Measuring Computer Systems: How to Measure Performance

Margo Seltzer, Aaron Brown

Harvard University
Division of Engineering and Applied Sciences

{margo, abrown}@eecs.harvard.edu

Abstract

"Benchmarks shape a field (for better or worse); they are how we determine the value of change." (David Patterson, 1994).

If benchmarks shape a field, then computer science is in poor condition. Our field is characterized by a few excellent benchmarks, and a large number of poorly conceived and executed measurements. Many benchmarks are created with the sole purpose of making one system look better than competing systems. Other studies use good, fair benchmarks, but offer no conclusions, draw completely unsupportable conclusions, or neglect to present the data necessary to support the conclusions drawn. Another common approach is to take good benchmarks and use them in inappropriate ways. The prevalence of bad benchmarking practice is understandable: careful, thorough measurement is both extremely difficult and enormously time consuming.

In this talk, we will examine current practice in computer system measurement, citing surprisingly good and embarrassingly bad real-world examples. We will use these examples to illustrate the common pitfalls that await an eager system evaluator. We will examine one study in depth, highlighting the strengths and weaknesses of its methodological approach. Based on the discussion of existing practice and the sample case study, we will offer some benchmarking tips and guidelines.

Outline

- Why is measurement important?
- So, what's the problem?
- Some basic rules for measurement.
- A measurement case study.
- Lessons learned.

Why Benchmark?

"Benchmarks shape a field (for better or worse); they are how we determine the value of change."

—David Patterson, 1994

Benchmarks Shape a Field

- **Database and TP Community**
  - Debit/Credit
  - TP1
  - TPC/A-B
  - TPC/C
  - 001
  - TPC/D
- **Architecture Community**
  - Linpack
  - SPEC

Shaping a Field (2)

- **File System Community**
  - Trace-based analysis of the 4.2 file system
  - Andrew File System Benchmark
  - Bonnie
  - LADDIS
- **PC Community**
  - Byte benchmarks
  - PC Magazine Winstone and WinBench
- **Spelling Community**
  - Whetstone
  - Dhrystone
  - NHFSstone

The Shape may be Incorrect

- **Bonnie: Ignores the software.**
- **Laddis: Ignores how file systems change over time.**
- **SPEC: Ignores the operating system.**
- **TPC/X: Ignores everything that's not transaction processing.**
- **Winstone: Ignores the user.**

Why You Care

- **System designers**
  - How does my system perform?
  - Are these features worthwhile?
  - Did my changes make a difference?
- **Application designers**
  - Did I make my product slower?
  - How do I stack up to the competition?
- **System administrators**
  - What system shall I recommend?
  - How shall I configure this machine?
  - Why is my machine slow?

Why You Care (2)

- **Customers**
  - What system shall I buy?
  - What software shall I buy?

Common Measurement Problems

- **Measuring the wrong thing.**
- **Drawing inappropriate conclusions.**
- **Using bad statistics.**
- **Ignoring system interaction.**
- **Ignoring timing granularity.**

For each of these categories, we will present examples from the literature.

More Problems

- Comparing apples to oranges.
- Comparing end-to-end measurements with the sum of parts.
- Using the wrong metrics.
- Ignoring how the system optimizes.
- Mistakes.

Problems Exist in Other Disciplines

- Conclusions unsupported by data.

Statement: Modern Physics has proved the existence of God.

Explanation: In “Quantum Leap” (September/October 1989), physicist David Bohm confirmed the parallels between traditional views of spirituality and divinity and the latest theories being examined in modern physics.

Statement: A tender embrace is more fulfilling than sexual intercourse.

In “The New Eroticism” (May/June 1993), Brenda Peterson reported that when 100,000 women were asked the question: “Would you be content to be held close and treated tenderly and forget about ‘the act’?”, 72 percent said yes, they would be content to be simply held. Remarkably, 40 percent of the respondents were under 40 years old.

From *New Age Journal*

What’s so Hard?

- Systems are complex, and interactions in systems are even more complex.
- Most systems are undocumented (no source code).
- Each system is different and has differing tools or implementations.
- Every user wants something different (e.g., OS researcher versus an end-user).
- Obtaining reproducible, statistically significant numbers is difficult.

More Problems

- Some operations are far faster than the timer resolution, and repeating measurements introduces other factors (e.g., warming of the cache).
- Systems are non-deterministic.
- Benchmarks do not accurately reflect any particular workload.
- Without complete detailed knowledge of both HW and SW, it is nearly impossible to explain exactly what is happening.

