BLUE WATERS

SUSTAINED PETASCALE COMPUTING

Performance Report Guidelines

Babak Behzad, Alex Brooks, Vu Dang

12/04/2013

NCSA

GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION

CRAY

Motivation

- We need a common way of presenting performance results on Blue Waters!
  - Different applications
    - Different needs
    - Different metrics
    - …
  - Different architecture
    - GPU vs CPUs
- How to present performance results correctly?
- How to report scalability correctly?

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

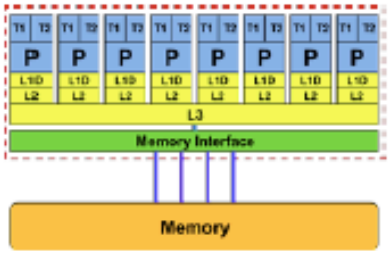
David H. Bailey, Supercomputing Review, August **1991** , p. 54-55

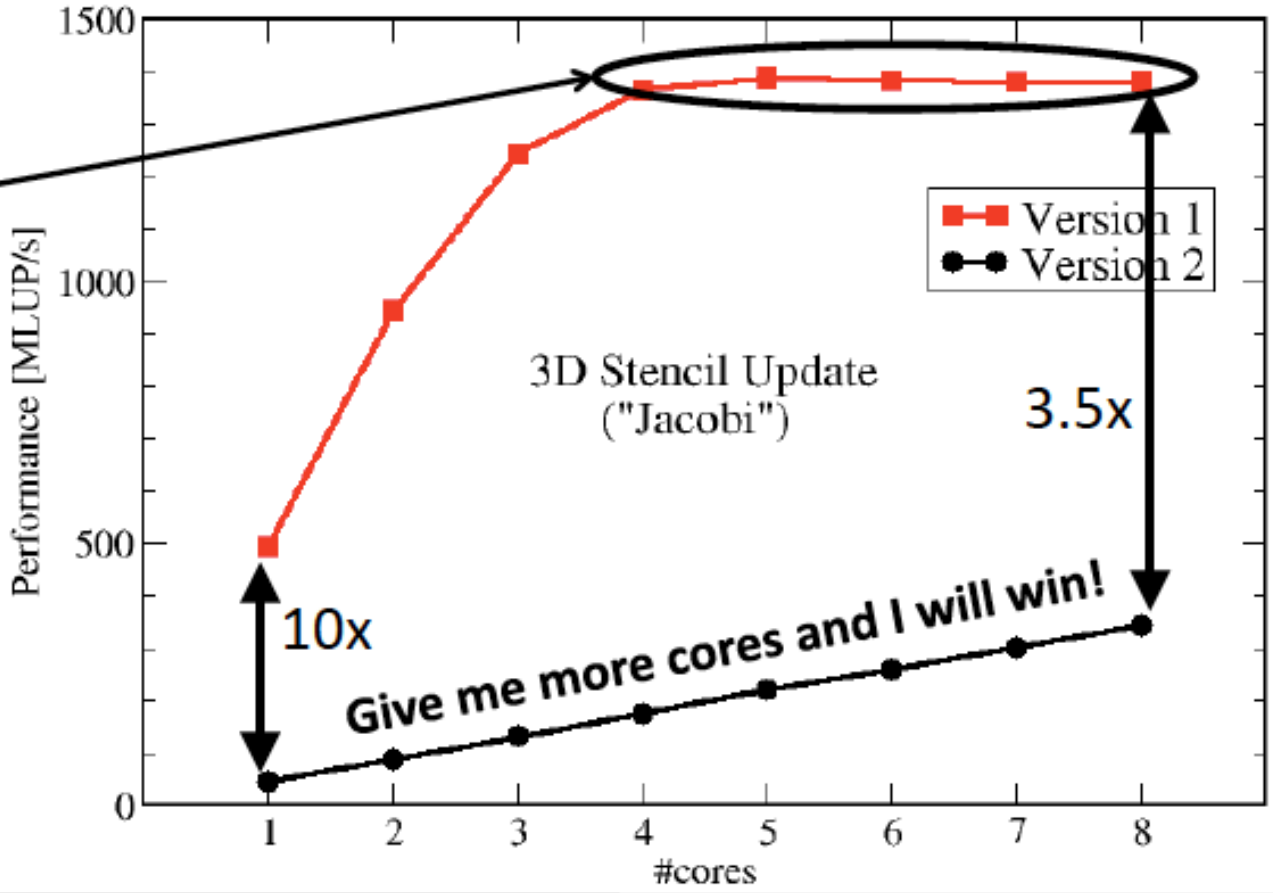
1. Quote only 32-bit performance results, not 64-bit results.
2. Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application.
3. Quietly employ assembly code and other low-level language constructs.
4. Scale up the problem size with the 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
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.

