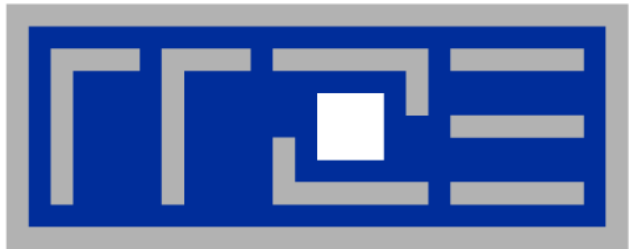


Fooling the Masses with Performance Results: Old Classics & Some New Ideas

Gerhard Wellein[1,2] , Georg Hager[2]

[1]Department for Computer Science
[2]Erlangen Regional Computing Center
Friedrich-Alexander-Universität Erlangen-Nürnberg

Legal disclaimer

The information contained in this talk is for general guidance on matters of interest only. The application and impact of laws can vary widely based on the specific facts involved. Given the changing nature of laws, rules and regulations, and the inherent hazards of electronic communication, there may be delays, omissions or inaccuracies in information contained in this talk. Accordingly, the information in this talk is provided with the understanding that the authors and publishers are not herein engaged in rendering legal, accounting, tax, or other professional advice and services. As such, it should not be used as a substitute for consultation with professional accounting, tax, legal or other competent advisers. Before making any decision or taking any action, you should consult an HPC professional.

While we have made every attempt to ensure that the information contained in this talk has been obtained from reliable sources, we are not responsible for any errors or omissions, or for the results obtained from the use of this information. All information in this talk is provided "as is", with no guarantee of completeness, accuracy, timeliness or of the results obtained from the use of this information, and without warranty of any kind, express or implied, including, but not limited to warranties of performance, merchantability and fitness for a particular purpose. In no event will we, our related partnerships or corporations, or the partners, agents or employees thereof be liable to you or anyone else for any decision made or action taken in reliance on the information in this talk or for any consequential, special or similar damages, even if advised of the possibility of such damages.

Certain links in this talk connect to other websites maintained by third parties over whom we have no control. We make no representations as to the accuracy or any other aspect of information contained in other talks, websites, or papers.

And finally, we take no responsibility whatsoever for the consequences of you showing these slides around and getting **spanked** by your boss, your peers, your spouse, your kids, your mother, or anyone who might be offended because they don’t get the inherent irony. So there.

Fooling the masses with performance results: The history

Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers

David H. Bailey

June 11, 1991

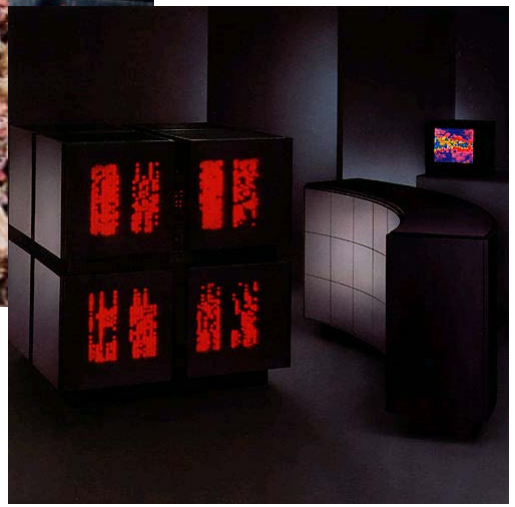
Ref: Supercomputing Review, Aug. 1991, pg. 54--55

Abstract

Many of us in the field of highly parallel scientific computing recognize that it is often quite difficult to match the run time performance of the best conventional supercomputers. This humorous article outlines twelve ways commonly used in scientific papers and presentations to artificially boost performance rates and to present these results in the "best possible light" compared to other systems.

1991 …

If you were plowing a field, which would you rather use?



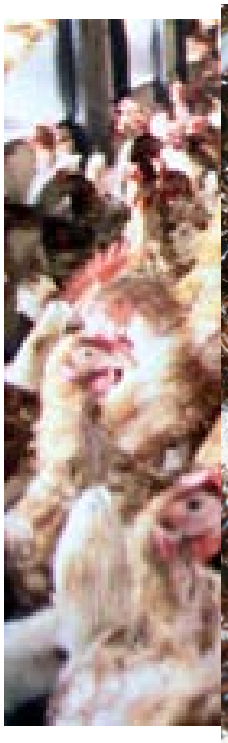
Two strong oxen
or 1024 chickens?

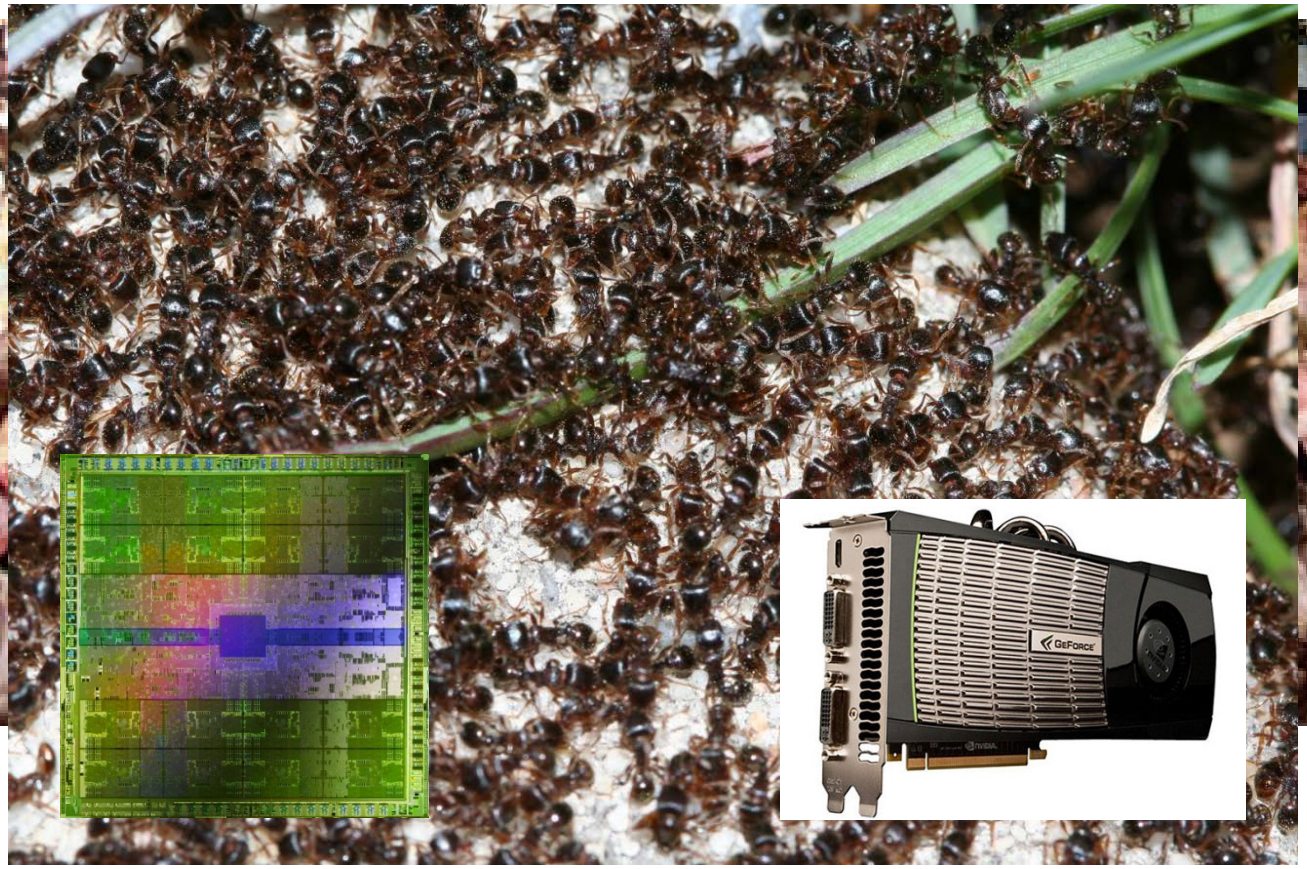
(Attributed to Seymour Cray)

