P2P MOLECULAR DYNAMICS AND YOU

Scott Le Grand

Amazon Web Services

Outline

- Some Initial GPU Controversy
- GPU Molecular Dynamics
- Generalized Born Performance
- Particle Mesh Ewald Performance
- Hotrodding AMBER for great science™
- Conclusions

Brawny versus Wimpy

Brawny cores still beat wimpy cores, most of the time

Urs Hölzle
Google

"Slower but energy efficient "wimpy" cores only win for general workloads if their single-core speed is reasonably close to that of mid-range "brawny" cores."

STRAW MAN, MUCH?

If you only had a brain...

NVIDIA fell for it…

- GeForce GTX Titan Black: 2,880 CORES!
- GeForce GTX Titan: 2,688 CORES!
- Tesla K40: 2,880 CORES!
- Tesla K10: 3,072 CORES!

One SIMD Lane == One Core

By this definition, these cores are really wimpy…

Definition of "Core"

Core: a set of processing elements that share an L1 cache (or equivalent) and register file

Processor: One or more cores on a single die

Data Locality Matters!

CPUs are indeed brawny cores

Fast CPU: Intel Xeon E5-2670 SandyBridge-EP 2.6 GHZ (3.3 GHz Turbo Boost) 20 MB L3 Cache LGA 2011 115W 8-Core Server Processor ($1,499 on NewEgg)

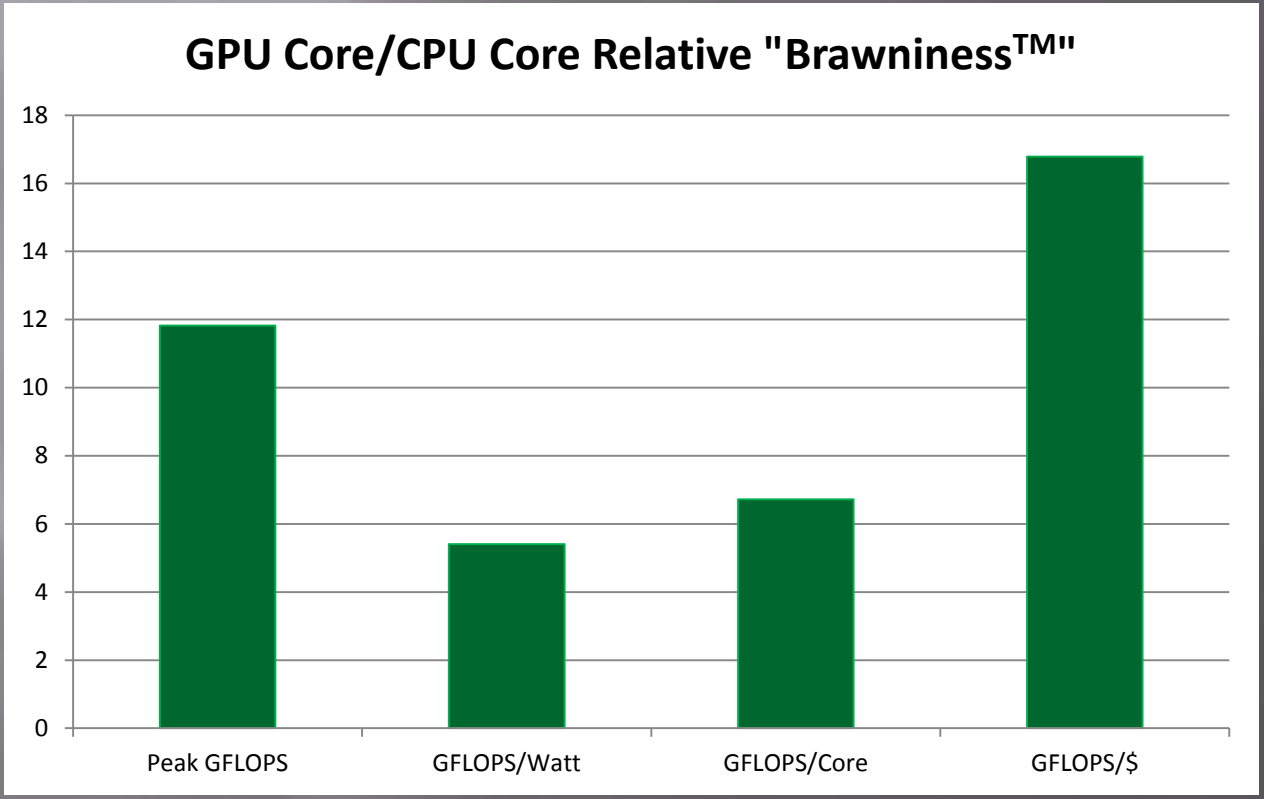
Peak GFLOPS:	~422
GFLOPS/W:	~3.7
GFLOPS/Core:	~53
GFLOPS/$:	~0.28