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 ?!





*Slide from Gerhard Wellein, Georg Hager : http://blogs.fau.de/hager/files/2013/06/2013-06-20-ISC13-FTM.pdf

Speedup and Scalability

- Speedup:

$$S(P) = \frac{time\ with\ 1\ process}{time\ with\ P\ processes}$$

- Scalability: relative effectiveness with which parallel algorithm can utilize additional processors
- Good scalability: Linear scalability

$$S(P) = P$$

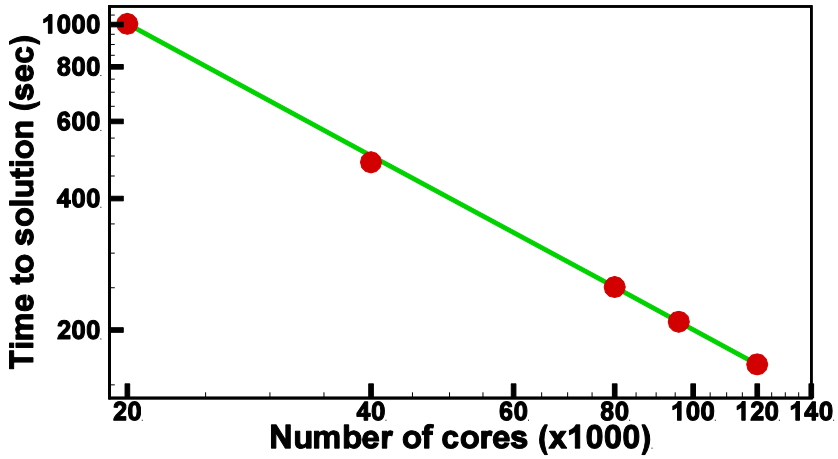
Scalability?

- Why use more processors?
  - solve given problem in less time
  - solve larger problem in same time
  - obtain sufficient memory to solve given (or larger) problem
  - solve ever larger problems regardless of execution time
- Keep some quantity constant as number of processors increases
  - serial work → Strong scaling
  - serial work per processor → Weak scaling

Strong Scaling

- Strong Scaling:

Number of cores	Time (sec)	Speed up	Ideal
20,000	1003		
40,000	484	2.07	2
80,000	251	1.93	2
96,000	209	1.20	1.2
120,000	167	1.25	1.25