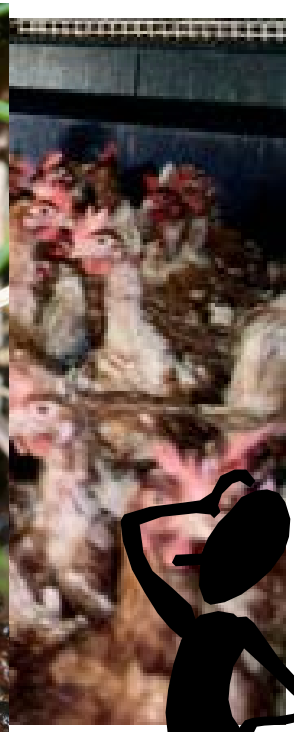
Today we have…

Ants all over the place

GPGPUs, Intel Xeon/Phi, ARM... Some already gone…







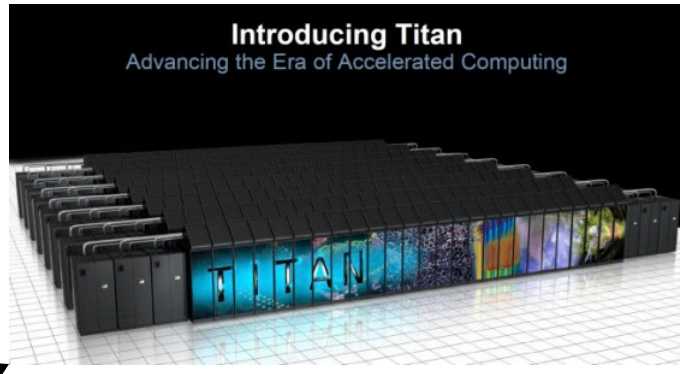
“Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers”

***David H. Bailey*, Supercomputing Review, August 1991, p. 54-55**

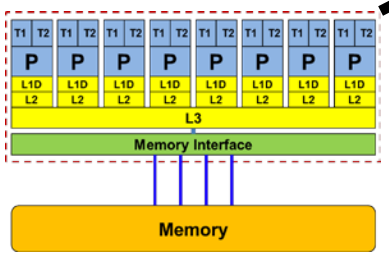
1. Quote only 32-bit performance results, not 64-bit results.
2. Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application.
3. Quietly employ assembly code and other low-level language constructs.
4. Scale up the problem size with the 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 Crays.
7. When direct run time comparisons are required, compare with old code on an obsolete system.
8. If MFLOPS 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 MFLOPS 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.

The landscape of HPC and the way we think about HPC has changed over the last 2 decades, and **we present an update**!

Still, most of Bailey's points are valid without change



Scalability matters!



Scalability matters!

Report scalability,

Parallel Speedup:

$$S(N) = \frac{\text{work/time with } N \text{ workers}}{\text{work/time with } 1 \text{ worker}}$$

“Good” scalability ↔ *S*(*N*) ≈ *N*

Frequent Assumption:

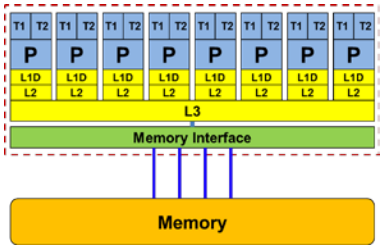
If your code does not scale you can not use current or next generation parallel computers → modern supercomputers have 10^6+ cores!

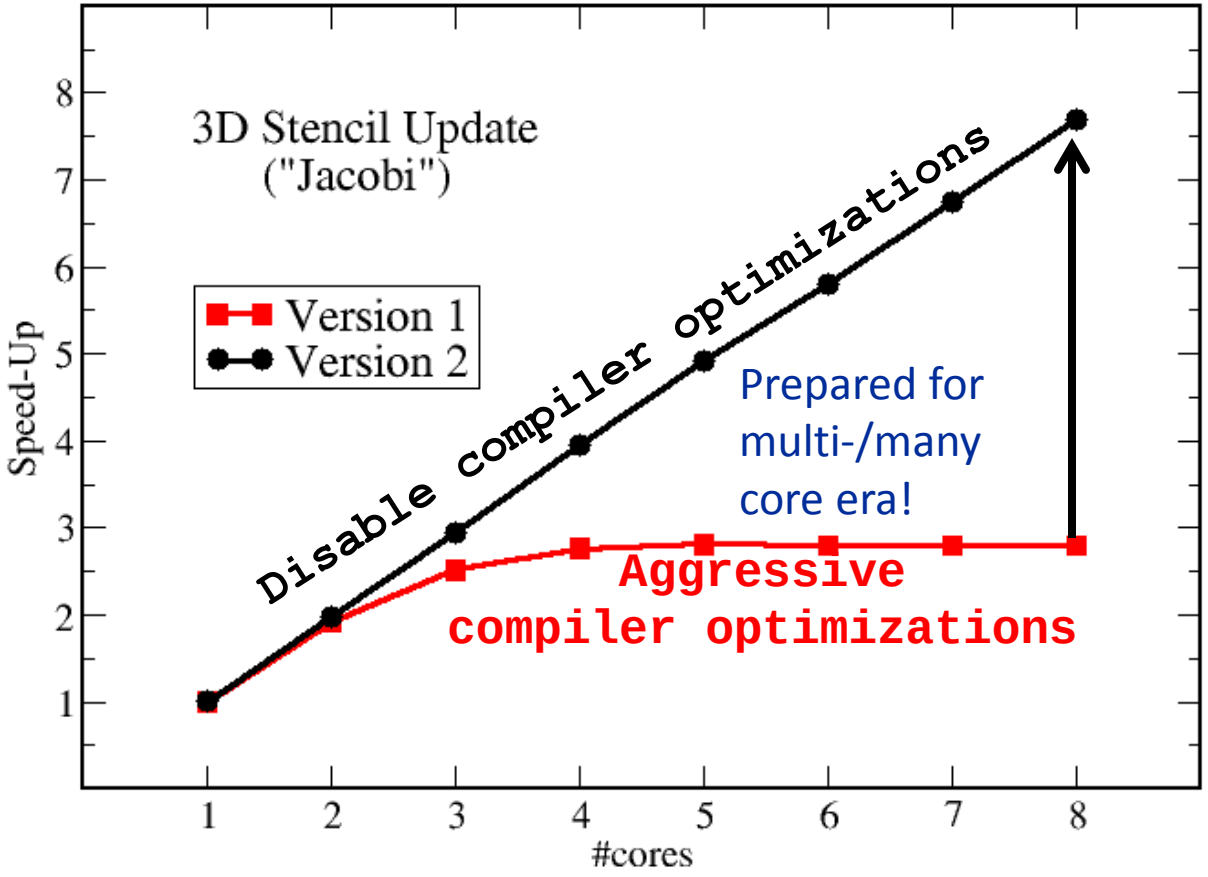
→ Make your code scale

Scalability matters!

```
!$OMP PARALLEL DO
do k = 1 , Nk
  do j = 1 , Nj; do i = 1 , Ni
     y(i,j,k)= b*(    x(i-1,j,k)+ x(i+1,j,k)+ x(i,j-1,k)+
                      x(i,j+1,k)+ x(i,j,k-1)+ x(i,j,k+1))
    enddo; enddo
enddo
```

There is no reason that applications on multicore processors do not scale!



