Misleading Performance Claims in Parallel Computations

David H. Bailey [*]

July 6, 2009

Abstract

In a previous humorous note entitled "Twelve Ways to Fool the Masses ... ," I outlined twelve common ways in which performance figures for technical computer systems can be distorted. In this paper and accompanying conference talk, I give a reprise of these twelve "methods" and give some actual examples that have appeared in peer-reviewed literature in years past. I then propose guidelines for reporting performance, the adoption of which would raise the level of professionalism and reduce the level of confusion, not only in the world of device simulation but also in the larger arena of technical computing.

1 Introduction

Some readers may have read (or heard about) a tongue-in-cheek article I wrote entitled "Twelve Ways to Fool the Masses When Giving Performance Reports on Parallel Computers" [1]. This article attracted an astonishing amount of attention at the time, including mention in the *New York Times* [5]. Evidently it struck a responsive chord among many professionals in the field of technical computing who shared my concerns. The following is a very brief summary of the "Twelve Ways":

1. Quote only 32-bit performance results, not 64-bit results, and compare your 32-bit results with others' 64-bit results.
2. Present inner kernel performance figures as the performance of the entire application.

[*]Computational Research Dept., Lawrence Berkeley National Laboratory, Berkeley, CA 94720, `dhbailey@lbl.gov`. Supported in part by the Director, Office of Computational and Technology Research, Division of Mathematical, Information and Computational Sciences, U.S. Department of Energy, under contract number DE-AC02-05CH11231.

3. Quietly employ assembly code and other low-level language constructs, and compare your assembly-coded results with others' Fortran or C implementations.
4. Scale up the problem size with the number of processors, but don't clearly disclose this fact.
5. Quote performance results linearly projected to a full system.
6. Compare your results against scalar, unoptimized, single processor code on Crays [prominent vector computer systems at the time].
7. Compare with an old code on an obsolete system.
8. Base megaflops operation counts on the parallel implementation instead of on the best sequential implementation.
9. Quote performance in terms of processor utilization, parallel speedups or megaflops per dollar (peak megaflops, not sustained).
10. Mutilate the algorithm used in the parallel implementation to match the architecture. In other words, employ algorithms that are numerically inefficient in order to exhibit artificially high megaflops rates.
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.

Both at the time this article appears, and in the intervening years, some have suggested that my intent in publishing the "Twelve Ways" article was to criticize computer vendors for their sometimes exuberant claims of performance. There have been some instances of dubious claims of performance by computer vendors, but my primary targets were scientists and engineers themselves, when presenting performance results of their applications. This is also the main focus of this article. Here I present a brief summary of some of the most common abuses, and then close by presenting some principles that will help reduce confusion in the field. Portions of this material are condensed from a previous paper [2].

2 The Need for Professional Discipline

We do not need to look very far to see examples in other fields where well-meaning but sloppy practices in reporting experimental results have led to distortion and major embarrassment. One amusing example is the history of measurements of the speed of light [4]. Accurate measurements performed in the late 1930s and 1940s seemed to be converging on a value of roughly 299,760 km/s. After World War II, some researchers made more careful measurements, and thereafter the best value converged to its present-day value of 299,792.458 km/s (which is now taken as a

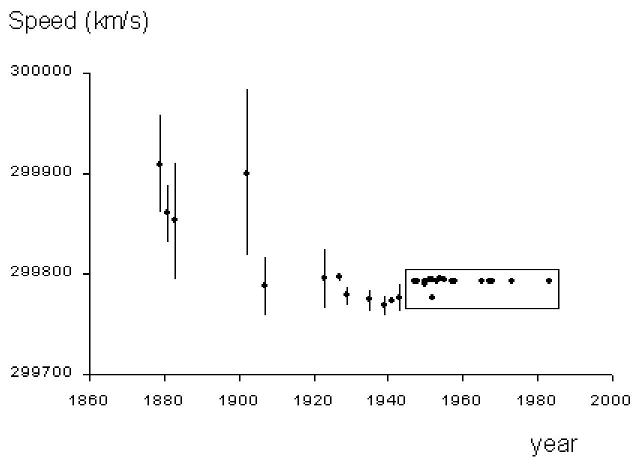

Figure 1: History of measurements of the speed of light

standard, indirectly defining the meter and the second). Certainly few researchers seriously believe that the speed of light changed over a few years. Thus we are left with the conclusion that sloppy experimental practices and possibly some form of self-deluding "group-think" led researchers to converge on the earlier (incorrect) value.

