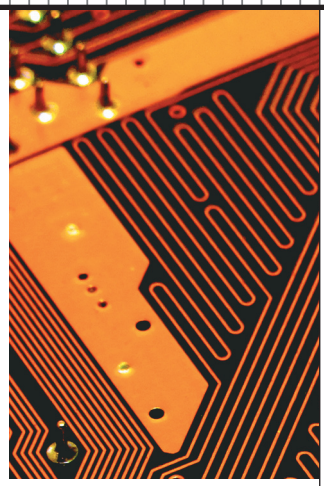
# A Methodology for Analyzing Commercial Processor Performance Numbers



Kenneth Hoste and Lieven Eeckhout, *Ghent University*

**The wealth of performance numbers provided by benchmarking corporations makes it difficult to detect trends across commercial machines. A proposed methodology, based on statistical data analysis, simplifies exploration of these machines' large datasets.**

Benchmarking consortia and corporations publish performance numbers on commercial computer systems for a set of industry-standard benchmarks. For example, the Standard Performance Evaluation Corporation (SPEC; www.spec.org) provides performance results for various benchmarks from application domains such as compute-intensive workloads, Java workloads, graphics, Web servers, mail servers, and network file systems. The information obtained from these benchmarking experiments is valuable for comparing commercial machines across applications, manufacturers, and processor generations for different types of workload behaviors. However, the abundance of data makes analysis difficult.

To gain insight from these large datasets, we developed a performance analysis methodology and framework using principal components analysis (PCA),[1] which reduces the dataset's dimensionality. To illustrate the power of the proposed performance analysis methodology, we used it to analyze and expose trends in a SPEC CPU2000 dataset consisting of performance numbers for 26 benchmarks and more than 1,000 machines. We then served the output of the statistical analysis into the processor performance visualizer, an interactive visualization tool that yields quick and intuitive navigation through large performance datasets.

## DATA COLLECTION

Our dataset includes the performance numbers reported on the SPEC CPU website (www.spec.org/cpu2000/results) for the SPEC CPU2000 benchmark suite. As Table 1 shows, we use the speedup ratios with base optimization—that is, SPECint base2000 (integer) and SPECfp base2000 (floating-point)—for numerous machines with different architectures, processors, and configurations from a variety of computer manufacturers. These speedup numbers are relative to a Sun Ultra 5/10 workstation with a 300-MHz Sparc processor and 256 Mbytes of main memory.

Of the 1,381 SPECint and 1,399 SPECfp machines in the SPEC CPU2000 dataset, we selected all machines for which both SPECint and SPECfp results were submitted on the same date, resulting in 1,123 machines. Table 2 provides an overview of our dataset per architecture. These performance numbers were published between the fourth quarter of 1999 and the first quarter of 2007. The processors are implemented in 65- to 500-nanometer complementary metal-oxide semiconductor technology with a clock frequency ranging from 250 MHz to 3.8 GHz and average base SPECint and SPECfp speed numbers ranging from 93.7 and 84.4, to 3,108 and 3,369, respectively.

We denote the data matrix of commercial machine performance numbers as $S$. The $S$ matrix contains perfor-

 Published by the IEEE Computer Society 

mance speedup numbers for all machines and benchmarks in the SPEC CPU2000 benchmark suite. The $S$ matrix consists of 1,123 rows—there are 1,123 machines in our dataset—and 26 columns—there are 26 benchmarks in the SPEC CPU2000 benchmark suite.

## PERFORMANCE ANALYSIS METHODOLOGY

To simplify our raw dataset, we need a statistical analysis technique that can reduce the $S$ matrix into a tractable dataset without losing too much information. Thus, we transform the $S$ matrix into a lower-dimensional $S_t$ matrix in two steps: data normalization, followed by statistical analysis using PCA. We then normalize the output of PCA to create the $S_{t,n}$ matrix.

### Data normalization

The input given to PCA can be a raw dataset or a normalized dataset. A raw input dataset, such as our $S$ matrix, gives a higher weight in the PCA analysis to variables that range across a larger span of absolute values. For example, if variable A ranges from 1 to 100 and variable B ranges from 1 to 20, variable A will have a higher weight in the analysis. A normalized dataset, in which all variables are normalized to a zero mean and unit variance, gives equal importance to all variables.

In our setup, we normalize the $S$ matrix columnwise to create $S_n$, rendering all benchmarks on a common scale. Note that normalizing the dataset prior to statistical data analysis is an experimental design decision made by the performance analyst. If desired, the analyst can give a higher weight to a particular benchmark.