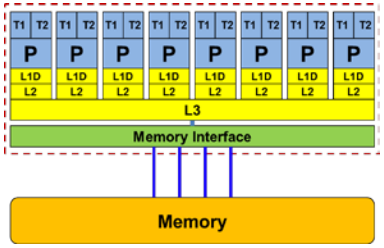


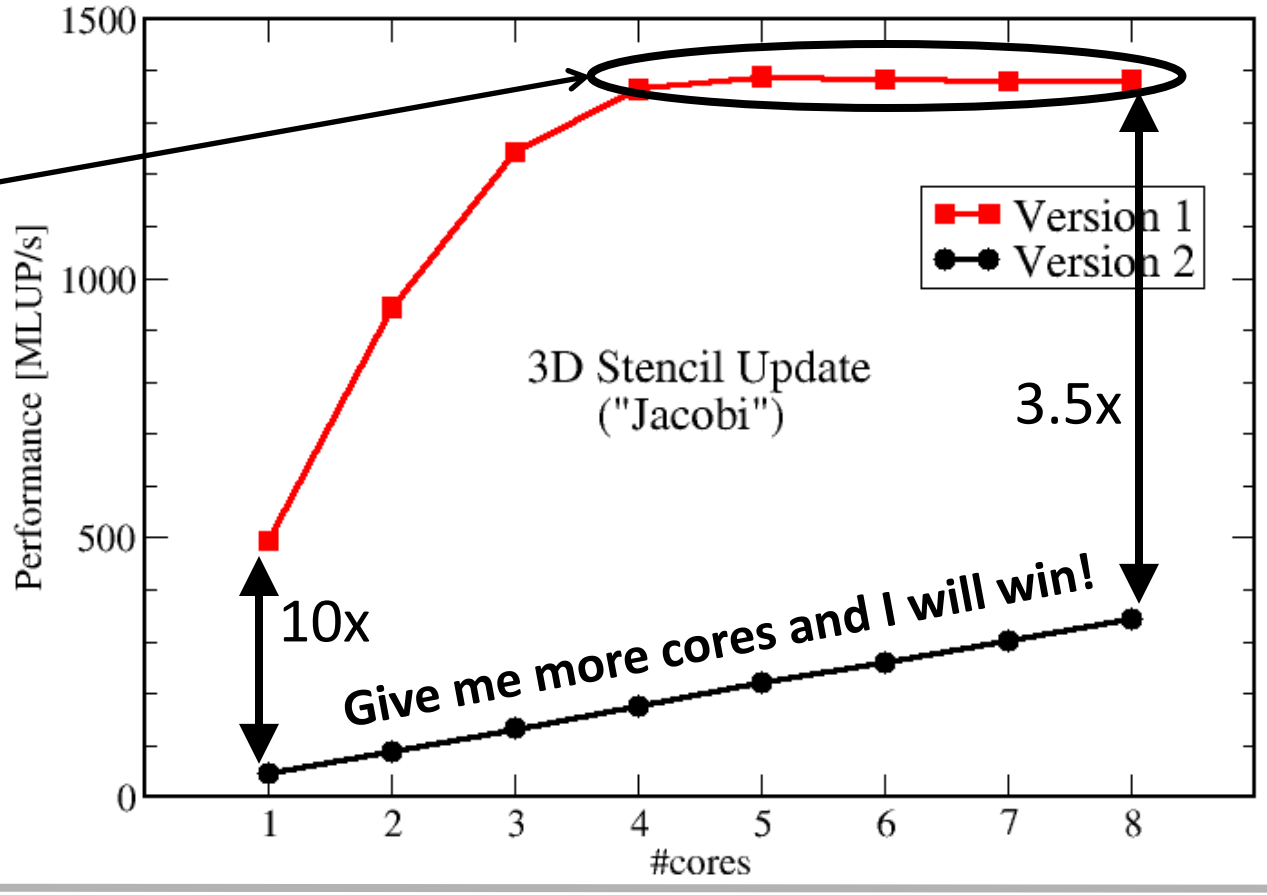
Scalability matters!

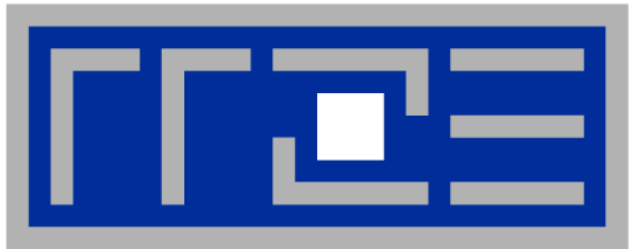
```
!$OMP PARALLEL DO
do k = 1 , Nk
  do j = 1 , Nj; do i = 1 , Ni
     y(i,j,k)= b*(    x(i-1,j,k)+ x(i+1,j,k)+ x(i,j-1,k)+
                      x(i,j+1,k)+ x(i,j,k-1)+ x(i,j,k+1))
    enddo; enddo
enddo
```

Is this the maximum performance ?!

→ Our tutorial last Sunday







Slow down code execution!

Slow down code execution!

This improves scalability whenever there is some noticeable "non-execution" overhead, e.g. communication

Parallel speedup with work ~ N^{α}:
(α=0: strong, α=1: weak scaling)

$$S(N) = \frac{s + (1-s)N^{\alpha}}{s + (1-s)N^{\alpha-1} + c_{\alpha}(N)}$$

Now let's slow down execution by a factor of μ>1 (for strong scaling):

$$S_{\mu}(N) = \frac{\mu}{\mu\left(s + (1-s)/N\right) + c(N)} = \frac{1}{s + (1-s)/N + c(N)/\mu}$$

i.e., if there is overhead (c(N)>0), the slow code/machine scales better:

$$S_{\mu}(N) > S_{\mu=1}(N) \quad \text{if} \quad c(N) > 0$$

Slow Computing 101

1. Do not use high compiler optimization levels or the latest compiler versions, because of **numerical stability**

2. Use fancy C++/JAVA/Python/… frameworks – they are much more **maintainable and flexible**

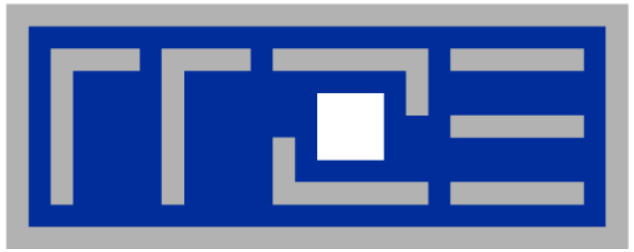
3. Scalability is still bad?

   → Parallelize **short loops** with **OpenMP**

   and earn some extra bonus for a **scalable hybrid code**.

Time to solution?

"If I had a bigger machine, I could get the solution as fast as you want. This is of course due to the superior scalability of my code which is ready to scale on exaflop machines….."



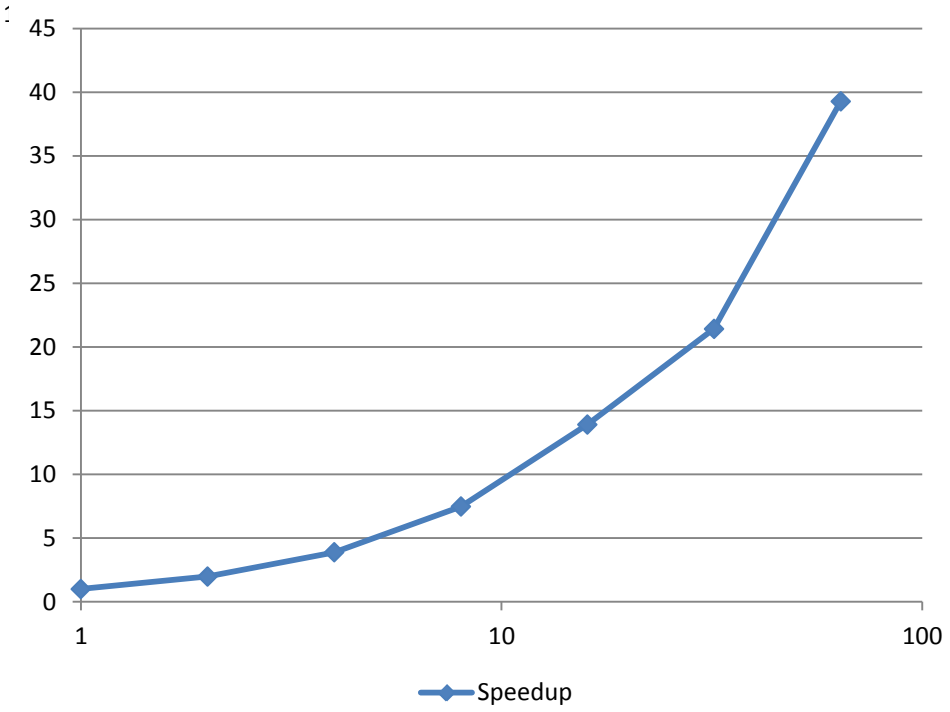
The fine arts of graph design

The Log Scale is your friend!