Difficulties with experimental methods and dispassionate assessment of results have also been seen in the social sciences. In his book *The Blank Slate: The Modern Denial of Human Nature*, Harvard psychologist Steven Pinker chronicles the fall of the "blank slate" paradigm of the social sciences, namely the assertion that heredity and biology play no significant role in human psychology—all personality and behavioral traits are socially constructed. This paradigm prevailed in the field until the 1980s, when indisputable empirical evidence forced a change. The current consensus, based on latest research, is that humans at birth universally possess sophisticated facilities for language acquisition, pattern recognition and social life, and that heredity, evolution and biology are major factors in human personality—some personality traits are as much as 70% heritable.

Along this line, anthropologists, beginning with Margaret Mead in the 1930s, painted an idyllic picture of primitive societies such as South Sea Islanders, claiming that they had little of the violence, jealousy, warfare or social hangups that afflict Western societies. Beginning in the 1980s, a new breed of anthropologists began to re-examine these findings. They found, contrary to earlier results, that these societies typically had murder rates several times higher than large U.S. cities, and death rates from inter-tribe warfare exceeding those of warfare among Western nations by factors of 10 to 100. What's more, complex, jealous taboos surrounded courtship and marriage—some even condoned violent reprisals if a bride were found not to be a virgin on her wedding night.

How did these scientists get it so wrong? Pinker and others have concluded that the principal culprit was sloppy experimental methodology and wishful-thinking analysis of results [6].

3 The Pressure for Reporting High Performance

For many years, parallel computers were almost exclusively the province of universities and government laboratories. More recently, processor manufactures such as Intel and Advanced Micro Devices, recognizing that basic physical laws no longer permit them to continue increasing the clock frequency, are instead offering increased performance by incorporating multiple processing cores on a single chip. In other words, like it or not even single-user personal computers are now parallel computers, and engineering workstations often incorporate 16, 32 or more processing cores. The message to both end users and to third-party technical software firms is clear: adopt your codes to run on parallel systems or risk being left behind.

The field of semiconductor device modeling and system engineering is no exception. In fact, the exploding complexity of designs by itself is requiring engineers to utilize the most powerful systems available for all stages of design, ranging from basic device physics to chip layout to full-system simulation. What's more, the increasing emphasis on reducing time to market by itself is a strong incentive to utilize highly parallel systems wherever possible. The price of failing to utilize this technology is shown by such unfortunate episodes as the Pentium divide fiasco, which in the end cost Intel over $500 million.

When this external pressure is added to the natural human tendency of scientists and engineers to be exuberant about their own work, it should come as little surprise that some have presented sloppy and potentially misleading performance claims in papers and conference presentations. And since the reviewers of these papers are themselves in many cases caught up in the excitement of this new technology, it should not be surprising that they have tended to be relatively permissive with questionable aspects of these papers.

Clearly the field of technical computing does not do itself a favor by condoning inflated performance reports, whatever are the motives of those involved. In addition to fundamental issues of ethics and scientific accuracy, the best way to insure that computer systems are effective for engineering design or other applications is to provide early feedback to manufacturers regarding their weaknesses. Once the reasons for less-than-expected performance rates on certain problems are identified, improvements can be made in the next generation.