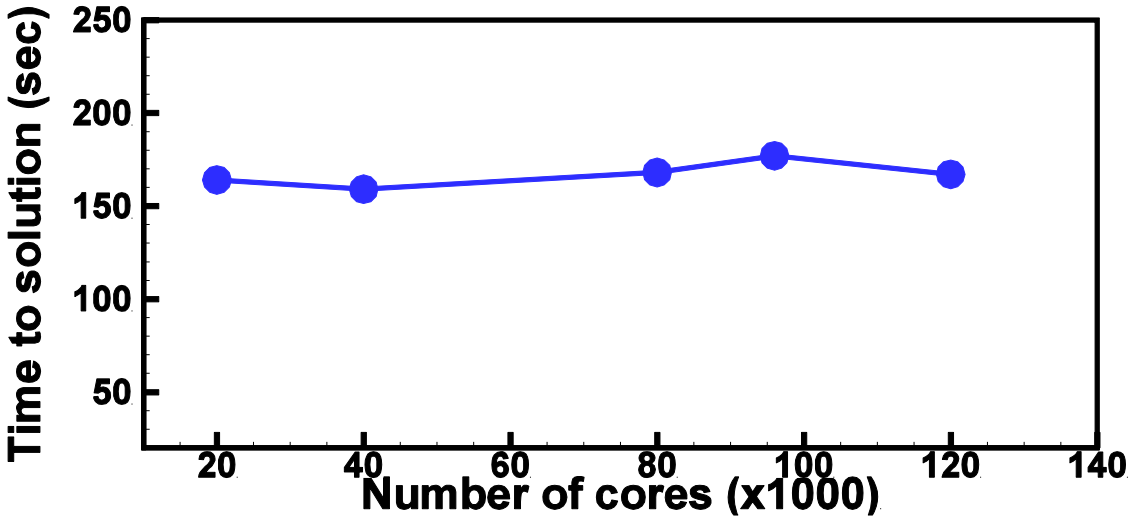


Strong scaling results of the kinetic code. The green line shows ideal performance. The red circles are observed time.

Weak Scaling

- Weak Scaling:

Number of cores	Time (sec)
20,000	164
40,000	159
80,000	168
96,000	177
120,000	167



Weak scaling results of the kinetic code

TODO 1: Report Both Scaling

- Report both Strong scaling and Weak scaling of your code.

- Define what the scaling unit means.
  - Tasks,
  - cores,
  - sockets,
  - nodes, …

TODO 2: Define your Sequential Code

- Should compare all parallel speedups to sequential code
  - Define exactly what sequential means
  - If it is a parallel implementation that only uses 1 task, please be clear.

TODO 3: Speedup of which part?

- Define how do you measure speedup
  - Is it for the entire code?
  - If a subset, explain the contribution of that subset to the overall run time

TODO 4: Be clear in changes

- When comparing from an older technology to a new technology, be very clear on the contributions for the architectural and technology changes.
  - Well describe all systems with references – not just nodes but other things
- This also holds if you migrate from one platform to other platform.

TODO 5: Selection of results

- Explain how the result was selected.
  - Best of all runs?
  - Average of all runs?
  - Worst case?

- In case of comparing with some other code, make sure comparisons are fair.
  - Don’t compare your best case with others’ worst case

TODO 6: Define App-specific metrics

- Define application specific metrics
  - Simulation time unit/real time unit,
  - days/hour,
  - ns/minute
- Explain them and say why do you choose that metric
- Explain what range of that metric is better

TODO 7: Define time unit clearly

- Define time unit clearly – how measured – wall clock time is the preferred

TODO 8: Note Use of Fancy Features

- If you are using fancy features (e.g. topology aware mapping) make sure you note what they are and how different it is

- How the options on the software are set?
  - Whether using options that are not available.