### Statistical data analysis

We use principal components analysis to reduce the dataset's dimensionality. The input to PCA is the normalized matrix $S_n$ in which the rows are the machines and the columns are the normalized variables (benchmark speedup numbers). PCA computes new variables called principal components (PCs), which are linear combinations of the original variables, such that all principal components are uncorrelated. PCA transforms the $p$ variables $S_{n,1}$, $S_{n,2}$, …, $S_{n,p}$ into $p$ principal components $S_{t,1}$, $S_{t,2}$, …, $S_{t,p}$ with $S_{t,i} = \sum_{j=1}^{p} T_{ij} S_{n,j}$. The $T_{i,j}$ coefficients are computed by PCA and are the factor loadings of variable $j$ for principal component $i$. This transformation has the following properties:

- $Var[S_{t,1}] \geq Var[S_{t,2}] \geq \ldots \geq Var[S_{t,p}]$—$S_{t,1}$ contains the most information and $S_{t,p}$ the least.
- $Cov[S_{t,i}, S_{t,j}] = 0, \forall i \neq j$—There is no information overlap between the principal components.

Note that the total variance remains the same before and after the transformation, namely $\sum_{i=1}^{p} Var[S_{n,i}] = \sum_{i=1}^{p} Var[S_{t,i}]$. Variable $S_{n,i}$ represents the normalized speedup for benchmark $i$; therefore, $S_{t,i}$ is the $i$th principal component after PCA. $Var[S_{n,i}]$ is the variance of the normalized speedup for benchmark $i$ computed across all machines; likewise, $Var[S_{t,i}]$ is the variance of principal component $i$ across all machines.

Some principal components account for a higher variance than others. Removing the principal components with the lowest variance from the analysis reduces the

**Table 1. The SPEC CPU2000 integer and floating-point benchmarks.**

| | | |
|---|---|---|
| **SPECint** | bzip2 | Compression |
| | crafty | Chess game |
| | eon | Computer visualization |
| | gap | Group theory, interpreter |
| | gcc | GNU C compiler |
| | gzip | Compression |
| | mcf | Combinatorial optimization |
| | parser | Word processing |
| | perlbmk | Perl programming language |
| | twolf | Place and route simulator |
| | vortex | Object-oriented database |
| | vpr | FPGA circuit placement and routing |
| **SPECfp** | ammp | Computational chemistry |
| | applu | Parabolic/Elliptic partial differential equations |
| | apsi | Meteorology: pollutant distribution |
| | art | Image recognition/Neural networks |
| | equake | Seismic wave propagation simulation |
| | facerec | Image processing: face recognition |
| | fma3d | Finite-element crash simulation |
| | galgel | Computational fluid dynamics |
| | lucas | Number theory/Primality testing |
| | mesa | 3D graphics library |
| | mgrid | Multigrid solver |
| | sixtrack | High-energy nuclear physics accelerator design |
| | swim | Shallow water modeling |
| | wupwise | Physics/Quantum chromodynamics |

**Table 2. Machines used in this study.**

| Architecture | No. of machines |
|---|---|
| AMD x86 (32-bit) | 28 |
| AMD x86 (64-bit) | 181 |
| DEC Alpha | 23 |
| Fujitsu Sparc64 | 32 |
| HP PA-RISC | 14 |
| IBM PowerPC | 83 |
| Intel Itanium (IA-64) | 43 |
| Intel x86 (32-bit) | 250 |
| Intel x86 (64-bit) | 410 |
| MIPS | 10 |
| Sun UltraSparc | 49 |