But GPUs are brawnier™ cores

Fast GPU: EVGA 06G-P4-2793-KR NVIDIA Ge Force GTX Titan, 14-core, 928 GHz ($1069.99 on NewEgg)

Peak GFLOPS:	~4,989
GFLOPS/W:	~20
GFLOPS/Core:	~356
GFLOPS/$:	~4.7

GPU Cores are roughly 5-15x brawnier…



So GPUs themselves should be 10-20x brawnier than CPUs or you're not doing it right…

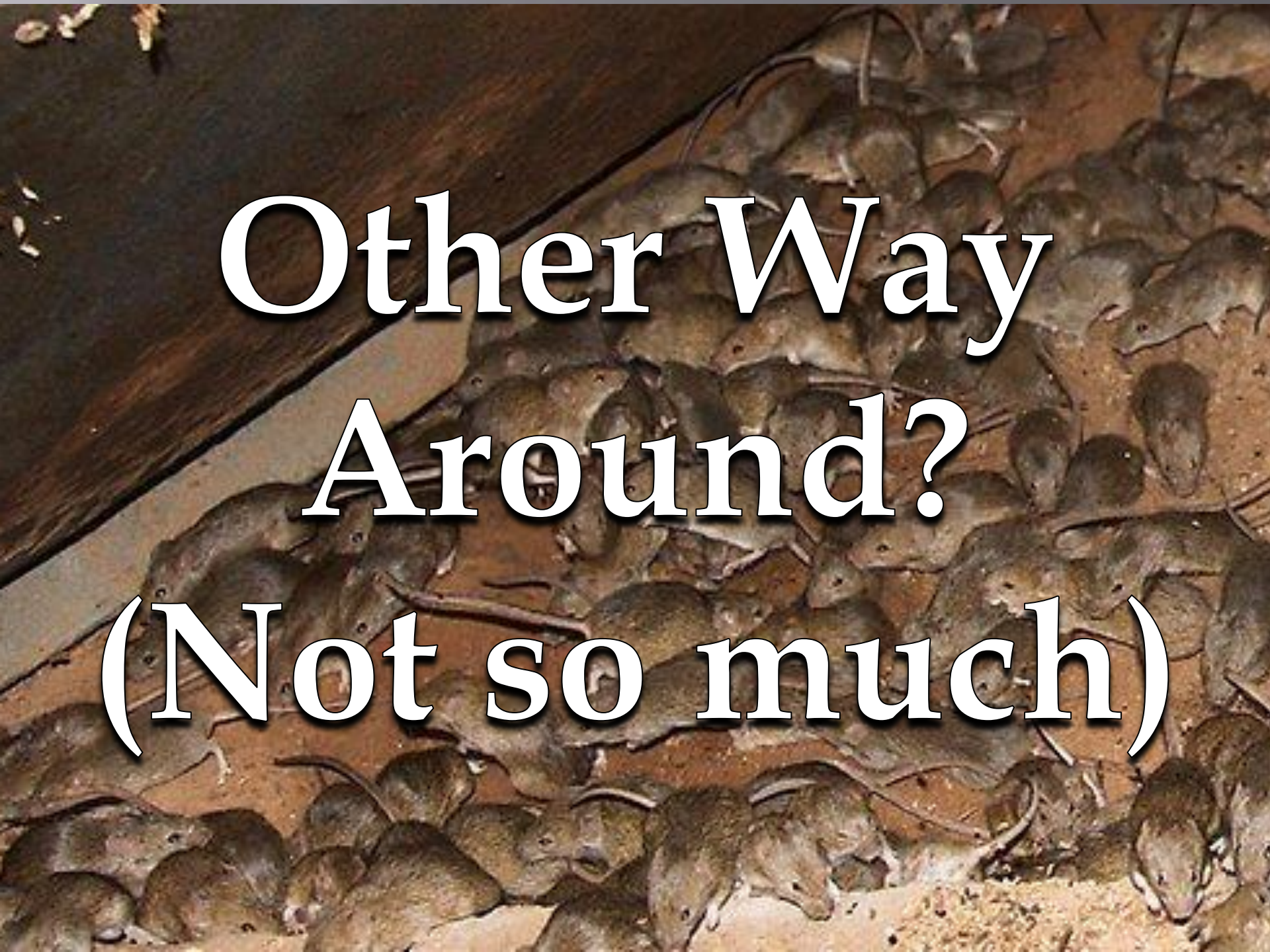
Or the algorithm you're running is inherently serial*…