If scalability doesn't look good enough, use a logarithmic scale to drive your point home.

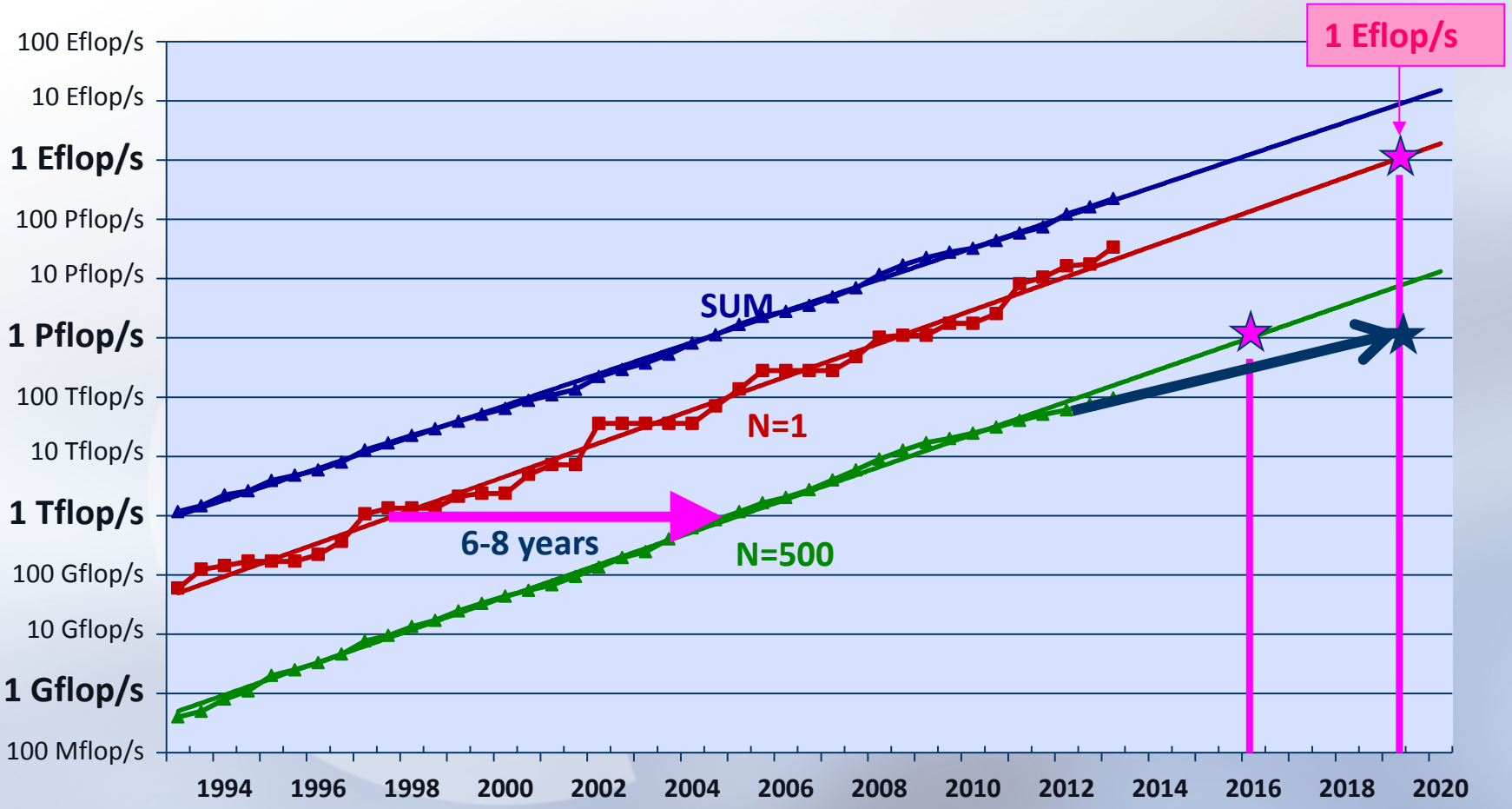
Everything looks OK if you plot it the right way!

1. Linear plot: bad scaling, strange things at N=32
2. Log-log plot: better scaling, but still the N=32 problem
3. Log-linear plot: N=32 problem gone
4. … and remove the ideal scaling line to make it perfect!



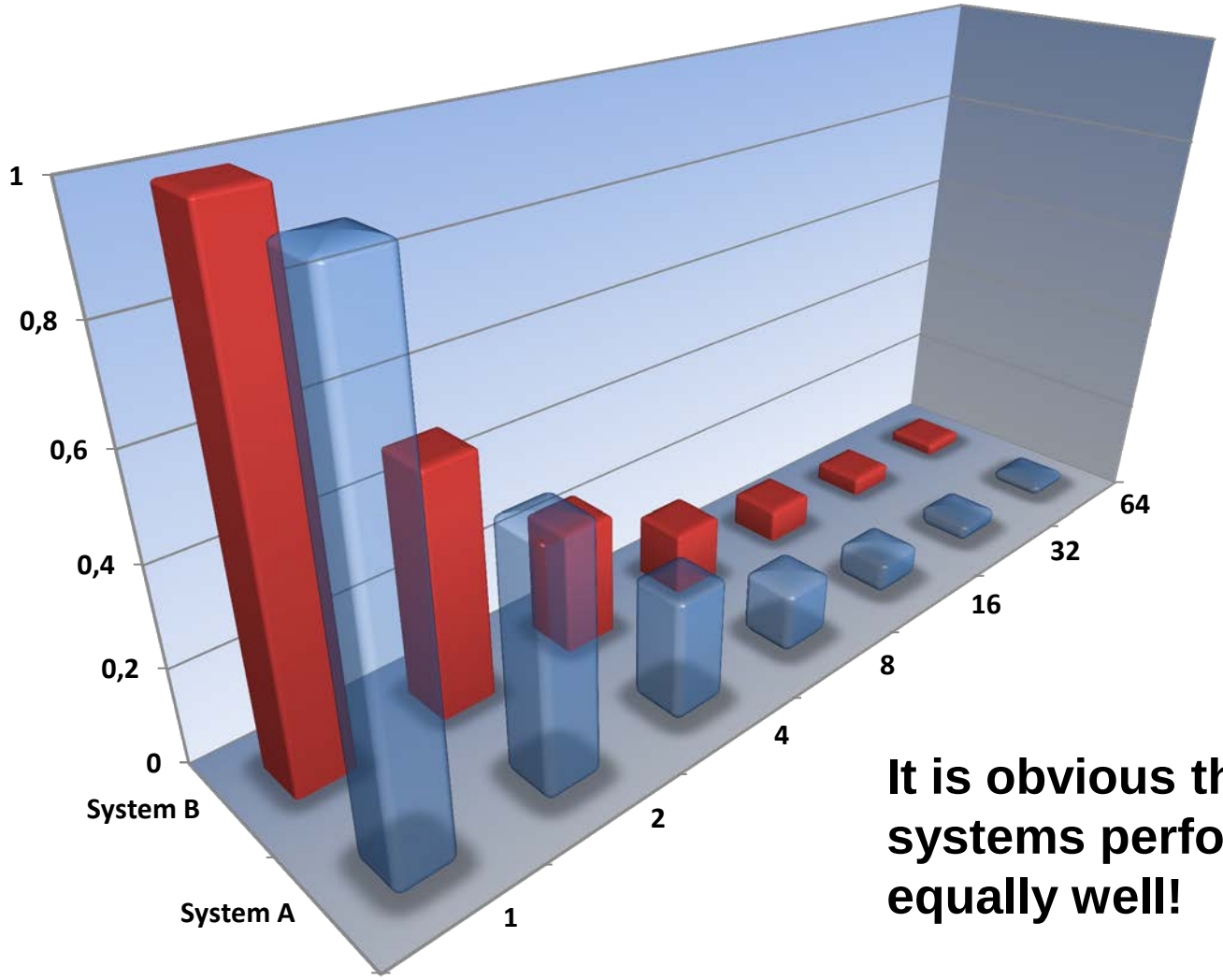
List 1 (Jun 1993) to 41 (Jun 2013)

Performance Projection



By courtesy of Hans Meuer

Use the power of present day visualization tools!