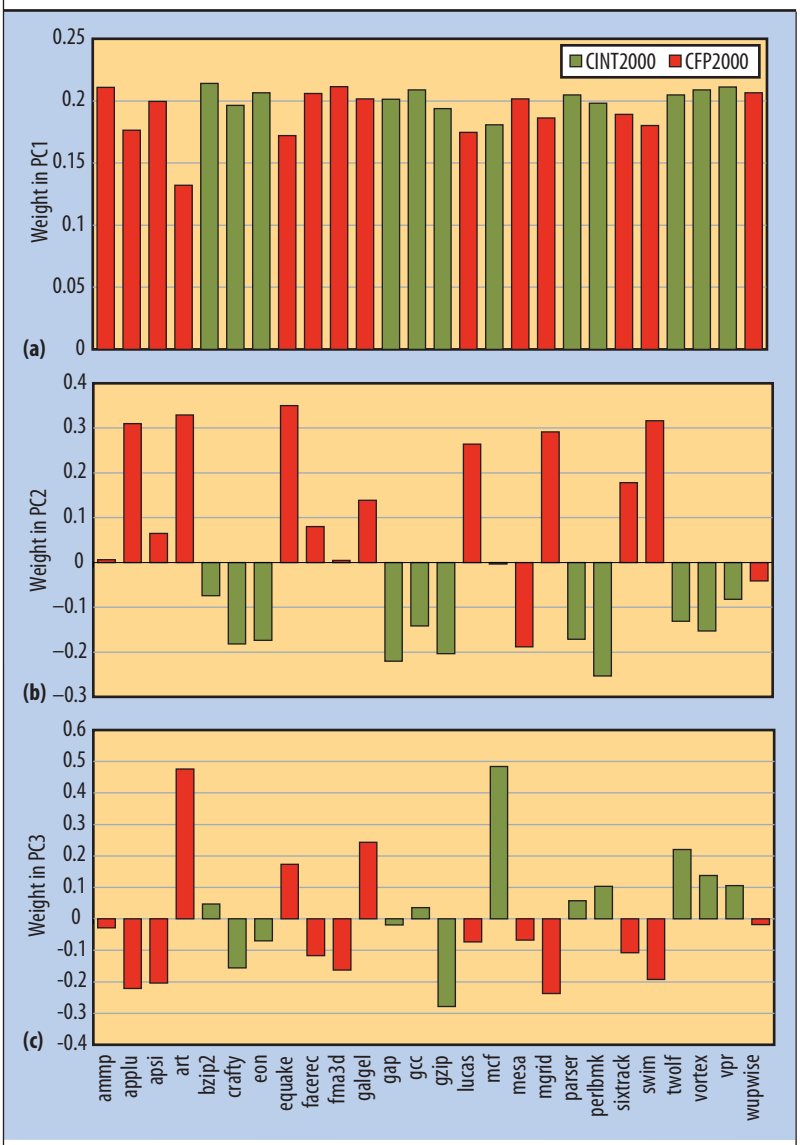



**Figure 1.** The factor loadings per benchmark for (a) PC1, (b) PC2, and (c) PC3.

dimensionality of the dataset while controlling the amount of information lost. We retain $q$ principal components, which is a significant information reduction because $q << p$ in most cases. To measure the fraction of information retained in this $q$-dimensional space, we use the amount of variance $(\sum_{i=1}^{q} Var[S_{t,i}]) / \sum_{i=1}^{p} Var[S_{n,i}])$ accounted for by these $q$ principal components. Performance analysts can choose an appropriate percentage of total variance.

PCA's output is the $S_t$ matrix in which the rows are the machines and the columns are the retained principal components.

## PC normalization

The next step is normalizing the principal components, which places all principal components on a common scale. PCA finds the key underlying mechanisms that correlate with the dataset, then represents these as principal components. After normalization, the underlying mechanisms get equal weight in the transformed dataset after PCA. The end result of the PCA analysis and its subsequent normalization step is a transformed matrix $S_{t,n}$.

## ANALYZING THE SPEC CPU2000 PERFORMANCE NUMBERS

Visualizing the 1,123 machines in terms of the retained principal components—the new $S_{t,n}$ dataset—reveals major performance trends in the original dataset.

## Principal components

The first three principal components encompass 92.9 percent of the total variance observed in our original dataset. The first principal component explains 81 percent of the total variance; the second, 7.7 percent; and the third, 4.2 percent. We limit our analysis to three principal components because they explain most of the variance observed in the dataset while enabling data visualization; the transformed $S_{t,n}$ dataset preserves the major performance trends.

Figure 1 represents the three principal components in terms of the normalized benchmark performance numbers, that is, the factor loadings ($T_{ij}$) for the three principal components. For example, Figure 1a shows that the score along the first principal component for a given machine is computed as: $0.21 \times S_{n,ammp} + 0.18 \times S_{n,applu} + 0.20 \times S_{n,apsi} + 0.13 \times S_{n,art} + \ldots$

Because the weights along the first principal component, shown in Figure 1a, are approximately the same for all benchmarks, PC1 represents the average normalized speedup across all benchmarks. Therefore, a machine with a high score for PC1 achieves relatively better average performance than a machine with a low score for PC1. The only benchmark that has a relatively small weight in the first principal component is art; the reason is that the speedup numbers vary widely for art across the machines, more so than for any other benchmark. For about 10 percent of the machines, art reports a speedup number greater than 8,236 (the maximum speedup number observed across the other benchmarks) with a maximum of 26,443; none of the other benchmarks show this large a speedup on any of the machines. The reason for this high speedup range lies in the aggressive compiler optimizations applied on some machines.[2]

The most prominent dimension in the dataset (the most significant principal component) has the largest vari-

ance and correlates well with average performance. The correlation coefficients between the first principal component and the SPECint and SPECfp base scores are 96.9 percent and 97.4 percent, respectively. PCA thus recognizes that the large performance increase achieved over the past several years through enhancements in compiler support, architecture, and chip technology is by far the most predominant dimension in the dataset.

