results. A natural way to think about program structure is a composition of loops and conditional branches. As a program executes, it bounces to different sections of its code when a branch is taken. While our IPC data are continuous, this branching dynamics gives the embedded trajectory some map-like behavior, in which temporally correlated points jump large distances in the reconstructed state space. For this reason, the Theiler correction window was not a necessary element of our analysis.

In order to investigate the dynamics of computer performance, and facilitate easy comparison with prior work, we focused on a single program—`bzip2` from the SPEC2000 suite—and ran it on an IBM 1.2GHz POWER4+ processor. We compiled `bzip2` with `gcc` version 3.2, ran the program with the SPEC2000 CPU `input.source` data set as input, and gathered data using the monitoring facility described above. All experiments were run on the Linux operating system and analyzed with the TISEAN tool set[4]

We first filtered the data using Grassberger's method, as implemented in TISEAN's `ghkss` tool. Figure 1 shows the first 35 billion cycles of filtered IPC versus time from the `bzip2` program running on the POWER4+ microprocessor. We only show a prefix of the time-series for readability; the total execution of the program takes roughly 250 billion cycles and essentially repeats this same pattern. This periodicity reflects the internal loop structure of the `bzip2` code[17]. Collapsing this trace into aggregate metrics—average IPC over the entire run, for instance—would not capture, among other things, this

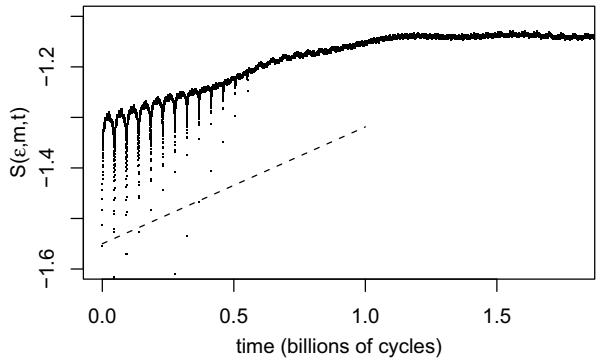

FIG. 2: Average divergence ($\log_{10}$) of neighboring points in the reconstructed state-space as a function of time. A characteristic linear regime, the slope of which is an estimate of the largest Lyapunov exponent, is shown for the POWER4+ micro-architecture

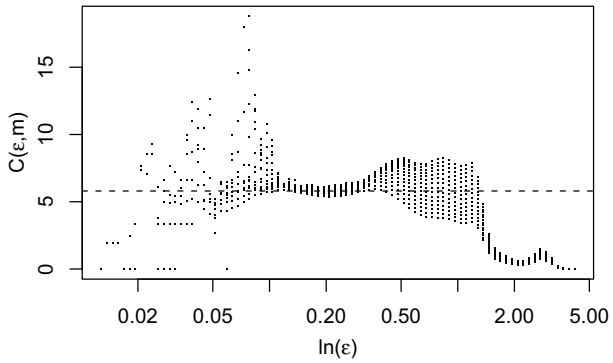

FIG. 3: Local slopes of the correlation sum as a function of neighborhood size for $11 \leq m \leq 25$. A characteristic linear regime exists in the range of $\ln(0.1) \leq \ln(\varepsilon) \leq \ln(0.5)$. We estimate $D_2 = 5.6 \pm 0.01$ (horizontal line) for the POWER4+ micro-architecture. This argues for determinism at work, rather than a random process, and hints at a fractal state-space geometry.

obvious feature of the data. Taking a physics-based dynamical systems approach to this problem *does* capture this feature.

Our dynamical systems approach uses delay-coordinate embedding to reconstruct the state-space trajectory from the filtered POWER4+ IPC time series. We followed standard procedures to choose appropriate embedding parameters: the first minimum of the mutual information curve[10] as an estimate of the delay τ and the false-nearest neighbors technique of Kennel *et al.*[11] to estimate the embedding dimension m. For the POWER4+ IPC data, the resulting values, obtained using TISEAN's `mutual` and `false_nearest` tools, were $\tau = 460$ and $m = 13$, respectively.