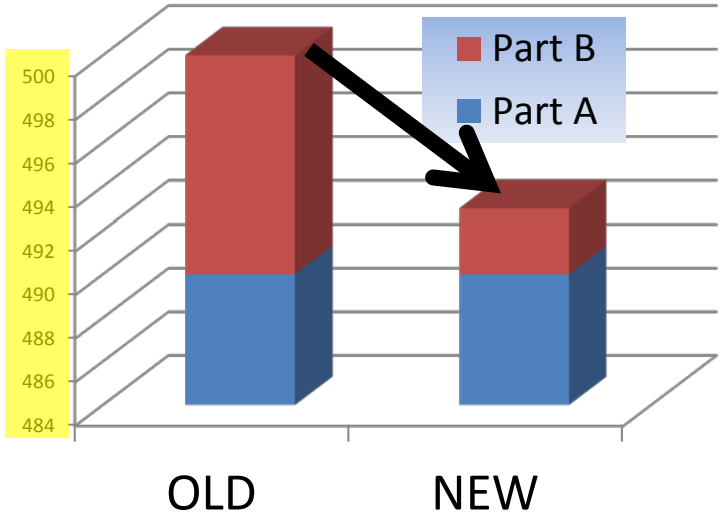


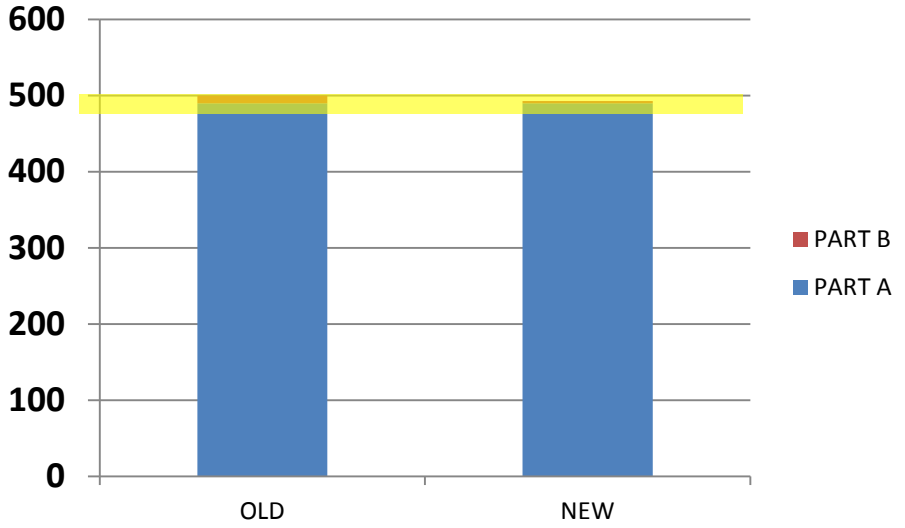
It is obvious that both systems perform equally well!

Keep focus on relevant information

Keep graphs simple and focus to the most important region of data to make your point.

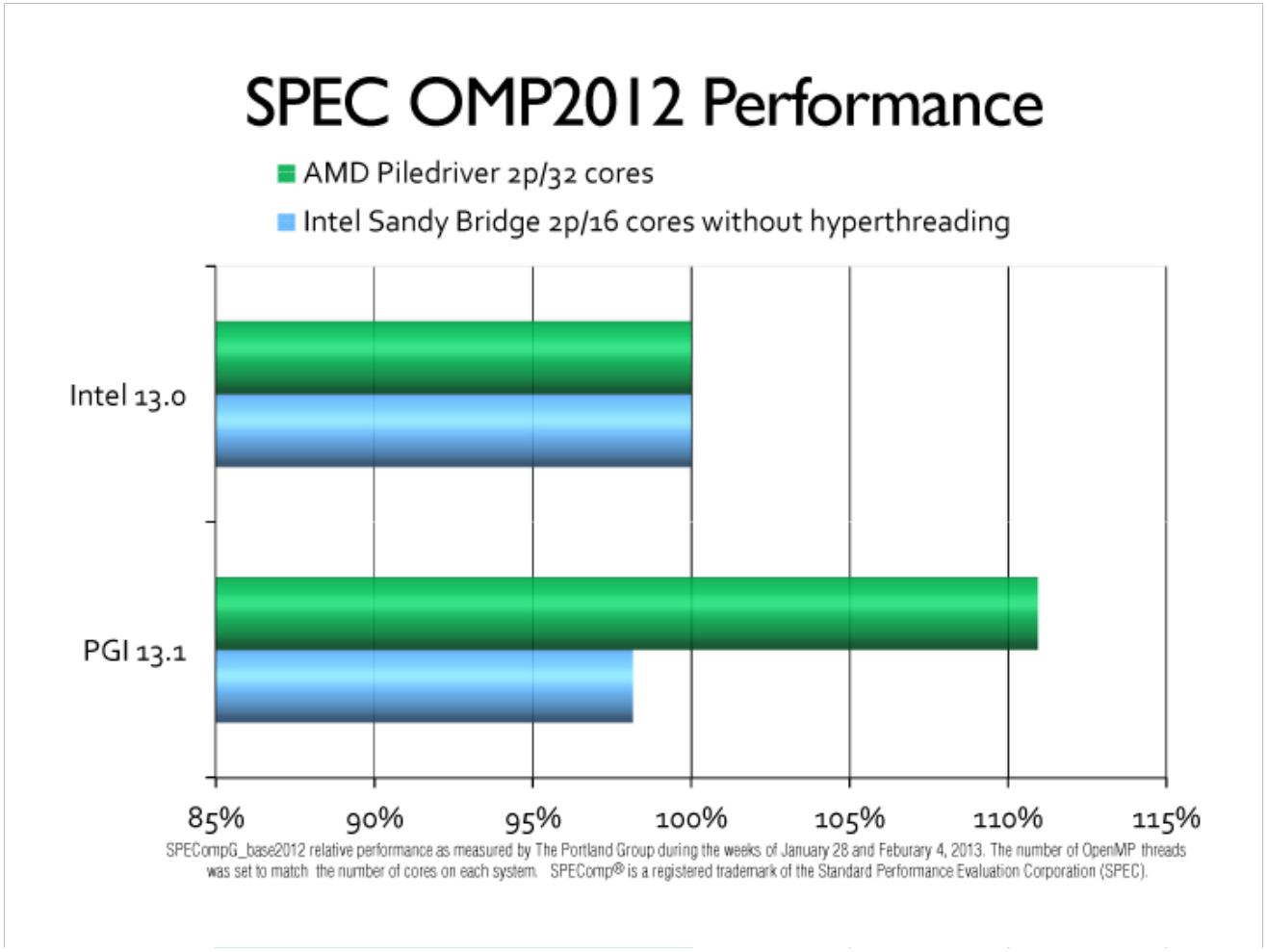
“Fig. 3 demonstrates the benefit of our new scheme for Part B which reduces overall execution time of B by 71%”

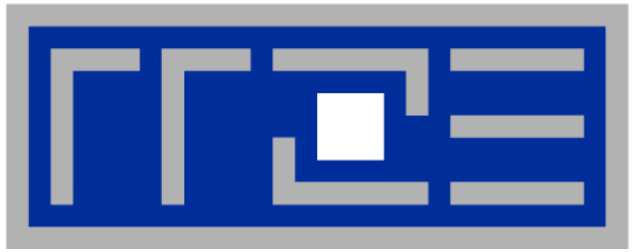




Adding a strong/bold arrow further emphasizes the importance of your achievement and 3D bars really look professional.