The second principal component, shown in Figure 1b, gives a positive weight to most of the floating-point benchmarks and a negative weight to all integer benchmarks. As such, a machine with a high score for PC2 yields relatively better normalized performance for the floating-point benchmarks than for the integer benchmarks, whereas a low value for PC2 indicates relatively better normalized integer performance compared to floating-point performance.

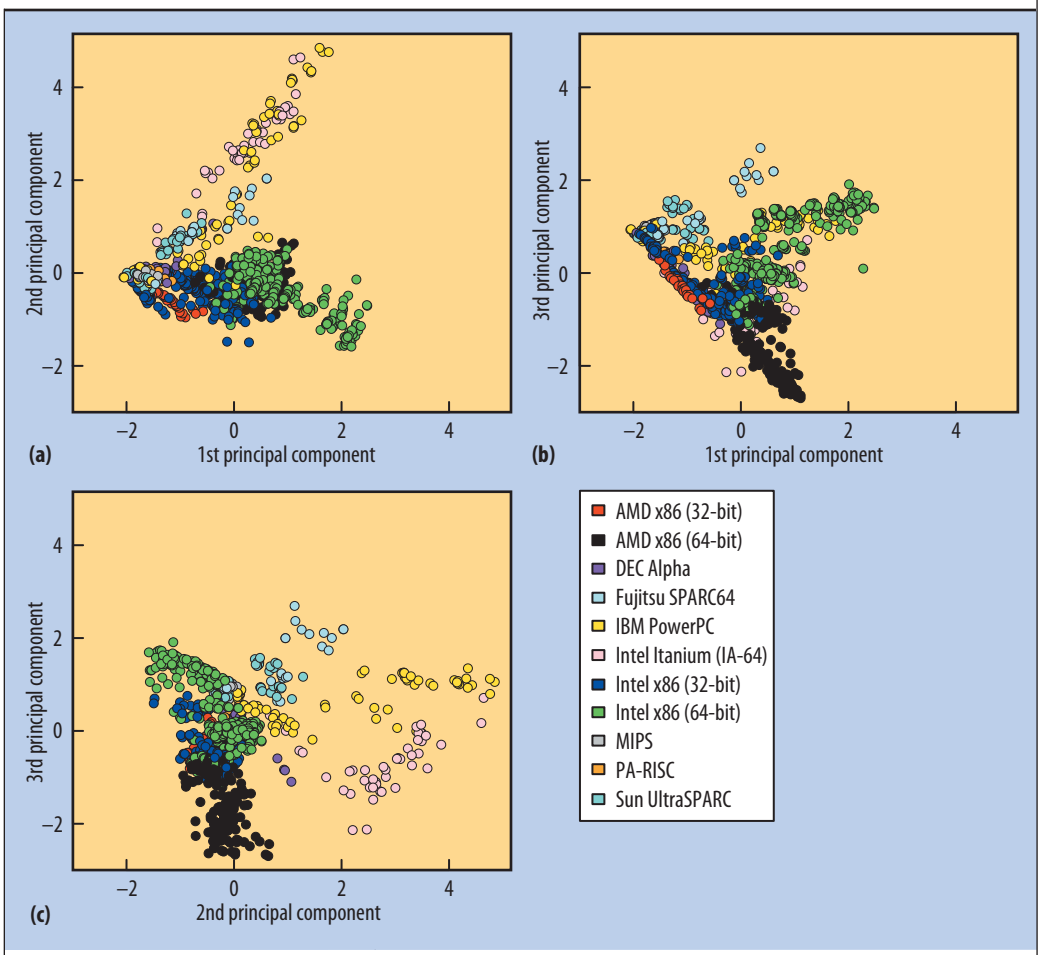



**Figure 2.** Visualizing the SPEC CPU2000 performance space: (a) PC2 versus PC1, (b) PC3 versus PC1, and (c) PC3 versus PC2 in terms of architecture.

