

---

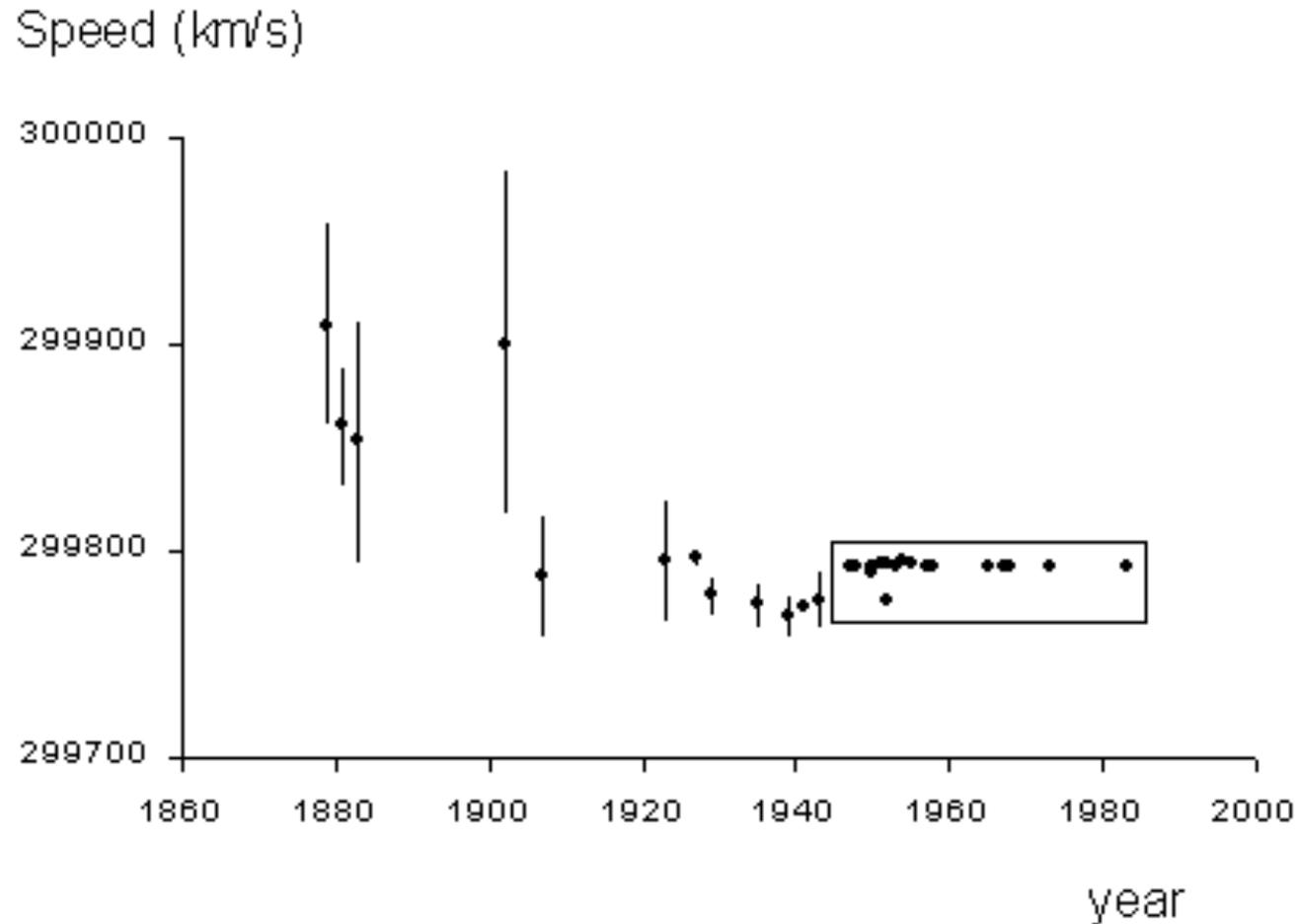
# **Misleading Performance Claims in Parallel Computations**

**David H. Bailey**  
**Lawrence Berkeley National Laboratory**  
**<http://crd.lbl.gov/~dhbailey>**

---

# Example from Physics: Measurements of Speed of Light

---



Why the discrepancy between pre-1945 and post-1945 values?  
Probably due to biases and sloppy experimental methods.

---

# Example from Psychology: The “Blank Slate”

---

## The “blank slate” paradigm (1920-1980):

- ◆ The human mind at birth is a “blank slate.”
- ◆ Heredity and biology play no significant role in human psychology – all personality and behavioral traits are socially constructed.