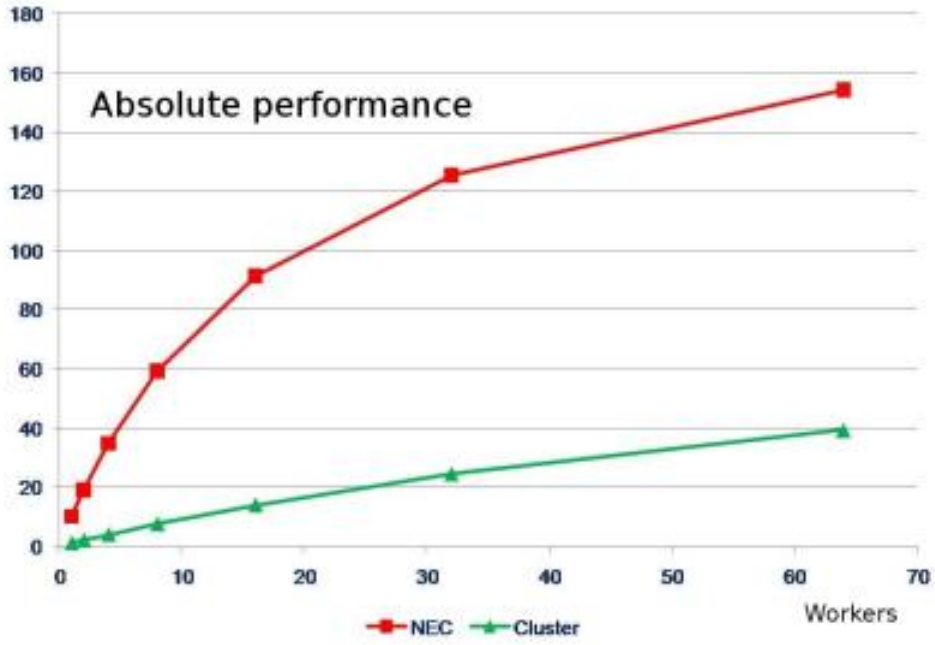
TODO 9: Correct Comparison

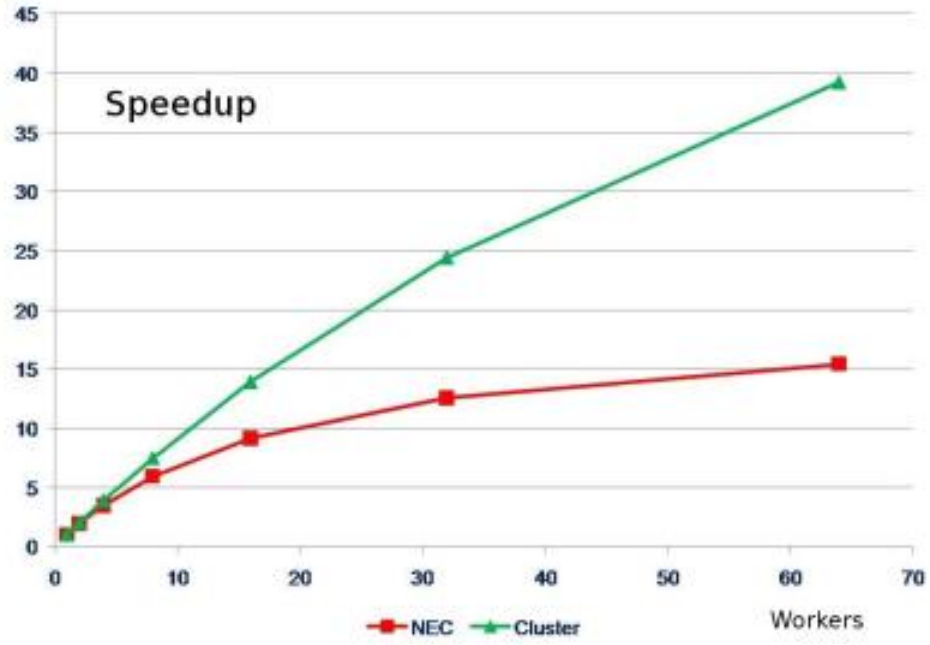
- You can compare very basic ported case and very aggressive optimization with different compilers
- But do not compare between.

TODO 10: Error Bars

- On a big platform such as Blue Waters, there are sources of variability in performance!
- It would be great to run your code several times and show us an error bar
  - It would be nice to explain what are these sources of variability in your code.
  - Don’t blame everything on OS jitter or I/O subsystem please!

Never TODO 1: Just Reporting Speedup





- If one wants to sell “Cluster”, he/she can easily just show the plot on right!

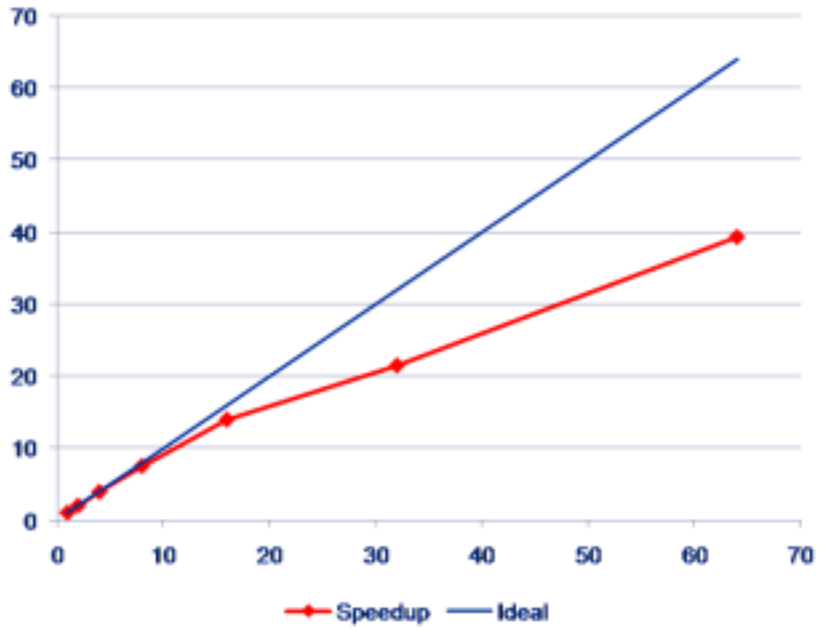
***Slide from Gerhard Wellein, Georg Hager :** http://blogs.fau.de/hager/files/2013/06/2013-06-20-ISC13-FTM.pdf