The Lyapunov exponent λ is a measure of the sensitivity of the dynamics to perturbations; a positive largest λ_1 is commonly taken as an indication of chaos in the system. We used the algorithm of Rosenstein *et al.*[12] to estimate λ_1 from the POWER4+ time-series, embedded using $\tau = 460$ and $m = 13$. Figure 2 shows the average divergence of neighboring points in the reconstructed state space as a function of time, computed using TISEAN's `lyap_r` tool. This graph shows a linear regime from $0 \leq time \leq 1$ billion cycles, and then saturates because the spread between the points tracked by the algorithm reaches the diameter of the data. The linear region indicates exponential divergence in the reconstructed dynamics. We used linear regression to fit a line in this region and calculated $\lambda_1 = 0.179 \pm 0.0006$ per billion cycles. We repeated this analysis with $m > 13$ and obtained identical results, which both corroborates the `false_nearest` result and strengthens our belief that the dynamics of `bzip2` on the POWER4+ microprocessor exhibit sensitive dependence on initial conditions. These dynamics were not desired or anticipated by the engineers who designed this computer system, nor can their analysis tools elucidate it; a physics-based approach, however, brings out this interesting and important dynamical property clearly.

We then calculated the correlation sum[13] of the embedded dynamics using the TISEAN tool `d2`. Figure 3 shows the local slopes of the correlation sum from $12 \leq m \leq 25$ as a function of the natural log of the neighborhood size ε. Clearly there is a linear scaling region in the range $\ln(0.1) \leq \ln(\varepsilon) \leq \ln(0.5)$. The slope of a linear fit in this range, shown as the line in the plot, yields an estimate of the correlation dimension $D_2 = 5.6 \pm 0.01$. This argues for determinism at work, rather than a random process, and hints at the fractal state-space geometry that is seen in the state-space dynamics of many chaotic systems.

Lastly, we calculated the correlation entropy, which is a lower bound on *all* the positive Lyapunov exponents[14] and thus a good check on our estimate of $\lambda_1 = 0.179$. Figure 4 shows the estimate of the correlation entropy as a function of $\ln(\varepsilon)$. A linear scaling region exists in the same range ($\ln(0.1) \leq \ln(\varepsilon) \leq \ln(0.5)$) as in the correlation dimension estimate. A linear fit to this region provides an estimate of $h_2 \approx 0.35$. This value is a barometer by which we can gauge our estimation of λ_1. For our data, $h_2 \geq \lambda_1$ which reflects the fact that there may be more than one positive Lyapunov exponent and strengthens our estimation of both dynamical invariants.

Using nonlinear techniques to diagnose the dynamics of any experimental process should be done with extreme caution. Many of the associated algorithms are highly sensitive to noise, data quantity, and their own parameters, and all of their results require expert human interpretation. To truly trust the output of any of these methods, one seeks corroboration with different algorithms, much like we just described with the correlation entropy and the largest Lyapunov exponent.

In our results, the various dimensions corroborate quite nicely. The original embedding theorems require $m \geq 2d$, where d is the dimension of the underlying dynamics, but other, tighter bounds have been established since then (e.g., $m \geq 2D_A$, the box-counting dimension[15]). The false-near neighbor algorithm—which is highly pessimistic—produced

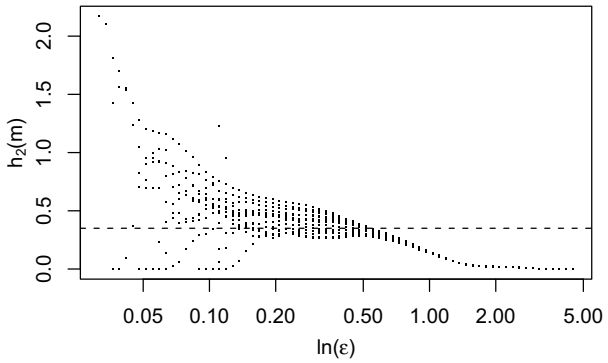

FIG. 4: Correlation entropy as a function of neighborhood size for $11 \leq m \leq 25$. In the range of $\ln(0.1) \leq \ln(\varepsilon) \leq \ln(0.5)$ for values of m, the estimate of the correlation entropy collapses to a scaling region. We estimate the correlation entropy to be $h_2 \approx 0.35$ (horizontal line).

an estimate of $m = 13$ for `bzip2` IPC dynamics on the POWER4+ architecture, while the arguably tighter `ghkss` algorithm suggested that $m = 5$ was adequate and the correlation dimension was 5.6 ± 0.01.