## Current consensus, based on latest research:

- ◆ Humans at birth universally possess sophisticated facilities for language acquisition, pattern recognition and social life.
- ◆ Heredity, evolution and biology are major factors in human personality.
- ◆ Some personality traits are as much as 70% heritable.

## How did the early 20th century scientists get it so wrong?

- ◆ Sloppy experimental methodology and analysis.
- ◆ Pervasive “politically correct” biases and wishful thinking.

Ref: Steven Pinker, *The Blank Slate: The Modern Denial of Human Nature*

---

## Example from Anthropology: The “Noble Savage”

---

Anthropologists, beginning with Margaret Mead in the 1930s, taught that primitive societies (such as South Sea Islanders) were idyllic:

- ◆ Virtually no violence, jealousy or warfare.
- ◆ Happy, uninhibited – few psychological problems or “hangups.”

Beginning in the 1980s, a new breed of anthropologists visited these islands and did more careful studies. They concluded:

- ◆ These societies typically have murder rates several times higher than large U.S. cities.
- ◆ Death rates from inter-tribe warfare exceed those of warfare among Western nations by factors of 10 to 100.
- ◆ Complex, jealous taboos surround sexuality, courtship and marriage.
- ◆ Example: Until recently, in one large island society, customs condoned the beating or killing of non-virgin brides and suspected adulterers.

Why were the earlier studies so wrong?

Answer: “Anthropological malpractice” – Pinker

---

# Lessons From History

---

- ◆ High standards of honesty and scientific rigor must be vigilantly enforced within a field.
  - ◆ Rigorous peer review is essential.
  - ◆ Scientific research must be based on solid empirical data and careful, objective analysis of that data.
  - ◆ Scientists must be willing to provide all details of the experimental environment, so others can reproduce their results.
  - ◆ A “politically correct” conclusion is no excuse for poor scholarship.
  - ◆ Erudite-sounding technical terminology and fancy mathematical formulas are no substitutes for sound reasoning.
  - ◆ Hype has no place in the scientific enterprise.
-

# Semiconductor Device Modeling Meets Parallel Computing

---

- ◆ Laws of physics no longer permit clock rates to be increased significantly – future performance increases will come only by adopting massively multi-core designs.
- ◆ Systems used for modeling are subject to this same reality.
- ◆ Increasingly complex models, employing advanced physics and high-resolution grids, require correspondingly high performance.
- ◆ The exponentially increasing complexity and density of the devices being modeled require higher performance in modeling.

Conclusion: Device modeling is being dragged “kicking and screaming” into the world of highly parallel computing.

Those vendors (and their customers) who do not adapt risk being left behind.

---

# History of Parallel Computing

---

- ◆ 1976-1986: Initial research studies and demos.
- ◆ 1986-1990: First large-scale systems deployed.
- ◆ 1990-1994: Successes over-hyped; faults ignored. Shoddy measurement methods used. Questionable performance claims made.
- ◆ 1994-1998: Numerous firms fail; agencies cut funds.
- ◆ 1998-2002: Reassessment.
- ◆ 2002-2009: Recovering?
- ◆ 2009: 1 Pflop/s ( $10^{15}$  floating-point operations per second) demonstrated on a large scientific computation.

Have lessons been learned? Or slipping again into hype?

---

# Parallel System Performance Practices, circa 1993

---

- ◆ Performance results on small-sized parallel systems were linearly scaled to full-sized systems.
    - ◇ Example: 8,192-CPU results were linearly scaled to 65,536-CPU results, simply by multiplying by 8.
    - ◇ Rationale: “We can’t afford a full-sized system.”
    - ◇ Sometimes this was done without any clear disclosure in the paper or presentation.
-

# Parallel System Performance Practices, circa 1993

---

- ◆ Highly tuned programs were compared with untuned implementations on other systems.
    - ◇ In comparisons with vector systems, often little or no effort was made to tune the vector code.
    - ◇ This was the case even for comparisons with SIMD parallel systems – here the SIMD code can be directly converted to efficient vector code.
-

# Parallel System Performance Practices, circa 1993

---

- ◆ Inefficient algorithms were employed, requiring many more operations, in order to exhibit an artificially high Mflop/s rate.
    - ◇ Some scientists employed explicit PDE schemes for applications where implicit schemes were known to be much better.
    - ◇ One paper described doing a discrete Fourier transform by direct computation, rather than by using an FFT ( $8n^2$  operations rather than  $5n \log_2 n$ ).
-