*But then why exactly are you running it on 1,000+ machines at once**.

**Because you're I/O bound? Well then you're just wasting power using "Brawny" cores


Brawny versus
Wimpy is so
2011*
NEW
SHIMMER
NON-DAIRY
FLOOR WAX
HARD LUSTRE
CREAMY
*We need both at once (2014)


One Big Processor
that can virtualize
into many smaller
Processors
VOLTRON
DEFENDER OF THE UNIVERSE


Other Way
Around?
(Not so much)

Still no love for wimpy cores?

Core: a set of processing elements that share an L1 cache (or equivalent) and register file

Unless they can share data, wimpy cores get hammered by Amdahl's Law

Data Locality Matters! Avoid wimpy cores!

My own rules of thumb

- $O(N^2)$ Embarrassingly Parallel
- $O(N \log N)$ Annoyingly Parallel
- $O(N)$ Likely I/O-Bound

Molecular Dynamics on GPUs
(or how to keep 21,504++ threads occupied)

- On a CPU, the dominant performance spike is:

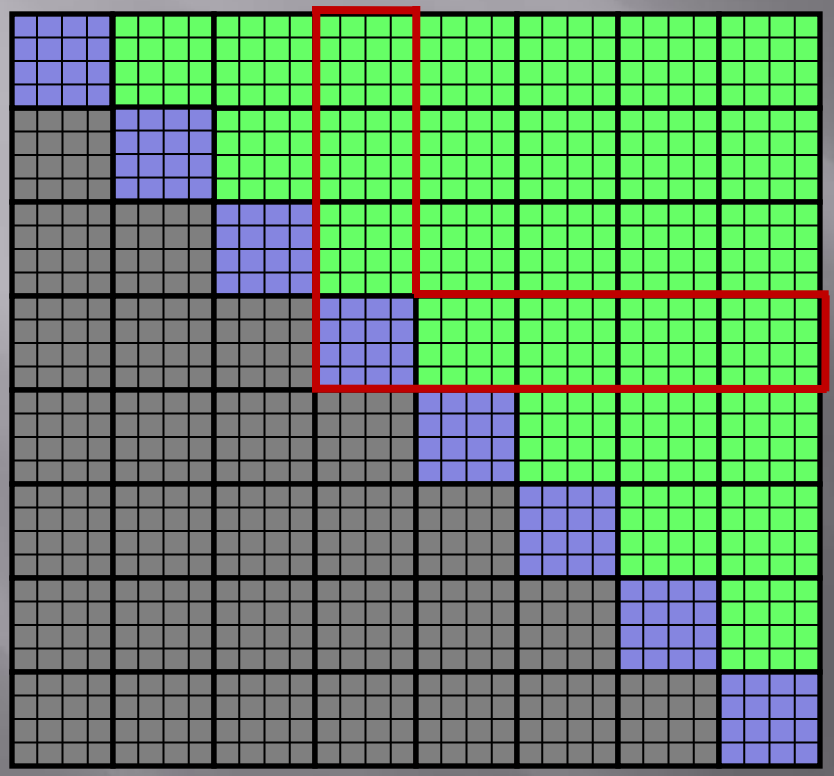
```
for (i =0; i < N; i++)
    for (j = i + 1; j < N; j++)
        Calculate fij, fji;
```

$O(N^2)$ Calculation

If we naively ported this to a GPU, it would die the death of a thousand race conditions and memory overwrites