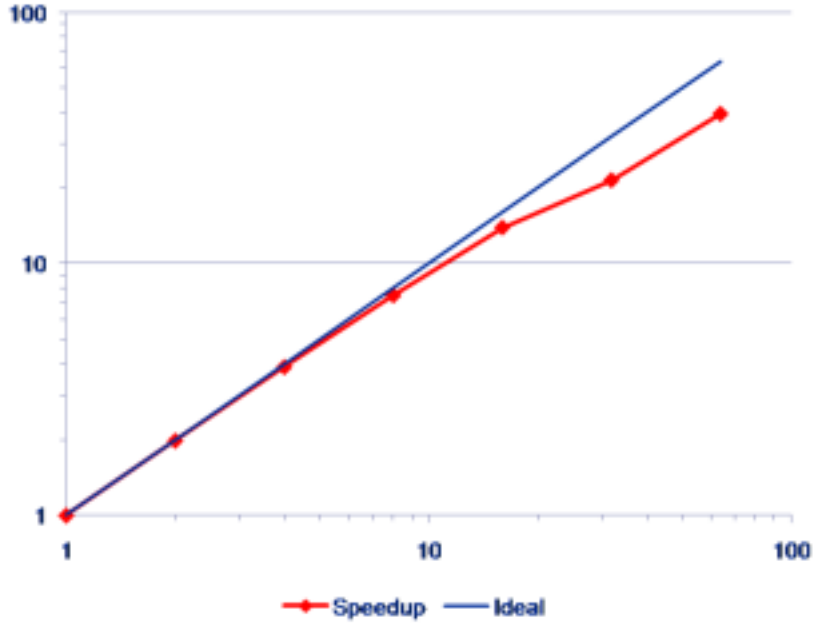
Never TODO 2: Mix Strong scaling with Weak scaling

- Bailey's #4: "Scale up the problem size with the number of processors, but omit any mention of this fact."

*Slide from Gerhard Wellein, Georg Hager : http://blogs.fau.de/hager/files/2013/06/2013-06-20-ISC13-FTM.pdf

Never TODO 3: Use Log-scale unless necessary!





- Log-scale shows a linear speedup on the plot on right!

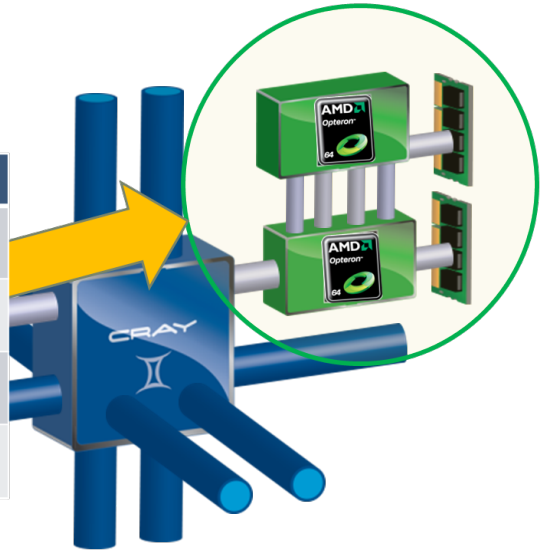
***Slide from Gerhard Wellein, Georg Hager :** http://blogs.fau.de/hager/files/2013/06/2013-06-20-ISC13-FTM.pdf