http://www.pgroup.com/images/charts/spec_omp2012_chart_big.png





Getting a decent speed-up for new, fancy compute devices aka accelerators

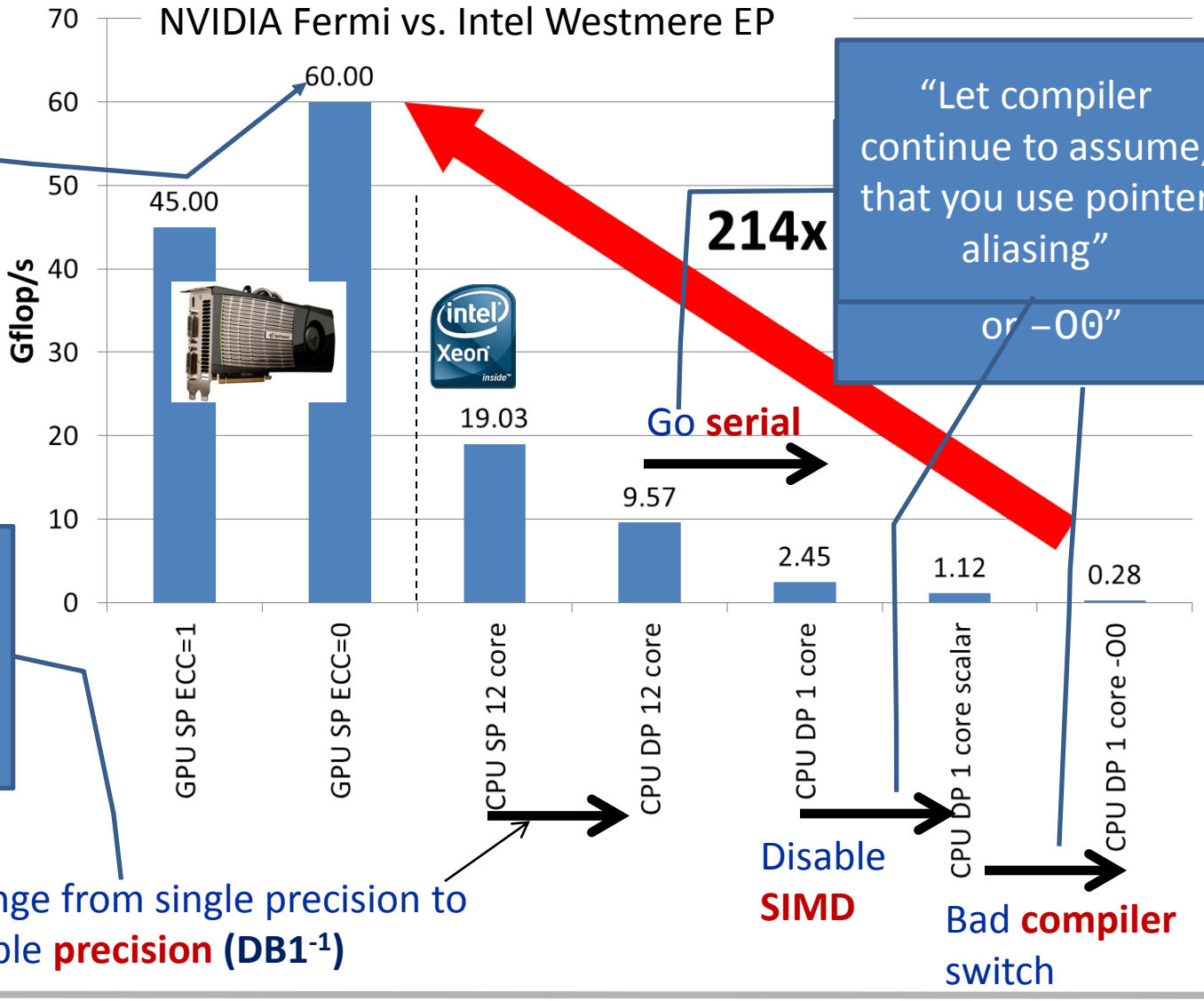
“Compare your results against scalar, **unoptimized code** on Crays.”

How to tell the 200x GPGPU speed-up story

"Numerically sensitive code: Does not require ECC!"

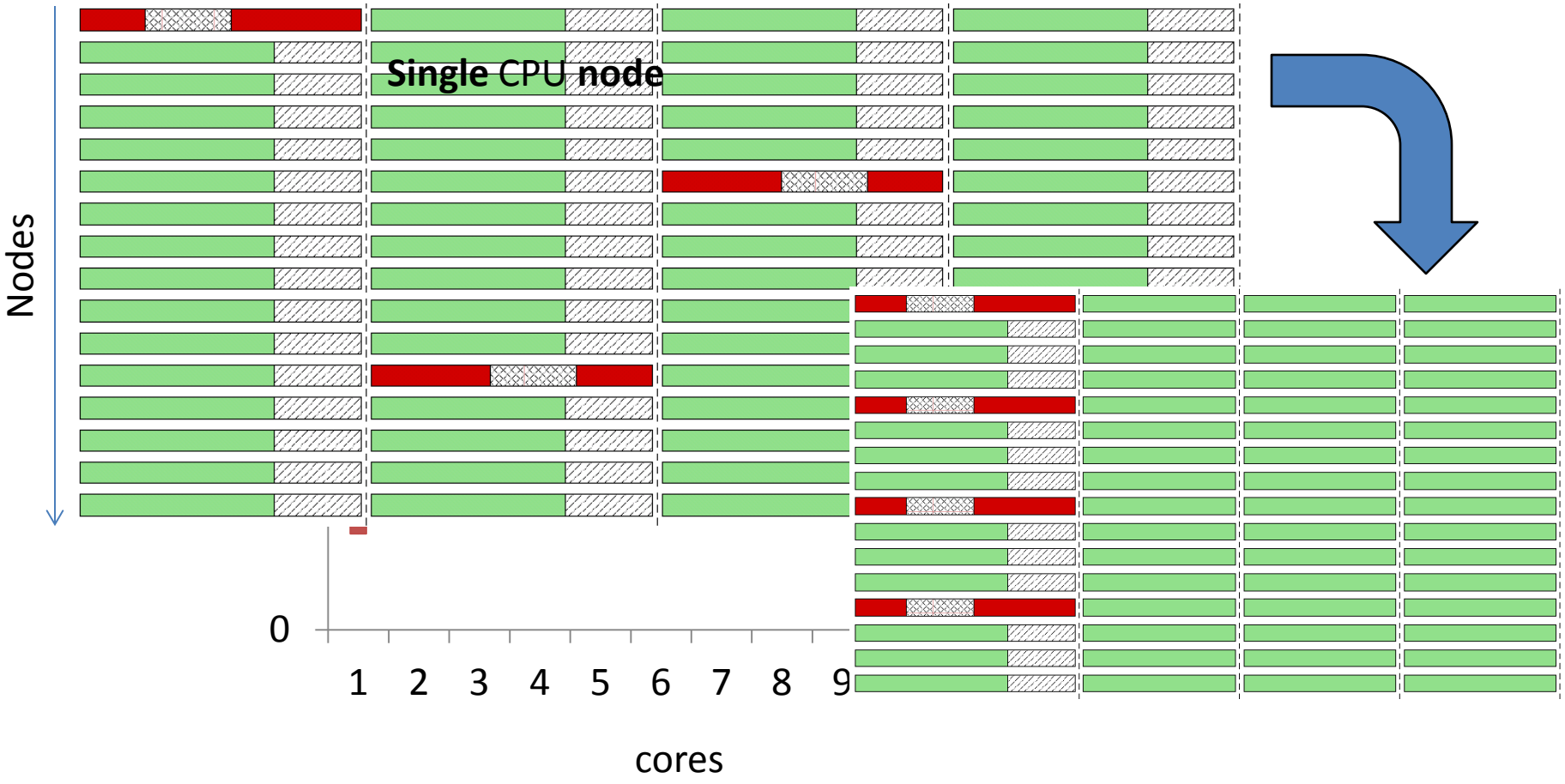
Dense Matrix-Vector-Multiplication (N=4500)

"Our CPU code is based on double precision and hard to change"



If they get you cornered, blame it on OS jitter

Strange scalability? Blame it on OS jitter [1] → Audience nod knowingly.