In the next section, I will present several examples of questionable performance reporting that have appeared in peer-reviewed papers. I confess that these examples are a bit "dated" at this point in time, but they are nonetheless instructive in pointing out practices that should be avoided. My only purpose in citing these examples is to provide concrete instances of the performance issues in question. I do

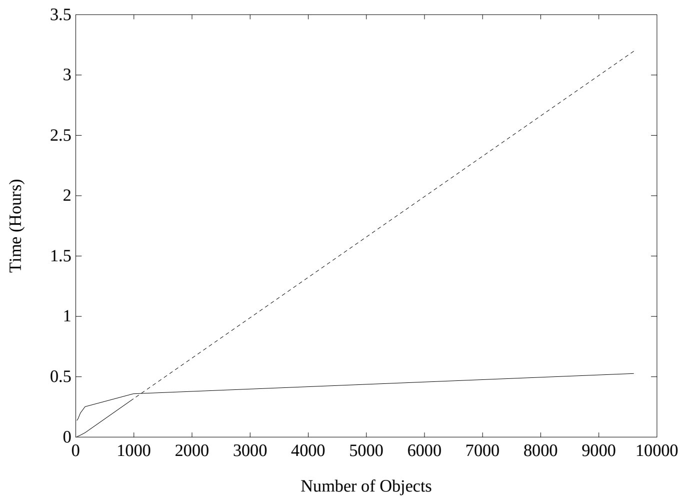

Figure 2: Timings of parallel (lower) and single-processor vector (upper) on a defense application

not wish for these citations to be misconstrued as criticisms of individual scientists, laboratories or supercomputer manufacturers. This is because I personally do not believe this problem is restricted to a handful of authors, laboratories and vendors, but that many of us in the field must share blame. For this reason I do not include references for these papers.

4 Plots

Plots can be effective vehicles to present technical information, particularly in verbal presentations. Further, plots are in many cases all that high-level managers (such as those with authority for computer acquisitions) have time to digest. Unfortunately, plots can also mislead an audience, especially if prepared carelessly or if presented without important qualifying information.

Figure 2 is a reconstruction of the final performance plot from a paper describing a defense application. The plot compares timings of the authors' parallel system code with timings of a comparable code running on a single-processor vector system. The plot appears to indicate an impressive performance advantage for the parallel system, on all problem sizes except perhaps a small region at the far left.

However, examination of the raw data used for this plot, which is shown in Table 1, gives a different picture. First of all, except for the largest problem size, all data points lie in the small region at the far left. In other words, most of the two curves shown are merely the linear connections of the next-to-last data points with the final points. Further, the single-processor vector system is actually faster than the parallel system for all sizes except for the largest problem size. At the least, it is clear that a logarithmic scale would have been far more appropriate for this data.

Total Objects	Parallel Run Time	Vector 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

Table 1: Raw data for plot in Figure 2. * denotes estimate.

Other difficulties are encountered when one reads the text accompanying this graph and table. First of all, the authors concede that the runs on the vector system used a code that "has not been optimized" for that system. Secondly, for the largest problem listed, the only one where the vector fails to out-perform the parallel system, the vector timing is, by the authors' admission, an estimate, an extrapolation based on a smaller run. In the paper, as in Figure 2, the lower curve leading out to the last point is dashed, possibly intending to indicate that this is an estimate, but this feature is not explained in either the caption or the text.

Figure 3 is a reconstruction of another performance plot from a paper describing a fluid dynamics application. This plot compares timings of the author's codes running on a parallel system with those of comparable codes running on a single-processor vector system. The two curves shown for each computer system represent a structured and an unstructured grid version of the code, respectively. As before, the plot appears to indicate a substantial performance advantage for the parallel system for all problem sizes and both types of grids.

Once again, careful examination of the text accompanying this plot places these results in a different light. First of all, the author admits that his parallel results have been linearly extrapolated to a full-sized system system from a smaller system. The author also acknowledges explains that the vector version of the unstructured code has not been tuned for vector computation. An additional difficulty with this plot can be seen by carefully examining the two vector system curves. In the original, as in Figure 2, these are precisely straight lines. Needless to say, it is exceedingly unlikely that a vector system code, scaled over nearly three orders magnitude in problem size, exhibits precisely linear timings. Thus one has to suspect that the two vector system "curves" are simply linear extrapolations from single data points. In summary, it appears that of all points on four curves in this plot, at most two points are clearly real timings. Also, it appears that the author is comparing 32-bit floating-point performance on the parallel system with 64-bit floating-point performance on the vector system.