This surprisingly consistent cohort of numbers is a strong indication of low-dimensional dynamics, which is somewhat surprising in a system as complex as a modern microprocessor. All modern computer hardware, however, uses an instruction set architecture (ISA), and high level programming languages are compiled down to this ISA. The ISA for the POWER4+ machine provides 32 general-purpose registers, only a few of which are used extensively during a program's execution. With this in mind, low-dimensional dynamics in computer systems make more sense.

In this Letter, we have shown strong indications—from multiple corroborating methods—of low-dimensional chaotic dynamics in `bzip2` traces on the POWER4+ micro-architecture. By using standard methods from the physics literature on this engineered system, we were able to keep measurements to a minimum, thereby limiting perturbation caused by a measurement infrastructure while still providing salient analysis that illuminate the actual dynamics of the microprocessor.

This has two important implications for the computer architecture community, which employs software simulation in lieu of real hardware prototyping and uses aggregate measures of performance, thereby ignoring the time-varying aspects of the behavior. First, the temporal complexity of the dynamics that we observe defies analysis by means of aggregate measures, so performance analysts should not rely on those kinds of metrics. Instead, computer architects can use the techniques of nonlinear dynamics, which were developed specifically to elucidate the kind of time-varying behavior that is so important in computer analysis and design. And second, the dynamics of computer systems are so sensitive to initial conditions, the common practices of testing out one's ideas on a model of the architecture—a hardware simulator implemented in code—should only be reserved for cases when one is sure that the dynamics of the model simulator closely match the target hardware. A physics based approach to understanding the performance of computer systems provides the architect a rich set of methods with which she can truly understand the dynamics of today's modern hardware.

* Electronic address: `Todd.Mytkowicz@colorado.edu`

[1] J. L. Henning, Computer **33**, 28 (2000), ISSN 0018-9162.
[2] F. Takens, in *Dynamical Systems and Turbulence*, edited by D. Rand and L.-S. Young (Springer, Berlin, 1981), pp. 366–381.
[3] H. Berry, D. G. Perez, and O. Temam, Chaos: An Interdisciplinary Journal of Nonlinear Science **16**, 013110 (pages 15) (2006).
[4] R. Hegger, H. Kantz, and T. Schreiber, Chaos: An Interdisciplinary Journal of Nonlinear Science **9**, 413 (1999).
[5] J. Gibson, R. Kunz, D. Ofelt, M. Horowitz, J. Hennessy, and M. Heinrich, SIGPLAN Not. **35**, 49 (2000), ISSN 0362-1340.
[6] J. Theiler and S. Eubank, Chaos **3**, 771 (1993).
[7] P. Grassberger, R. Hegger, H. Kantz, C. Schaffrath, and T. Schreiber, Chaos: An Interdisciplinary Journal of Nonlinear Science **3**, 127 (1993).
[8] H. Kantz, T. Schreiber, I. Hoffmann, T. Buzug, G. Pfister, L. G. Flepp, J. Simonet, R. Badii, and E. Brun, Phys. Rev. E **48**, 1529 (1993).
[9] A. I. Mees and K. Judd, Physica D Nonlinear Phenomena **68**, 427 (1993).
[10] A. M. Fraser and H. L. Swinney, Phys. Rev. A **33**, 1134 (1986).
[11] M. B. Kennel, R. Brown, and H. D. I. Abarbanel, Phys. Rev. A **45**, 3403 (1992).
[12] M. T. Rosenstein, J. J. Collins, and C. J. D. Luca, Physica D **65**, 117 (1993).
[13] P. Grassberger and I. Procaccia, Physica D Nonlinear Phenomena **9**, 189 (1983).
[14] Y. B. Pesin, Russian Math. Surveys **33** (1977).
[15] T. Sauer, J. A. Yorke, and M. Casdagli, Journal of Statistical Physics **65**, 579 (1991).
[16] Since 2004, 364 papers about understanding the performance of microprocessor innovations have been published in the top three micro-architecture conferences (ASPLOS, ISCA and MICRO), only nine of which looked at the time-varying behavior of real hardware.
[17] `bzip2` is a data compression utility that uses several encoding schemes