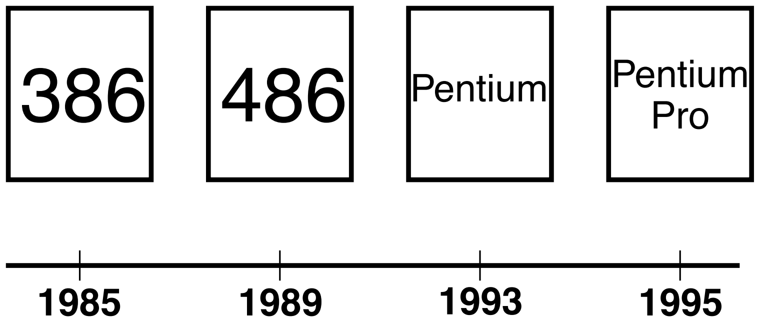
Case Study

- **Deceptively simple project:**
  - Start with a good benchmark (lmbench).
  - Select a single operating system (NetBSD).
  - Select a single processor family (Intel).
  - Goal: Understand (and quantify) how operating system primitives have scaled with architectural evolution.
- **PC Magazine does similar things all the time.**

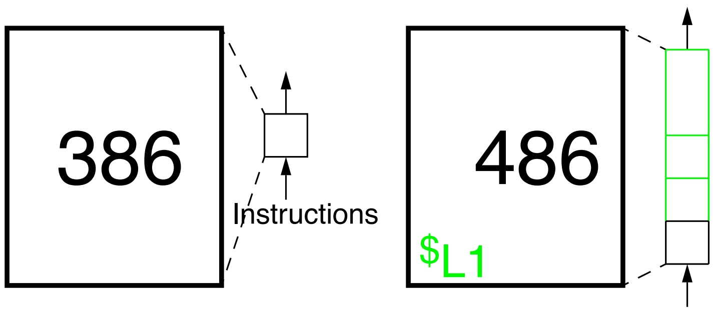
Good Benchmarking

- **Control variables.**
  - Same operating system.
  - Identical benchmarks.
  - Identical disk, executables, file system, etc.
- **Vary the hardware.**
  - Four processor generations (386–Pentium Pro).
  - Five memory systems.
- **Goal: explain each result in terms of its hardware dependencies.**

Test Systems



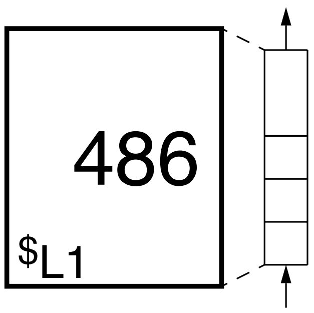
Evolution: 386–486

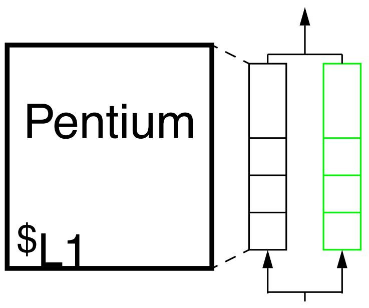


- 80386: 32-bit core
- max 33 MHz external bus
- non-pipelined, no on-chip caches

- 80486: 32-bit core
- max 33 MHz, 32-bit external bus
- pipelined, 8K L1 cache on-chip

Evolution: 486–Pentium

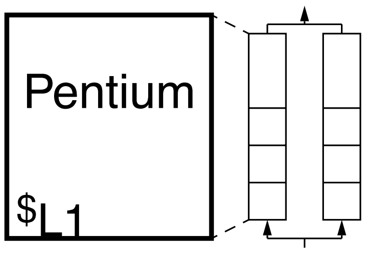


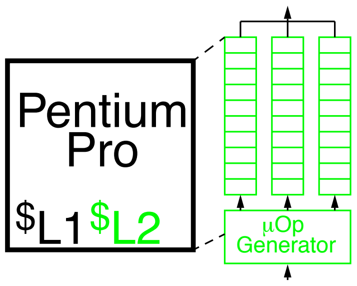


- 80486: 32-bit core
- max 33 MHz, 32-bit external bus
- pipelined, 8K L1 cache on-chip

- Pentium: 32-bit core
- max 66 MHz, 64-bit external bus
- dual-issue superscalar pipelined

Evolution: Pentium–Pro





- Pentium: 32-bit core
- max 66 MHz, 64-bit external bus
- dual-issue superscalar pipelined

- Pentium Pro: 32-bit core
- max 66 MHz, 64-bit external bus
- out-of-order RISC-like core with 3 micro-op pipelines
- 16K L1 cache on-chip
- 256K or 512K L2 cache in the

Challenges

- **Mapping the benchmarks into reality.**
  - What aspects of the system hardware and/or software were being measured.
- **Understanding the results.**
  - Why was file re-read so fast?
  - Why did read/write bandwidths differ?
- **Understanding the accuracy of the timer.**
  - What unit of time could we report?
- **Measurement methodology.**
  - When were averages calculated?
  - When was the reported result the minimum time measured?