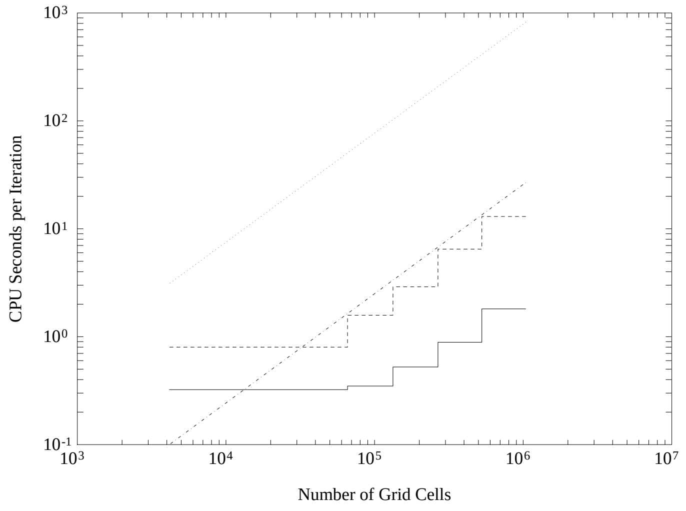

Figure 3: Timings of parallel system (solid and dashes) and single-processor vector system (dash-dots and dots) on a fluid dynamics application

5 Projections and Extrapolations

The practice of citing estimated and extrapolated performance results is, unfortunately, fairly widespread in the field. This may in part be an unintended consequence of limited budgets at many research institutions and private firms, where scientists and engineers often have to settle for scaled-down versions of highly parallel systems. As a result, authors frequently cite performance results that are merely linear projections from much smaller systems, often without the slightest justification.

The practice of linearly extrapolating one's performance results to a larger system is doubly perplexing because the question of whether various computer designs and applications will "scale" is in fact an important topic of current research in the field of parallel computing. It seems that many scientists and engineers using parallel computers are willing to assume as an established fact one of the most fundamental questions in the field!

We have already seen one instance of citing extrapolated results. In another paper in my files, the authors compare their application running on one moderately parallel system with comparable codes running on a single-processor vector system and a massively parallel system. Fortunately, all of the moderately parallel timings in the three tables are real timings. But out of a total of 33 figures listed for the vector system and the massively parallel system, more than half (17) are merely projections or estimates. There does not appear to be any attempt to mislead the reader, since the authors indicate which figures in each table are projections by means of asterisks. Nonetheless, one is left to wonder about how reliable these comparisons are, and whether they will always be quoted with the appropriate disclaimer.

Some authors have taken the practice of citing projections one step further. In one peer-reviewed paper in my files, the author states in his abstract that his code runs "at the speed of a single processor of a Cray-2 [a 1986 vintage 4-processor vector computer] on 1/4 of a CM-2 [a 1990 vintage massively parallel computer]". Some thirteen pages later, the author cites a timing on a Convex C210 [a 1990 vintage minicomputer] and then states "experience indicates that for a wide range of problems, a C210 is about 1/4 the speed of a single processor Cray-2." No further mention is made of the Cray-2.

It is well known that for any computer system, timings and megaflops rates can vary dramatically depending on how much effort has been expended in tuning for the particularly architecture being used. Thus any blanket performance ratio such as 1/4 is dubious. But the most troubling item here is the fact that the author, in the abstract of his paper, clearly implies a performance comparison with a Cray-2, even though he evidently has never run his code on a Cray-2.

6 Counting Flops

A common practice in the field of scientific computing is to cite performance rates in terms of millions of floating point operations per second (megaflops) or in billions of floating-point operations per second (gigaflops). For various reasons, some in the field have suggested that the practice of citing megaflops or gigaflops be abandoned. However, I am of the opinion that while direct timing comparisons are always preferred, megaflops or gigaflops rates may be cited if calculated and reported consistently.