[1]Fabrizio Petrini, Darren J. Kerbyson, and Scott Pakin. 2003. **The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q**. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing* (SC '03).

…equivalent single core best sellers

L1 cache hit ratio $\frac{LD\ instructions - L1\ misses}{LD\ instructions}$

```
a(1:N)=a(1:N)*s
```

	AVX	Scalar
L1 hit ratio	50.0%	87.5%

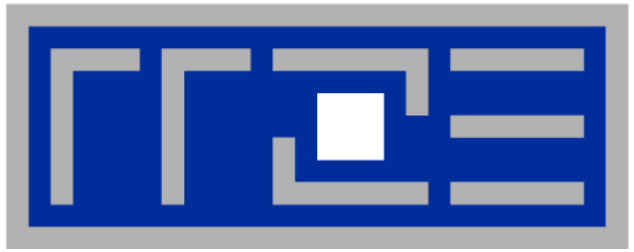
Scalar execution:
Every 8th 64-Bit LOAD
generates an L1 miss (512 Bit cache line)

AVX SIMD execution:
Every 2nd 256-Bit LOAD generates an L1 miss (512 Bit cache line)

CPI (cycles per instruction) rate – The higher the better

→ Scalar execution is your friend again!

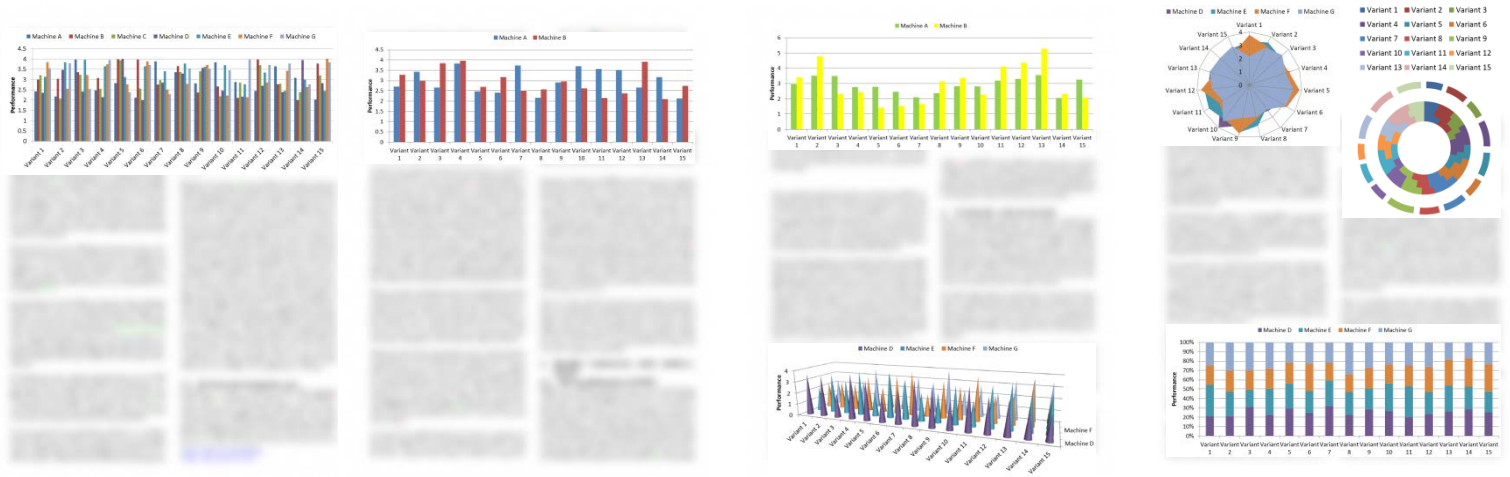
Depending on the audience, **TLB misses** may work just as fine.



Show plenty of real data…

… there are so many things to check/optimize

Show plenty of real data



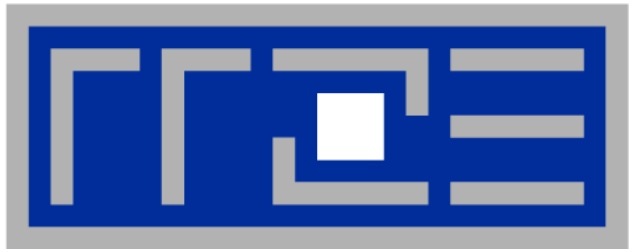
Don’t try to make sense of your data in terms of a performance model!

Show many densely populated colored graphs - You did a lot of work!

If nasty questions pop up:

- Code is so complex that no model can describe it

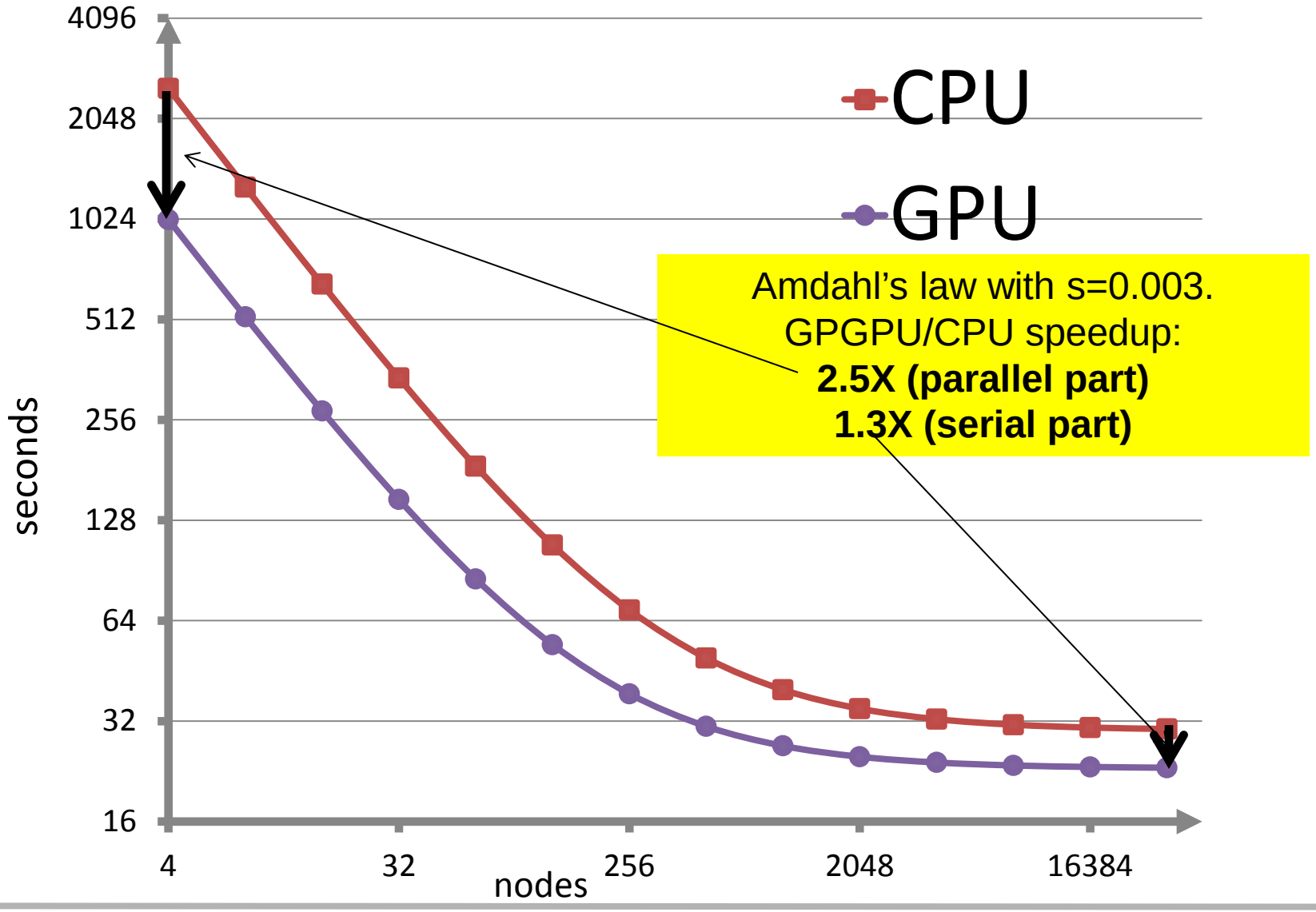
- If you need to **explain some of the measurements** (nobody will ask for all) – **L1 hit ratio, CPI, DTLB**,… will do their job



Accelerated parallel speed-ups!