Two floating-point benchmarks, mesa and wupwise, get a negative weight, while all the other floating-point benchmarks receive a positive weight. This categorization suggests that mesa and wupwise stress systems in a way similar to the integer benchmarks. We confirm this in a behavioral characterization using the Microarchitecture-Independent Characterization of Applications tool[3] (www.elis.ugent.be/~kehoste/mica). We found instruction-level parallelism, the fraction of memory accesses and floating-point operations, and the branch misprediction rates for mesa and wupwise to be in the range of the integer benchmarks and significantly different from the other floating-point benchmarks. More specifically, mesa and wupwise show less ILP, more memory operations, fewer floating-point operations, and higher branch misprediction rates than the other floating-point benchmarks.

In the third principal component, shown in Figure 1c, two benchmarks are assigned a significantly higher weight than the others, namely art and mcf. Both benchmarks are known to stress the memory hierarchy subsystem more than any of the other SPEC CPU2000 benchmarks. From a detailed characterization using microarchitecture-independent behavioral characteristics,[3] we found that these benchmarks exhibit many unique references between two accesses to the same memory location, that is, temporal locality is poor. Machines with a high value for PC3 thus yield relatively better performance for memory-intensive workloads with poor temporal data locality than machines with a lower value.

## Analyzing the SPEC CPU2000 performance space

Figure 2 shows the various machines in a PCA space. The three two-dimensional plots show the second PC as a function of the first PC, the third PC as a function of the first PC, and the third PC as a function of the second PC. Each symbol in these graphs represents one machine; there are 1,123 machines in each graph, and the colors symbolize the various architectures.

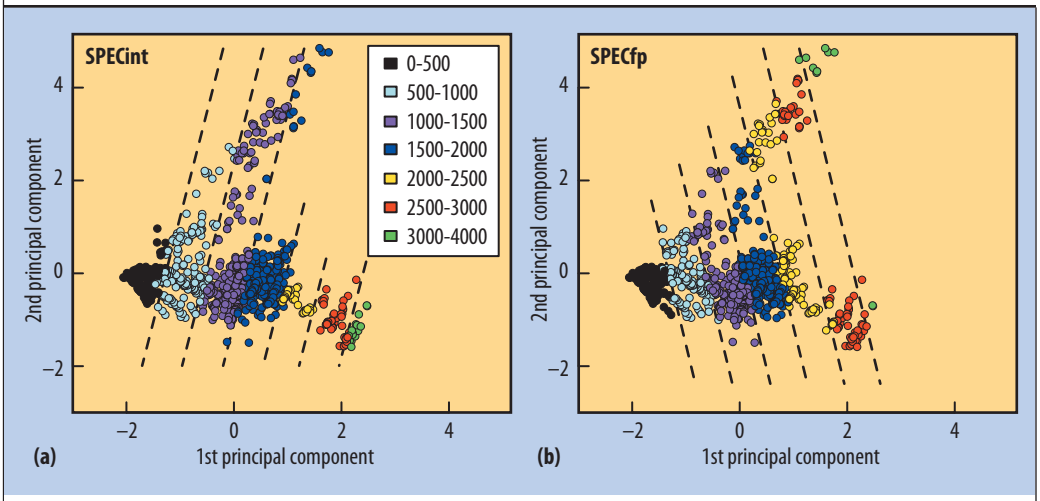



**Figure 3.** Visualizing PC2 versus PC1 for the PCA space obtained from the SPEC CPU2000 performance numbers in terms of average (a) SPECint and (b) SPECfp speedup numbers.

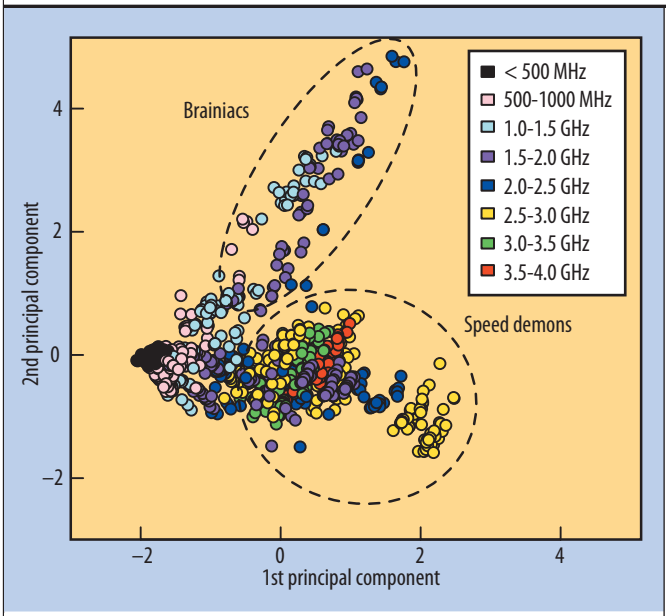



**Figure 4.** Visualizing the PCA space obtained from the SPEC CPU2000 performance numbers in terms of processor clock frequencies.