Part of the confusion derives from the method used to determine the number of floating point operations (flops) performed. Many authors count the number of flops actually performed in their parallel implementations, a number usually obtained by analyzing the parallel source code. However, parallel implementations almost always perform significantly more flops than serial implementations. For example, some calculations are merely repeated in each processor. Using the actual number of flops performed on a parallel computer thus results in megaflops rates that are inflated when compared to rates obtained from corresponding single-processor implementations.

Another difficulty with basing megaflops or gigaflops rates on the actual parallel flop count is that this practice tacitly encourages researchers to employ numerically inefficient algorithms in their applications, algorithms often chosen mainly for the convenience of the particular architecture being used. It is easy to understand how such choices can be made, since it is widely accepted in the field that algorithmic changes are often necessary when porting a code to a parallel computer. But when this practice is carried too far, both the audience and the scientist may be misled.

Because of the potential for misleading comparisons, it is clear that a single standard flop count should be used when comparing rates for a given application. In my view, the most sensible flop count for this purpose is the minimal flop count — the value based on an efficient implementation of the best practical serial algorithms. In this way, one is free to use an implementation with a higher flop count on a particular architecture if desired, but no extra credit is given for these extra operations when megaflops rates are computed. This standard also acts as a deterrent to the usage of numerically inefficient algorithms.

7 Other Issues

Many authors report "speedup" figures for their parallel applications. Such figures indicate the degree to which the given application "scales" on a particular architecture. However, here also there is potential for the audience to be mislead, especially when speedup figures are based on inflated single processor timings.

For example, users of message passing parallel systems often base speedup figures on a single node timing of the multiple node version of the program. When running on a single node, the multiple node program needlessly synchronizes with itself and passes messages to itself. These "messages" are handled quite rapidly, since the operating system recognizes that these are local transmissions. Nonetheless, a significant amount of overhead is still required, and it is not unusual for the single node run time to increase by 20 percent with the addition of message passing code.

Some authors present "scaled speedup" figures, where the problem size is scaled up with the number of processors. Such figures may be informative, but it is essential that authors who quote such figures clearly disclose the fact that they have scaled their problem size to match the processor count. It is also important that authors provide details of exactly how this scaling was done.

Another aspect of performance reporting that needs to be carefully analyzed is how the authors measure run time. Most of the scientists I have queried about this issue feel that elapsed wall clock time is the most reliable measure of run time, and that if possible it should be measured in a dedicated environment. By contrast, CPU time figures may mask extra elapsed time required for input and output.

One final aspect of performance reporting is the source of untold confusion in the field: are the results for 32-bit or 64-bit floating point arithmetic? Since on many systems, 32-bit computational performance rates are nearly twice as high as 64-bit rates, there is a temptation for authors to quote only 32-bit results, to fail to disclose that rates are for 32-bit data, and to compare their 32-bit results with others' 64-bit results. It is clear that 32-bit/64-bit confusion is widespread in performance reporting, since we have seen several examples already.

In my view, quoting 32-bit performance rates is permissible so long as: (1) this data type is clearly disclosed and (2) a brief statement is included explaining why this precision is sufficient. Along this line, it should be kept in mind that with new much more powerful computer systems, it is now possible to attempt much larger problems than before. Inevitably, these much larger calculations greatly exacerbate any numerical sensitivities that may exist in the code. As a result, numerical issues that previously were not significant now are significant, and some programmers are discovering to their dismay that higher precision is necessary to obtain meaningful results. In fact, there are many applications where even 64-bit floating-point arithmetic is not sufficient, and where much higher precision is required [3].

I suspect that in the majority of cases where the authors do not clearly state the data type, the results are indeed for 32-bit data. One example is an award-winning paper, where the authors never state whether their impressive performance rates are for 32-bit or 64-bit calculations, at least not in any place where a reader would normally look for such information. That their results are indeed for 32-bit data can however be deduced by a careful reading of their section on memory bandwidth, where we read that operands are four bytes long.