Never TODO 4: Ignore affinity and other architectural features

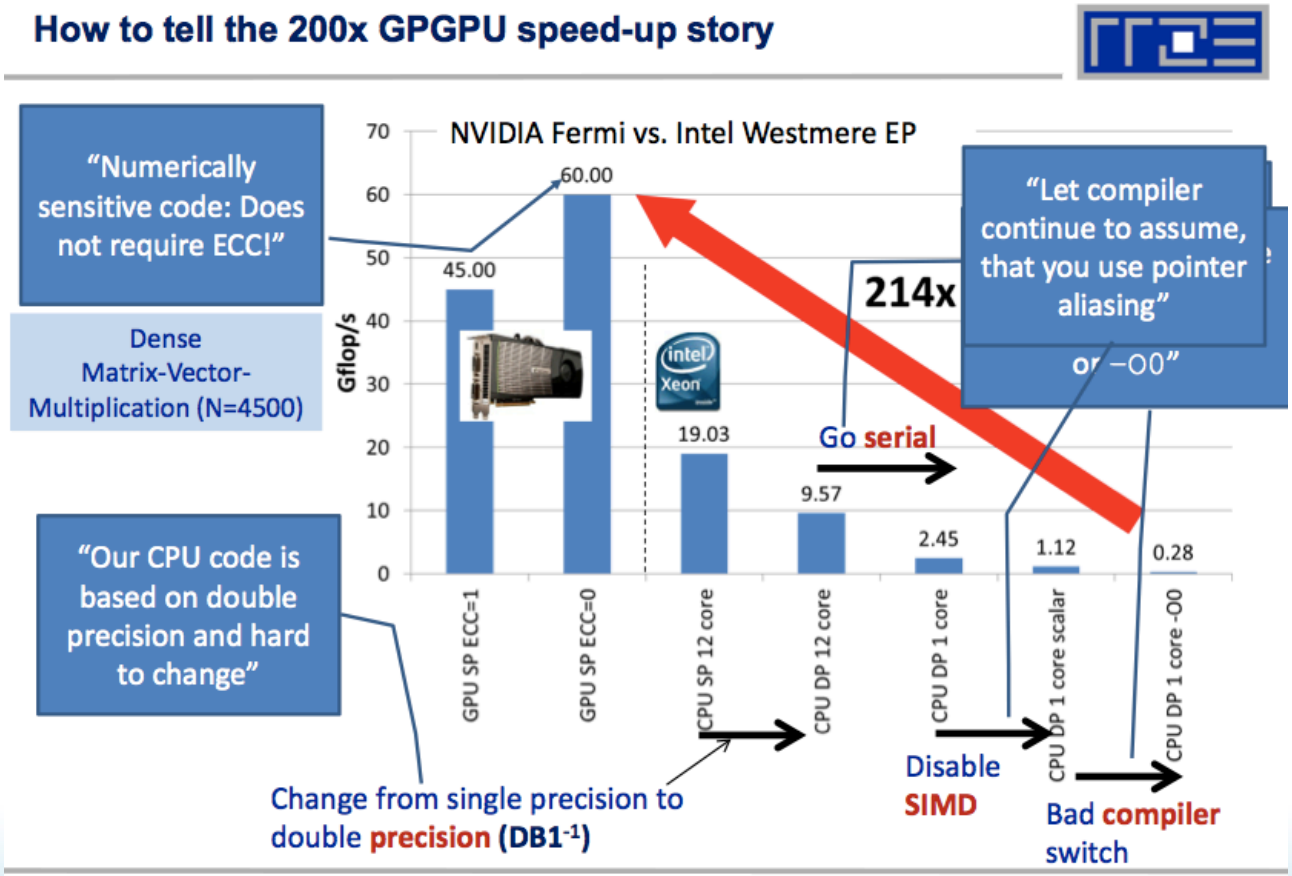
- Multicore
- Cache groups
  - L2/L3 caches
- NUMA
- SMT
- Network hierarchies
- Shared FP units

XE6 Compute Node
- Dual-socket AMD-Opteron
- 4x channel 1600 DDR3 memory
- High speed HT3 network link
- Upgradeable
- Blend with XK6 GPU systems

Node Characteristics	
Number of Cores	16
Peak Performance	313.6 Gflops/sec
Memory Sizes Available	64 GB per node
Memory Bandwidth (Peak)	102.4 GB/sec



Never TODO 5: Unfair GPU vs. CPU comparison



*Slide from Gerhard Wellein, Georg Hager : http://blogs.fau.de/hager/files/2013/06/2013-06-20-ISC13-FTM.pdf