# Parallel System Performance Practices, circa 1993

---

- ◆ Performance rates on 32-bit floating-point data on one system were compared with rates on 64-bit data on other systems.
    - ◇ Using 32-bit data instead of 64-bit data effectively doubles data bandwidth, thus yielding artificially high performance rates.
    - ◇ Some computations can be done safely with 32-bit floating-point arithmetic, but many (most?) cannot.
    - ◇ In some emerging applications, even 64-bit floating-point arithmetic is not enough – 128-bit is required.
-

# Parallel System Performance Practices, circa 1993

---

- ◆ In some cases, performance experiments reported in published results *were not actually performed*.
    - ◇ Abstract of published paper:

“The current Connection Machine implementation runs at 300-800 Mflop/s on a full [64K] CM-2, or at the speed of a single processor of a Cray-2 on 1/4 of a CM-2.”
    - ◇ Buried in text:

“This computation requires 568 iterations (taking 272 seconds) on a 16K Connection Machine.”

*In other words, the computation was not run on a full 64K CM-2.*

“In contrast, a Convex C210 requires 909 seconds to compute this example. Experience indicates that for a wide range of problems, a C210 is about 1/4 the speed of a single processor Cray-2, ...”

*I.e., the computation was not run on a Cray-2 – it was run on a Convex system, and a highly questionable scaling factor was used.*
-

# Parallel System Performance Practices, circa 1993

---

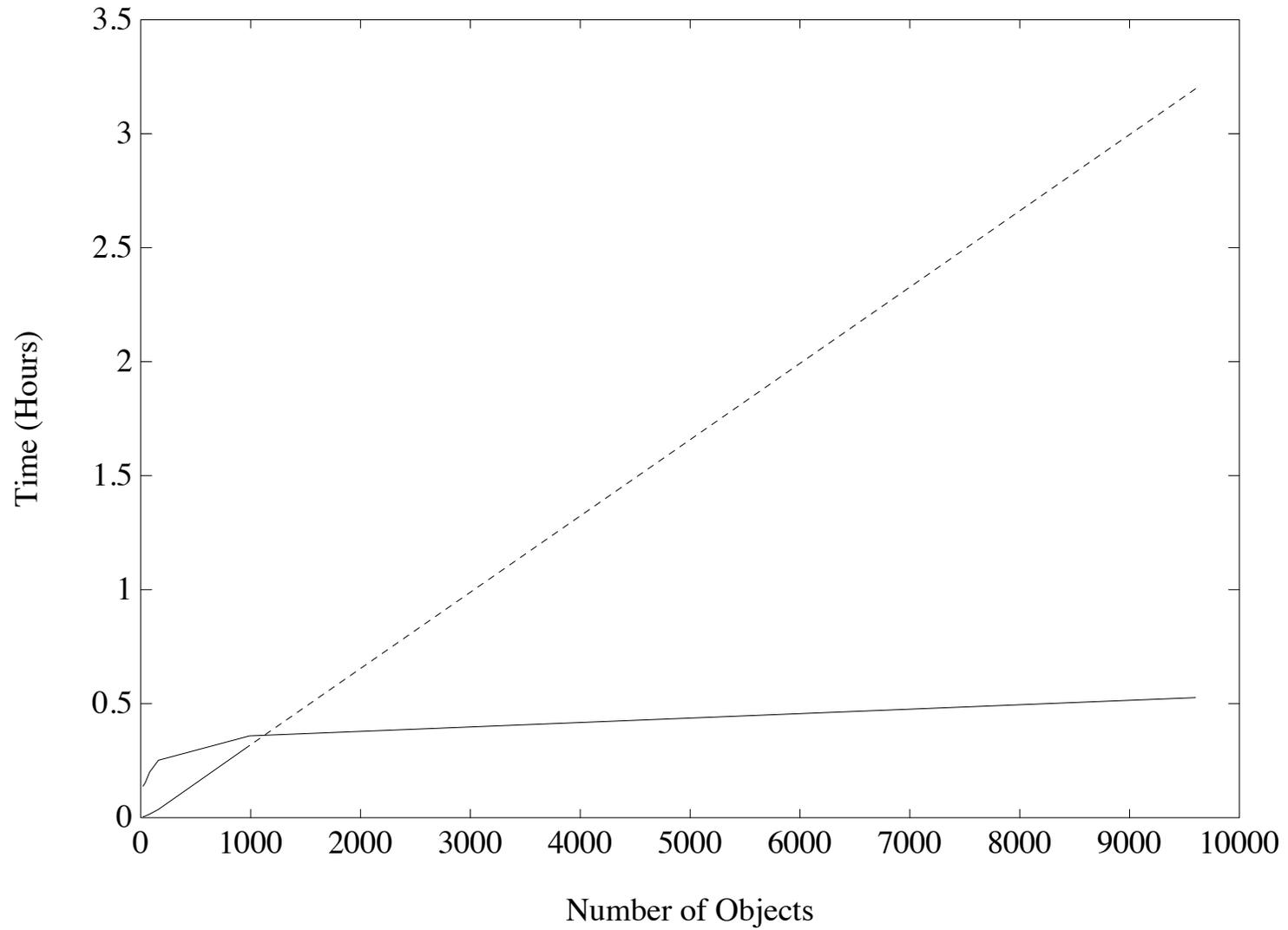
- ◆ Scientists were just as guilty as commercial vendors of questionable performance claims.
  - ◇ The examples in my files were mostly written by professional scientists and published in peer-reviewed journals and conference proceedings.
  - ◇ One example is from an award-winning paper.
  - ◇ Scientists in some cases accepted free computer time or research funds from vendors, but did not disclose this fact in their papers.