8 Proposed Guidelines

Clearly this field needs a detailed set of guidelines for reporting supercomputer performance, guidelines which are formally adopted and widely disseminated to authors and reviewers. Virtually every field of science has found it necessary at some point to establish rigorous standards for the reporting of experimental results, and ours should be no exception. To that end, I propose the following. These guidelines focus on computational performance, since that is the topic of this paper and apparently the most frequent arena of confusion. However, it is hoped that the spirit of these guidelines will be followed by researchers reporting performance in other areas of technical computing, such as in mass storage and local area networks.

1. If results are presented for a well-known benchmark, comparative figures should be truly comparable, and the rules for the particular benchmark should be followed.
2. Only actual performance rates should be presented, not projections or extrapolations. For example, performance rates should not be extrapolated to a full system from a scaled-down system. Comparing extrapolated figures with actual performance figures, such as by including both in the same table, is particularly inappropriate.
3. Comparative performance figures should be based on comparable levels of tuning.
4. Direct comparisons of run times are preferred to comparisons of megaflops rates or the like. Whenever possible, timings should be true elapsed time-of-day measurements (this might not be possible in some "production" environments).
5. Megaflops or gigaflops rates should be computed from consistent flop counts, preferably flop counts based on efficient implementations of the best practical serial algorithms. One intent here is to discourage the usage of numerically inefficient algorithms, which may exhibit artificially high performance rates on a particular parallel system.

6. If speedup figures are presented, the single processor rate should be based on a reasonably well tuned program without multiprocessing constructs. If the problem size is scaled up with the number of processors, then the results should be clearly cited as “scaled speedup” figures, and details should be given explaining how the problem was scaled up in size.
7. Any ancillary information that would significantly affect the interpretation of the performance results should be fully disclosed. For example, if the results are for 32-bit rather than for 64-bit data, or if assembly-level coding was employed, or if only one processor of a conventional system is being used for comparison, these facts should be clearly stated.
8. Due to the natural prominence of abstracts, figures and tables, special care should be taken to insure that these items are not misleading, even if presented alone. For example, if significant performance claims are made in the abstract of the paper, any important qualifying information should also be included in the abstract.
9. Whenever possible, the following should be included in the text of the paper: the hardware, software and system environment; the language, algorithms, the datatypes and programming techniques employed; the nature and extent of tuning performed; and the basis for timings, flop counts and speedup figures. The goal here is to enable other scientists and engineers to accurately reproduce the performance results presented in the paper.

9 Conclusions

The examples I have cited above are somewhat isolated in the literature, and I see no evidence that the problem of inflated performance reporting is out of control. However, clearly those of us in the technical computing field would be wise to arrest any tendency in this direction before we are faced with a significant credibility problem. As was mentioned above, scientists in many other disciplines have found it necessary to adopt rigorous standards for reporting experimental results, and ours should be no exception. It is my hope that this article, with the proposed guidelines above, will stimulate awareness and dialogue on the subject and lead to standards in the field.

References

[1] D. H. Bailey. Twelve ways to fool the masses when giving performance results on parallel computers. *Supercomputing Review*, pages 54–55, August 1991.

[2] D. H. Bailey. Misleading performance reporting in the supercomputing field. *Scientific Programming*, 1:141–151, 1992.

[3] D. H. Bailey. Resolving numerical anomalies in scientific computation. `http://crd.lbl.gov/~dhbailey/dhbpapers/numerical-bugs.pdf`, 2008.

[4] A. N. Cutler. A history of the speed of light. `http://www.sigma-engineering.co.uk/light/lightindex.shtml`, 2001.

[5] J. Markoff. Measuring how fast computers really are. *New York Times*, page 14F, September 1991.

[6] S. Pinker. *The Blank Slate: The Modern Denial of Human Nature*. Viking, New York, 2002.