The right bottom corner in the PC2 versus PC1 plot in Figure 2a shows that the Intel x86 64-bit machines yield better average performance for the integer benchmarks than the other machines do, while the IBM PowerPC and Intel Itanium (IA-64) machines perform better on the floating-point benchmarks. Figure 3 also shows various machines as a function of PC1 and PC2, categorized by average SPEC performance number. Note that the performance wave for the integer benchmarks shown in Figure 3a has a different orientation than the performance wave for the floating-point benchmarks in Figure 3b.

Further, the PC3 versus PC1 plot in Figure 2b and the PC3 versus PC2 plot in Figure 2c show that although both the IBM PowerPC and Intel Itanium machines achieve similar average performance for the integer and floating-point benchmarks, they exhibit different behavior in the third principal component. This behavior is due to the IBM PowerPC machines performing relatively better than the Intel Itanium machines for the memory-intensive benchmarks with poor temporal data locality (that is, art and mcf). Likewise, when comparing Intel x86 64-bit and AMD x86 64-bit machines, Intel performs better than AMD for memory-intensive workloads.

Figure 4 represents the same data in another way. Coloring the machines by their processor clock frequency reveals two groups of machines—the speed demons and the brainiacs. The speed demons achieve high performance mainly through high clock frequencies; the brainiacs, on the other hand, achieve high performance through a high instruction throughput per cycle at a relatively slow clock frequency.

The speed demons are the Intel and AMD x86 64-bit machines; these machines achieve high performance for the integer benchmarks through their more than 2.5-GHz clock frequencies. The brainiacs are the IBM PowerPC and Intel Itanium machines that achieve high performance on the floating-point benchmarks by attaining high instruction throughput per cycle at moderate clock frequencies of less than 2.5 GHz.

## Case study: Intel Pentium 4 architecture

Our case study features a subset of the Intel Pentium 4 (NetBurst) architecture machines. Figure 5 shows the evolution across the various Intel machines, categorized by processor type. The analysis confirms the general expectation, namely average performance improves across the Intel Pentium 4 machine generations (the data points go from left to right across the different generations). The different machines within a single generation show a negative slope in both the PC2 versus PC1 (Figure 5a) and PC3 versus PC1 (Figure 5b) graphs.

This pattern suggests that increasing the clock frequency within a processor generation (which is the

main contributor to the variation in performance) improves performance more for the compute-intensive and integer benchmarks than for the memory-intensive and floating-point benchmarks. Because memory-intensive benchmarks spend more time waiting for memory, increasing clock frequency does not improve performance as much for memory-intensive applications. The Northwood processor generation has three subgenerations, each representing different memory bandwidth characteristics ranging from 1.6 gigabytes per second to 2.1-2.7 gigabytes per second and up to 3.2 gigabytes per second.