Solution: Map the problem into many subtasks and reduce the results

MapReduced Molecular Dynamics



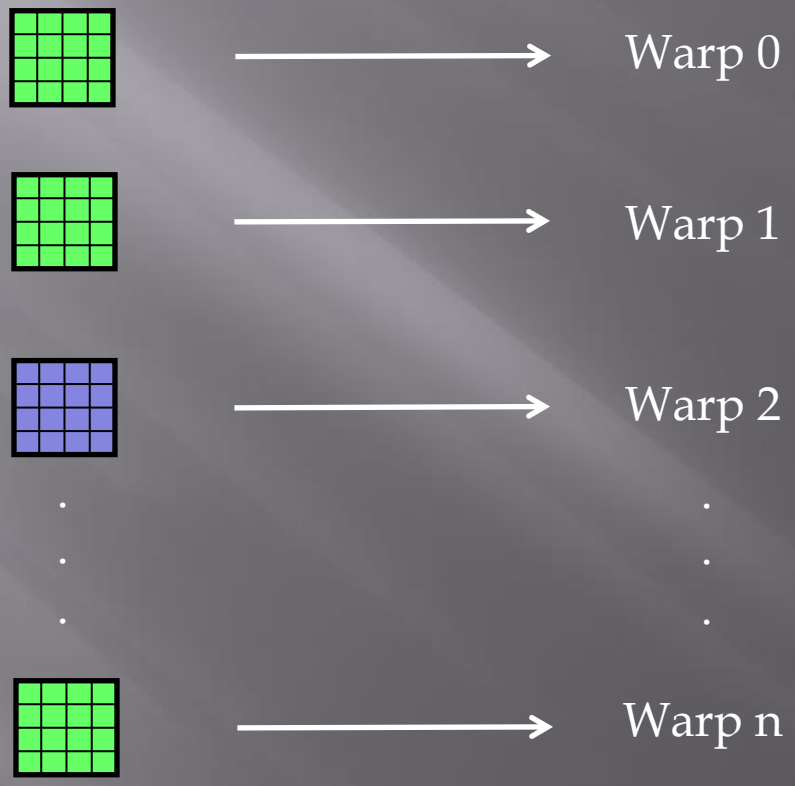
Subdivide force matrix into 3 classes of independent tiles

Off-diagonal

On-diagonal

Redundant

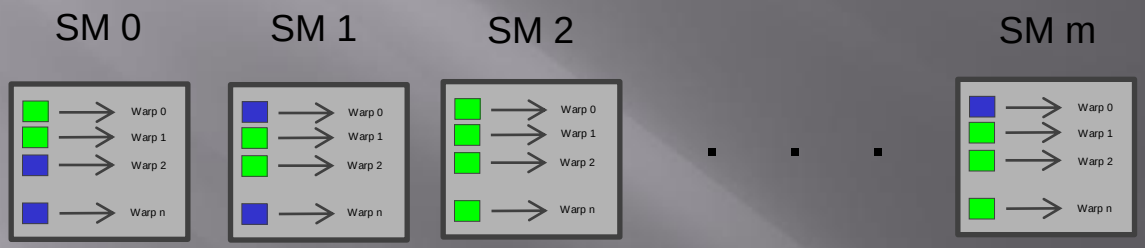
Map each nonredundant tile to a warp



Wait, what's a warp?

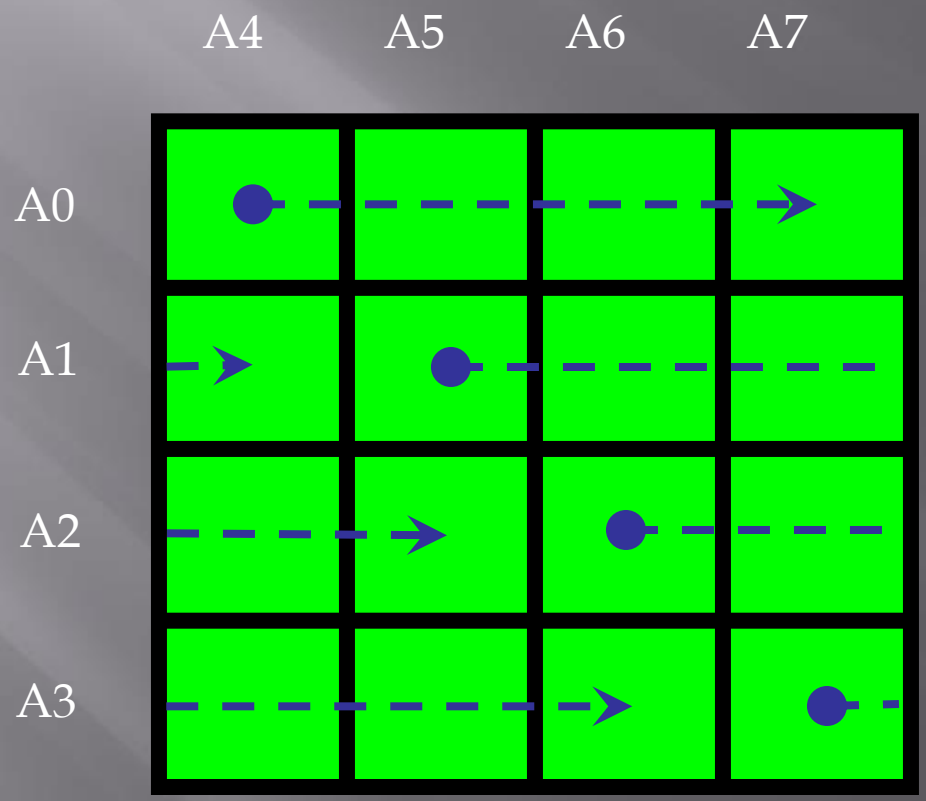
- The smallest unit of execution in a GPU
- Up through today, it's groups of 32 consecutive threads in a thread block that execute in lockstep
- GPU cores each run 8-64 warps at once
- May change in the future
- "lock-free computing"