Scientists should be held to a higher standard than vendor marketing personnel.

---

# Performance Plot A

---



# Data for Plot A

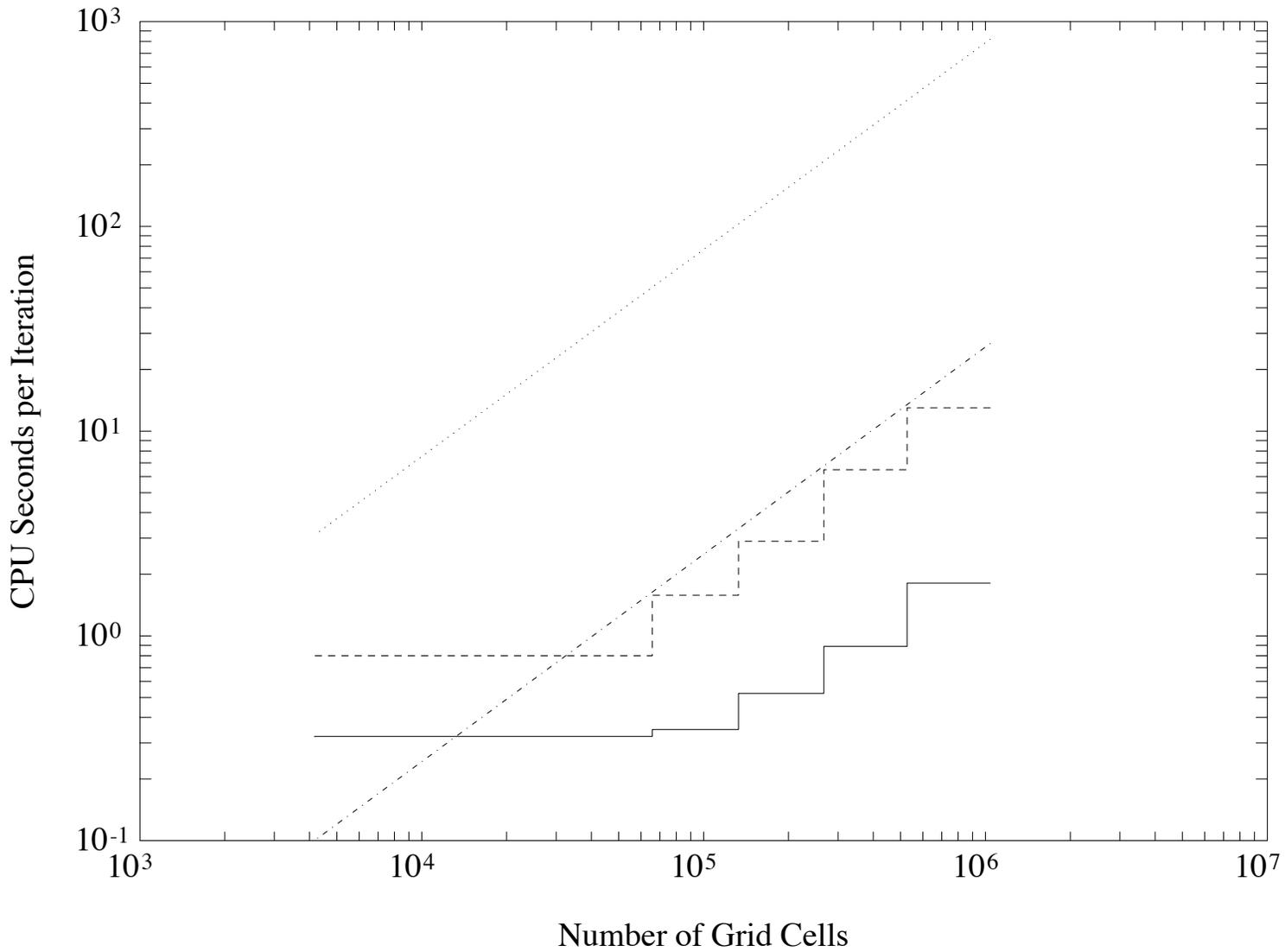
---

Total Objects	Parallel system Run time	Vector system Run time
20	8:18	0:16
40	9:11	0:26
80	11:59	0:57
160	15:07	2:11
990	21:32	19:00
9600	31:36	3:11:50*

## Notes:

- ◆ In last entry, the 3:11:50 figure is an “estimate.”
  - ◆ The vector system code is “not optimized.”
  - ◆ The vector system performance is better except for the last (estimated) entry.
-

# Performance Plot B



# Facts for Plot B

---

- ◆ 32-bit performance rates on a parallel system are compared with 64-bit performance on a vector system.
- ◆ Parallel system results are linearly extrapolated to a full-sized system from a small system (only 1/8 size).
- ◆ The vector version of code is “unvectorized.”
- ◆ The vector system “curves” are straight lines – i.e., they are linear extrapolations from a single data point.

## Summary:

It appears that of all points on four curves in this plot, at most four points represent real timings.

---

# Twelve Ways to Fool the Masses

---

1. Quote only 32-bit performance results, not 64-bit results.
  2. Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application.
  3. Quietly employ assembly code and other low-level language constructs.
  4. Scale up the problem size with the number of processors, but omit any mention of this fact.
  5. Quote performance results projected to a full system.
  6. Compare your results against scalar, unoptimized code on conventional systems.
  7. When direct run time comparisons are required, compare with an old code on an obsolete system.
  8. If Mflop/s rates must be quoted, base the operation count on the parallel implementation, not on the best sequential implementation.
  9. Quote performance in terms of processor utilization, parallel speedups or Mflop/s per dollar.
  10. Mutilate the algorithm used in the parallel implementation to match the architecture.
  11. Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment.
  12. If all else fails, show pretty pictures and animated videos, and don't talk about performance.