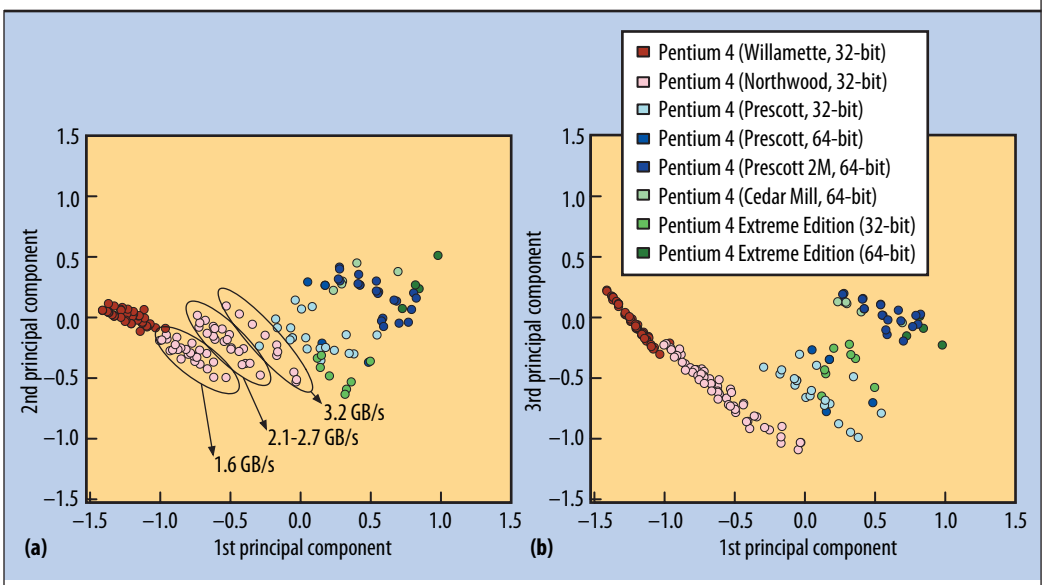



**Figure 5.** Case study: Intel Pentium 4 processors as a function of (a) PC2 versus PC1 and (b) PC3 versus PC1.

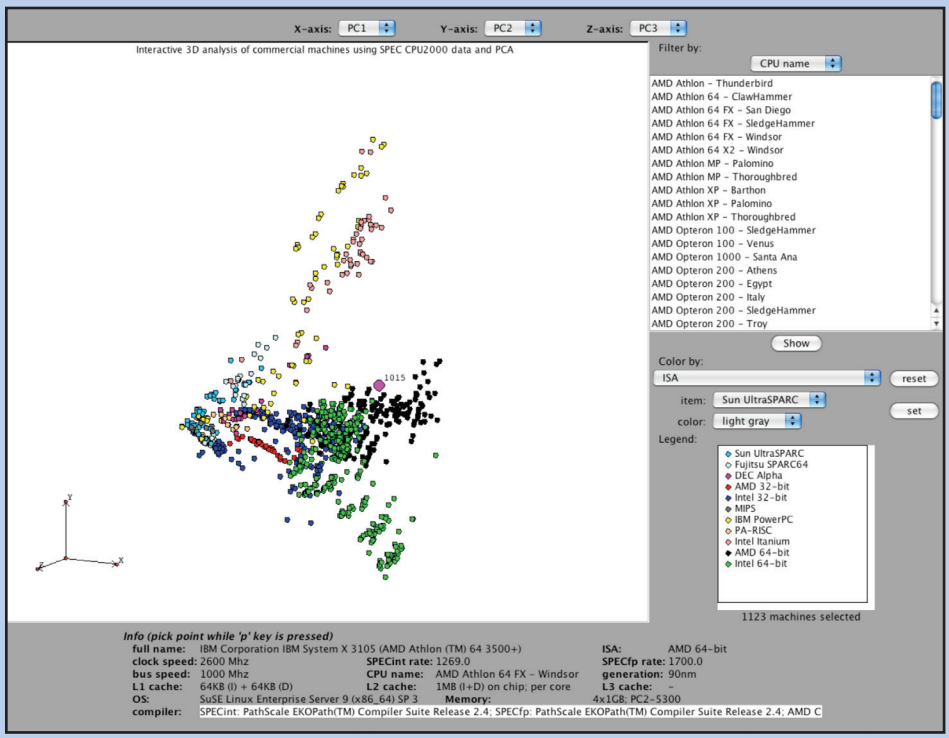



**Figure 6.** Processor performance visualizer (PPV) tool for interactively exploring the SPEC CPU2000 performance space. Various functions include getting detailed information about a particular datapoint by selecting it in the 3D plot, rotating and scaling the space, or applying a filter on the dataset.

## INTERACTIVE VISUALIZATION

To facilitate the exploration of the three-dimensional SPEC CPU2000 performance space, we developed the processor performance visualizer, an interactive Java applet (www.elis.ugent.be/~kehoste/PPV). The input to the PPV tool is the transformed data matrix $S_{t,n}$ obtained after PCA and normalization. Figure 6 shows PPV applied to the SPEC CPU2000 performance numbers. The source code is available so performance analysts can use the tool on their own datasets.

Computer architects deal with a wealth of performance numbers on a daily basis. From our research, we learned that a relatively simple statistical data analysis provides valuable, high-level insights into these large perform-

## OTHER WORK IN STATISTICAL PROCESSOR PERFORMANCE ANALYSIS

Recently, interest in using statistics in microprocessor performance analysis has increased.

In particular, David Lilja[1] described various statistical data analysis techniques that computer architects can readily use for taking meaningful conclusions from large datasets. These techniques include the t-test, linear regression models, and design of experiments. Joshua Yi and colleagues[2] described the Plackett-Burman design of experiments, a fractional factorial method for identifying microarchitecture design parameters that have a large impact on overall performance using a limited number of simulations.

Alaa Alameldeen and David Wood[3] observed that multiprocessor performance is susceptible to nondeterministic effects—subtle changes that can lead to different interleavings and interactions between threads executing on a multiprocessor. To account for these nondeterministic effects, these authors proposed a methodology that introduces nondeterminism in deterministic simulators and then uses statistics for computing confidence bounds.