More Challenges

- **Modifying things and keeping them portable.**
- **Drawing high-level, useful observations out of low-level benchmarks.**
- **Distinguishing between problems in the benchmark, odd hardware behavior, and differing assumptions concerning what the benchmark was measuring.**

Identifying Problems

- **How we figured out what was going on.**
  - Source code analysis (high-level and in assembler).
  - Instrumentation with the Pentium counters.
  - Kernel profiling.
  - Code tweaking to isolate suspect behaviors.
- **Areas where we found problems.**
  - Timing methodology.
  - Statistical methodology.
  - Data collection methodology.
  - Reporting methodology.

Fixing Problems

- **Timing.**
  - Internal, dynamically-sized loops.
  - Support hardware counters/timers where available.
- **Statistics.**
  - Apply consistent policy (discard outliers, average rest).
- **Data collection.**
  - Standardize on array-index-style memory references.
  - Revamp overhead calculations for consistency.
  - Allow increased parameterization.
- **Reporting.**
  - Preserve all data; allow user to specify analysis policy.

hBench:OS

http://www.eecs.harvard.edu/hbench

- **Still warm cache results.**
- **Reproducible results (stability).**
- **Some things still unexplained.**
- **We understand the limits of what we can explain with these benchmarks and what we cannot.**
  - Does not directly explain application performance.
  - Micro-benchmark nature makes it difficult to draw useful comparisons between different OS/hardware combinations.

Explaining Results

- **Some phenomena are difficult to explain.**
  - Architecture-specific counters help, but introduce portability problems.
  - Difficult to verify hypotheses.
  - Architecture manuals are never as complete as one would like.
- **Example:**
  - Pentium-Pro demonstrates 3x performance differential between memory read and write performance.

Rules for Good Measurement: Consumer Perspective

- **Know what you are trying to understand.**
  - Why are you benchmarking?
  - What are you trying to learn?
- **Understand the limitations of your tools/ benchmarks.**
  - What timing mechanism is being used?
  - How accurate is it?
- **Test from a consistent state.**
  - Single-user/multi-user, pick one.
  - Freshly booted.
  - Network attached/unattached, pick one.

Consumer Perspective (2)

- **Find the causes of your results.**
  - Pay particular attention to the anomalies.
  - Make sure they are real anomalies, not measurement glitches.
  - Assume first that it’s something in your system.
  - When all else fails, consider that the benchmark might be at fault.
- **Real anomalies are often likely to give you real data.**
  - Cross reference results as much as possible.
  - Compare two different benchmark results.
  - Tweak variables to see if hypotheses are valid.

Consumer Perspective (3)

- **Synthesize your results to ensure a coherent and believable picture.**
  - Which of your conclusions can be proven?
  - Which are conjectures?

Benchmark Designer Perspective

- **Preserve all data.**
  - Summaries are useful.
  - Raw data must be available for more detailed analysis.
- **Portability**
  - Be precise about what cross-platform conclusions you can accurately draw.
  - Know whether you are stating something about the hardware, the operating system, the compiler, or an application.
  - Understand what you lose by portability (e.g., detailed counter measurements).

Designer Perspective (2)

- **Pick a consistent methodology.**
  - Measurement.
  - Reporting.
  - Data collection.
  - Timing.
  - Benchmark structure.
- **Isolate variables to the extent possible.**
  - Makes it easier to explain results.
  - Encourages/enables testing of hypotheses.
- **Know your audience**
  - Identify the target audience (e.g., researchers, end-users, vendors).

Designer Perspective (3)

- **Articulate exactly what your results mean.**
  - What is a SPECmark?
  - How can someone use your results?
  - How should (shouldn't) they use your results?
- **Tell the user how to interpret the results.**
- **Think about data presentation.**
- **Know what you are measuring (and ensure that it's what you intended to measure).**

Designer Perspective (4)

- **Show meaningful examples and comparisons.**
  - Comparing Windows NT on the Pentium versus the Pentium Pro will tell you something about hardware.
  - Comparing Windows NT to Windows 3.1 on the Pentium will tell you something about software.
  - Comparing Windows NT on a Pentium to Solaris on a SPARCstation doesn't tell you much about anything.
- **Use the smallest effective difference.**
  - Edward Tufte's "The Visual Display of Quantitative Information."

Lessons to Take Away

- **Measurement is important; do it carefully.**
- **Don't be dazzled by bad numbers (even if there are a lot of them).**
- **Understand what a number means before you publish, quote, or reference it.**
- **Apply the principle of smallest effective difference.**
- **Measurement is more than a marketing ploy; it's serious science, not an art.**