-

# Twelve Ways: Basic Principles

---

- ◆ Use well-understood, community-defined metrics.
- ◆ Base performance rates on operation counts derived from the best practical serial algorithms, not on schemes chosen just to exhibit artificially high Mflop/s rates on a particular system.
- ◆ Use comparable levels of tuning.
- ◆ Provide full details of experimental environment, so that performance results can be reproduced by others.
- ◆ Disclose any details that might affect an objective interpretation of the results.
- ◆ Honesty and reproducibility should characterize all work.

**Danger:** We can fool ourselves, as well as others.

---

# New York Times, 22 Sept 1991

## Technology

# Measuring How Fast Computers Really Are

By JOHN MARKOFF

**I**N the world of scientific and technical computing, everyone agrees that computer speeds are increasing at a geometric rate. But measuring that speed is a vexing task. Rival supercomputer and work station manufacturers are prone to hype, choosing the performance figures that make their own machines look best.

"It's like the Wild West," said David J. Kuck, of the Center for Supercomputing Research and Development at the University of Illinois. "They say whatever they want to."

In fact, said David H. Bailey, a scientist at the National Aeronautics and Space Administration, "It's not really to the point of widespread fraud, but if people aren't a little more circumspect, the entire field could start to get a bad name."

The matter is complicated by a new generation of computers that have dozens, or even thousands, of separate processors. These parallel computers split problems into small parts and solve them simultaneously to reach greater speeds.

As a result, dozens of programs for determining benchmarks — measurements of computer speed — have been developed by scientists at universities and in government agencies. Some are based on how long a computer takes to solve a certain set of equations, while more sophisticated benchmarks attempt to match the operations re-

quired by real-world programs. But each benchmark generally measures only a single aspect of computer performance.