Tom Conte and colleagues[4] and Roland Wunderlich and colleagues[5] also proposed using confidence bounds to determine how many sampling units to take for estimating uniprocessor performance. Magnus Ekman and Per Stenström[6] showed that a matched-pair comparison reduces the number of samples that need to be taken to estimate a change in performance between alternative processor architectures. Michael Van Biesbrouck and colleagues[7] use a statistical approach to select multiple starting points for simulating multiprogram workloads on multithreaded processor architectures.

Other previous work[8] used principal components analysis as a data reduction technique for identifying behavioral similarities across benchmarks with the intent to compose a representative set of benchmarks.

### References


1. D.J. Lilja, *Measuring Computer Performance: A Practitioner's Guide*, Cambridge Univ. Press, 2000.
2. J.J. Yi, D.J. Lilja, and D.M. Hawkins, "A Statistically Rigorous Approach for Improving Simulation Methodology," *Proc. 9th Int'l Symp. High-Performance Computer Architecture* (HPCA 03), IEEE CS Press, 2003, pp. 281-291.
3. A. Alameldeen and D. Wood, "Variability in Architectural Simulations of Multi-threaded Workloads," *Proc. 9th Int'l Symp. High-Performance Computer Architecture* (HPCA 03), IEEE CS Press, 2003, pp. 7-18.
4. T.M. Conte, M.A. Hirsch, and K.N. Menezes, "Reducing State Loss for Effective Trace Sampling of Superscalar Processors," *Proc. Int'l Conf. Computer Design* (ICCD 96), IEEE Press, 1996, pp. 468-477.
5. R.E. Wunderlich et al., "SMARTS: Accelerating Microarchitecture Simulation via Rigorous Statistical Sampling," *Proc. Ann. Int'l Symp. Computer Architecture* (ISCA), ACM Press, 2003, pp. 84-95.
6. M. Ekman and P. Stenström, "Enhancing Multiprocessor Architecture Simulation Speed Using Matched-Pair Comparison," *Proc. IEEE Int'l Symp. Performance Analysis of Systems and Software* (ISPASS 05), IEEE Press, 2005, pp. 89-99.
7. M. Van Biesbrouck, L. Eeckhout, and B. Calder, "Considering All Starting Points for Simultaneous Multithreading Simulation," *Proc. IEEE Int'l Symp. Performance Analysis of Systems and Software* (ISPASS 06), IEEE Press, 2006, pp. 143-153.
8. L. Eeckhout, H. Vandierendonck, and K. De Bosschere, "Workload Design: Selecting Representative Program-Input Pairs," *Proc. Int'l Conf. Parallel Architectures and Compilation Techniques* (PACT), IEEE CS Press, 2002, pp. 83-94.


ance datasets. To the best of our knowledge, this is the first work providing a comprehensive methodology for analyzing performance trends across a large dataset of commercial processor performance numbers. However, the "Other Work in Statistical Processor Performance Analysis" describes other recent research into the use of statistics in microprocessor performance analysis.

Our future work will try to advance this statistical analysis approach by addressing a key challenge in benchmarking: comparing the program characteristics of an application of interest with the industry-standard benchmarks in the performance dataset to estimate the application's performance on a range of commercial machines.

## Acknowledgments

Kenneth Hoste is supported through a PhD student fellowship from the Institute for the Promotion of Innovation by Science and Technology in Flanders, Belgium. Lieven Eeckhout is supported through a postdoctoral fellowship from the Fund for Scientific Research—Flanders (FWO). Additional support was provided by FWO project G.0255.08.

## References


1. R.A. Johnson and D.W. Wichern, *Applied Multivariate Statistical Analysis*, 5th ed., Prentice Hall, 2002.
2. M. Wolfe, "Compilers and More: Gloptimizations," *HPCwire*, 9 Nov. 2007; www.hpcwire.com/topic/developertools/Compilers_and_More_Gloptimizations.html.
3. K. Hoste and L. Eeckhout, "Microarchitecture-Independent Workload Characterization," *IEEE Micro*, May/June 2007, pp. 63-72.


***Kenneth Hoste*** *is a PhD student in the Electronics and Information Systems Department at Ghent University, Belgium. Hoste received an MS in computer science from Ghent University. Contact him at kehoste@elis.UGent.be.*

***Lieven Eeckhout*** *is an assistant professor in the Electronics and Information Systems Department at Ghent University. Eeckhout received a PhD in computer science and engineering from Ghent University. He is a member of the IEEE and the ACM. Contact him at leeckhou@elis.UGent.be.*