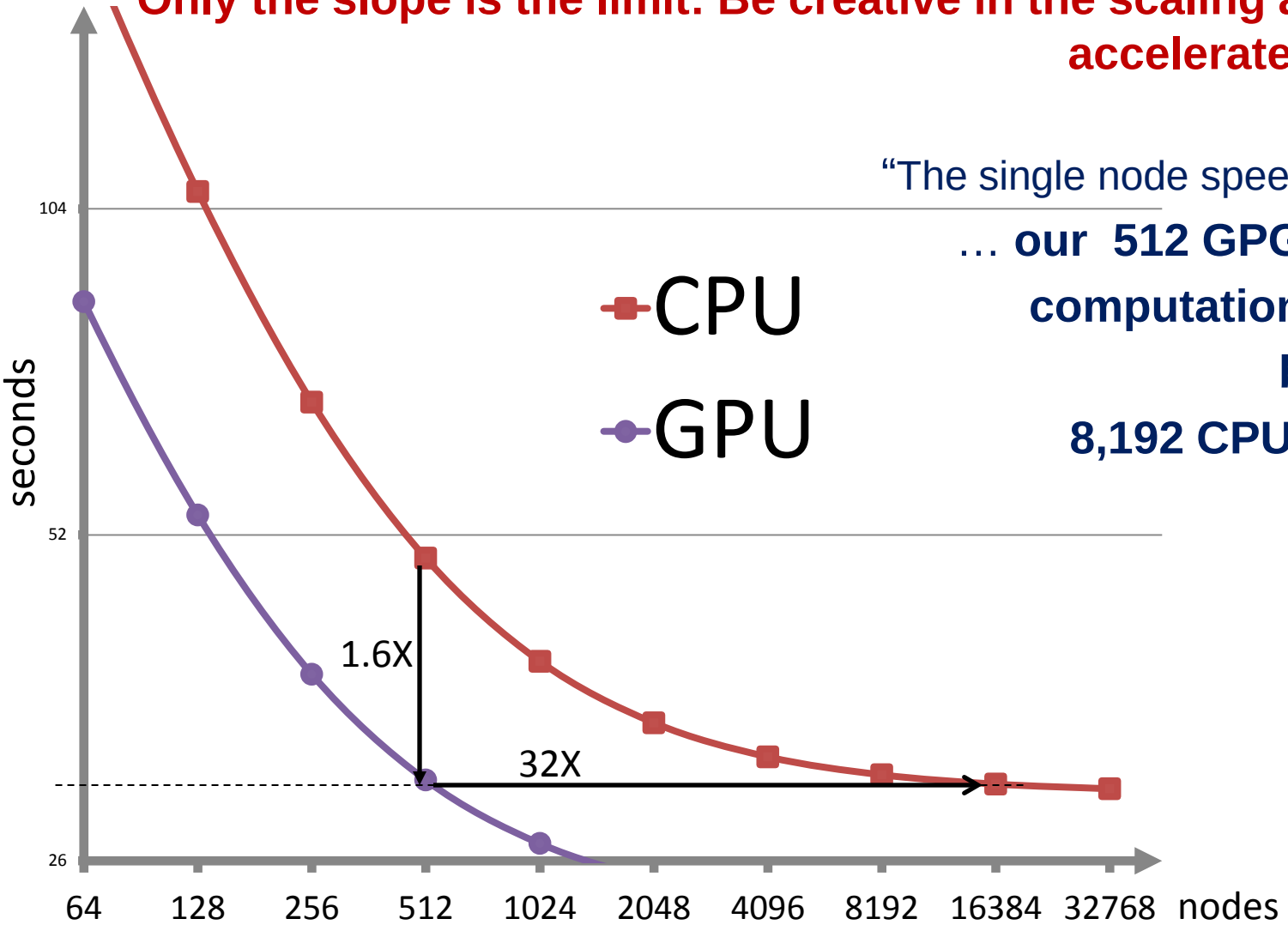
Be creative – there are nowadays so many opportunities

Accelerated speed-ups



Accelerated speed-ups

Only the slope is the limit: Be creative in the scaling analysis of accelerated systems



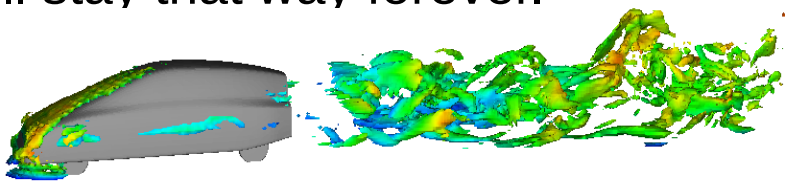
"The single node speed-up is 2.5x,

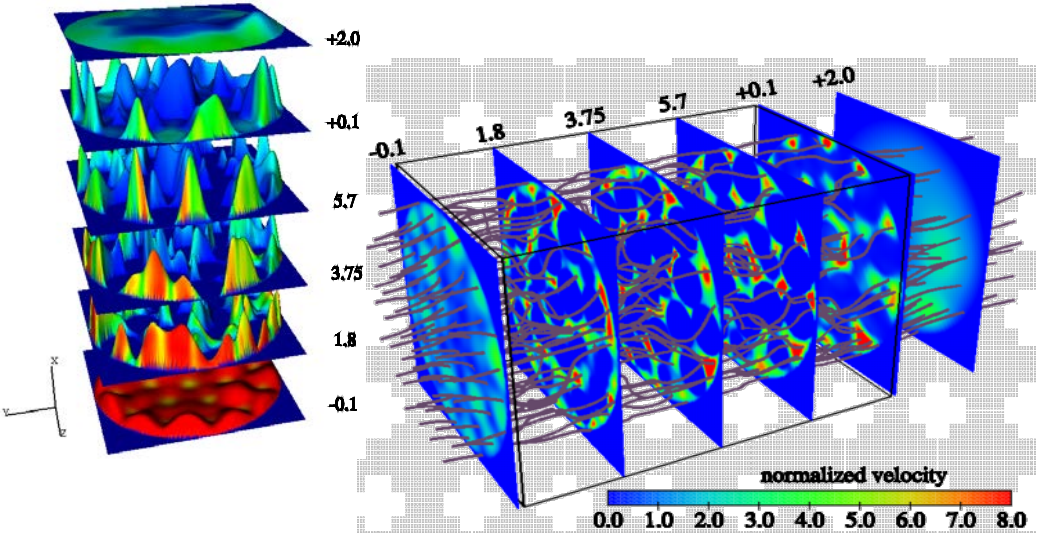
… **our 512 GPGPU nodes computation performs better than 8,192 CPU nodes**…."

If all else fails,…

show pretty pictures and animated videos, and don't talk about performance.

In four decades of supercomputing, this was always the best-selling plan, and it will stay that way forever.





Summary – Recommendations

- **Be careful!**
  Do not use Bailey's 12 ways or our stunts straight away

- **Be creative!**
  There are so many new hardware parameters
  If none of the existing metrics matches your problem – create a new one

→We are looking forward to your new ideas!

http://blogs.fau.de/hager/category/fooling-the-masses/