Just as a car buyer might buy a vehicle with the highest E.P.A. gas mileage rating for the price, a computer buyer could use benchmarks in deciding which machine to buy. But like their counterparts in the auto business, computer makers would do well to remind customers, "Your mileage may vary." The industry has no independent organization, analogous to the Environmental Protection Agency, to establish a single standard.

The proliferation of benchmarks is particularly problematic among the fastest scientific machines, where more than a dozen start-up companies compete to sell to university, corporate and Government laboratories.

These machines sell for hundreds of thousands of dollars or more, and the sale of only a few can mean success for a company. Supercomputers and smaller scientific work stations work on problems ranging from designing pharmaceuticals and weapons to weather modeling and the simulated crashing of automobiles.

Uneasy about the tendency for manufacturers to cite inflated claims, Mr. Bailey of NASA wrote a tongue-in-cheek indictment of performance claims for Supercomputing Review magazine in August. Titled "Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers," it pokes fun at the tendency of computer mak-

## Different Benchmarks, Different

The six fastest computers according to various benchmark point operations per second. Slalom, the only one accomplished in a set amount of time. The Perf

### LINPACK

<b>Cray Y-MP/16</b>	<b>403</b>
<b>NEC SX-3/14</b>	<b>314</b>
<b>Cray Y-MP/832</b>	<b>275</b>
<b>Fujitsu VP2600/10</b>	<b>249</b>
<b>Cray X-MP/416</b>	<b>178</b>
<b>Cray 2S/4-128</b>	<b>129</b>

Sources: Oak Ridge National Laboratory, Supercomputing Review, University of Illinois, University of Tennessee

ers to play fast and loose with speed claims.

It is common practice to "tune" computers and software to score better on benchmarks. "I know of a couple of companies who have full-time people, and all they do is optimize programs to achieve better benchmark results," said Gary Smaby, president of the Smaby Group, a consulting and market research firm in Minneapolis.

Such optimization is permissible under the rules established by benchmark designers to insure that computer makers can extract the full capability from their systems.

But some manufacturers go further and insert modules called "recognizers" into their compilers — software that translates a

# Excerpts from NYT Article

---

“Rival supercomputer and work station manufacturers are prone to hype, choosing the performance figures that make their own systems look better.”

“It’s not really to the point of widespread fraud, but if people aren’t somewhat more circumspect, it could give the field a bad name.”

---

# Fast Forward to 2009: Five New Ways to Fool the Masses

---

- ◆ Dozens of runs are made, but only the best performance figure is cited in the paper.
  - ◆ Runs are made on part of an otherwise idle system, but this is not disclosed in the paper.
  - ◆ Performance rates are cited for a run with only one CPU active per node.
  - ◆ Special hardware, operating system or compiler settings are used that are not appropriate for real-world usage.
  - ◆ “Scalability” is defined as a successful execution on a large number of CPUs, regardless of performance.
-

# The Role of Good Benchmarks in Combating Performance Abuse

---

- ◆ Well-designed, rigorous, scalable performance benchmark tests help bring order to the field.
- ◆ Well-thought-out and well-enforced “ground rules” are essential.
- ◆ A rational scheme must be provided for calculating performance rates.
- ◆ A well-defined test must be included to validate the correctness of the results.
- ◆ A repository of results must be maintained.

Recent example: The HPCS benchmark suite.

---

# Guidelines to Prevent Abuse

---

1. If results are presented for a well-known benchmark, comparative figures should be truly comparable, and rules should be followed.
  2. Only actual performance results should be presented, not projections or extrapolations (unless very clearly disclosed and justified).
  3. Performance figures should be based on comparable tuning.
  4. Direct comparisons of run times are preferred to Mflop/s rates, etc.
  5. Mflop/s or Gflop/s rates should be computed from consistent operation counts, preferably based on the best available serial algorithms.
  6. If speedup figures are presented, the single processor rate on which they are based should be based on an efficient implementation.
  7. Any ancillary information that would affect the interpretation of the results should be fully disclosed (e.g., 32-bit data vs 64-bit data, etc.).
  8. Special care should be taken for figures and graphs.
  9. Whenever possible, full background information should be provided: hardware and software configuration; language; algorithms; datatypes; tuning; timing method; basis for operation counts and speedup figures; etc.
-