Each iteration produces force tiles…

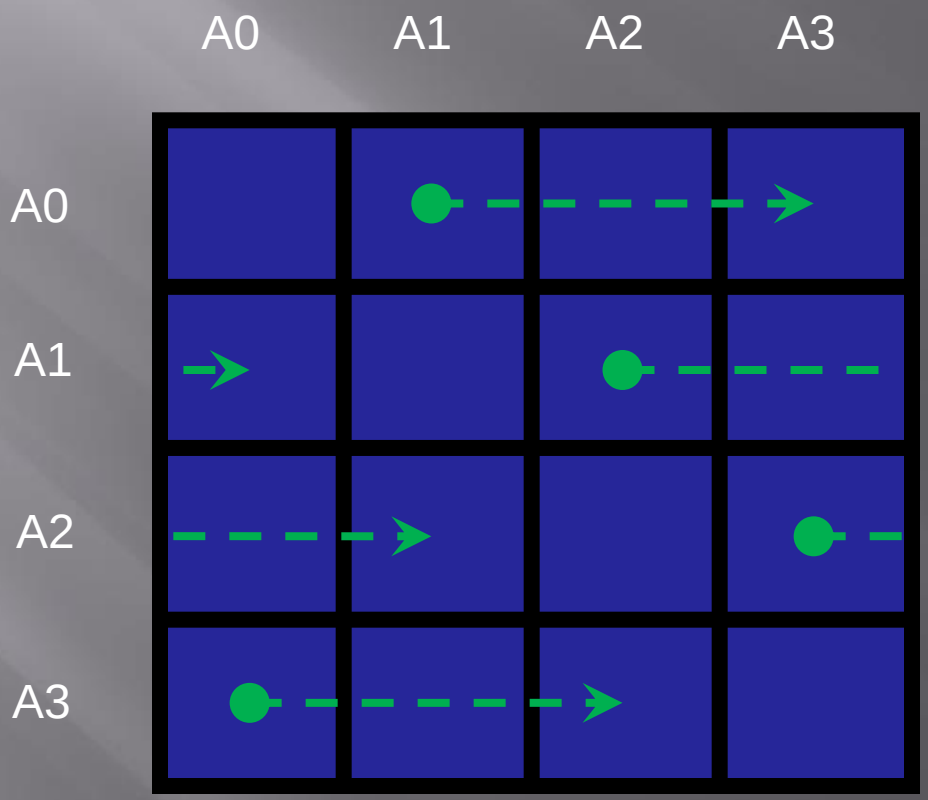


Each warp in the GPU cores consumes them…

Off-Diagonal Tile Calculation in Detail



On-Diagonal Tile Calculation in Detail



21,504 threads!!!!!

- Implies you really need 1,100 or so atoms before this becomes worthwhile
- And it's only going to get worse
- 1,280/32 == 40
- ((40 * 41) / 2) == 26,240
- It gets better, a lot better, from there…

Dynamic Range and GPUS

32-bit floating point has approximately 7 significant figures

```
  1.456702
+0.3046714
--------------
  1.761373
 -1.456702
--------------
  0.3046710
```

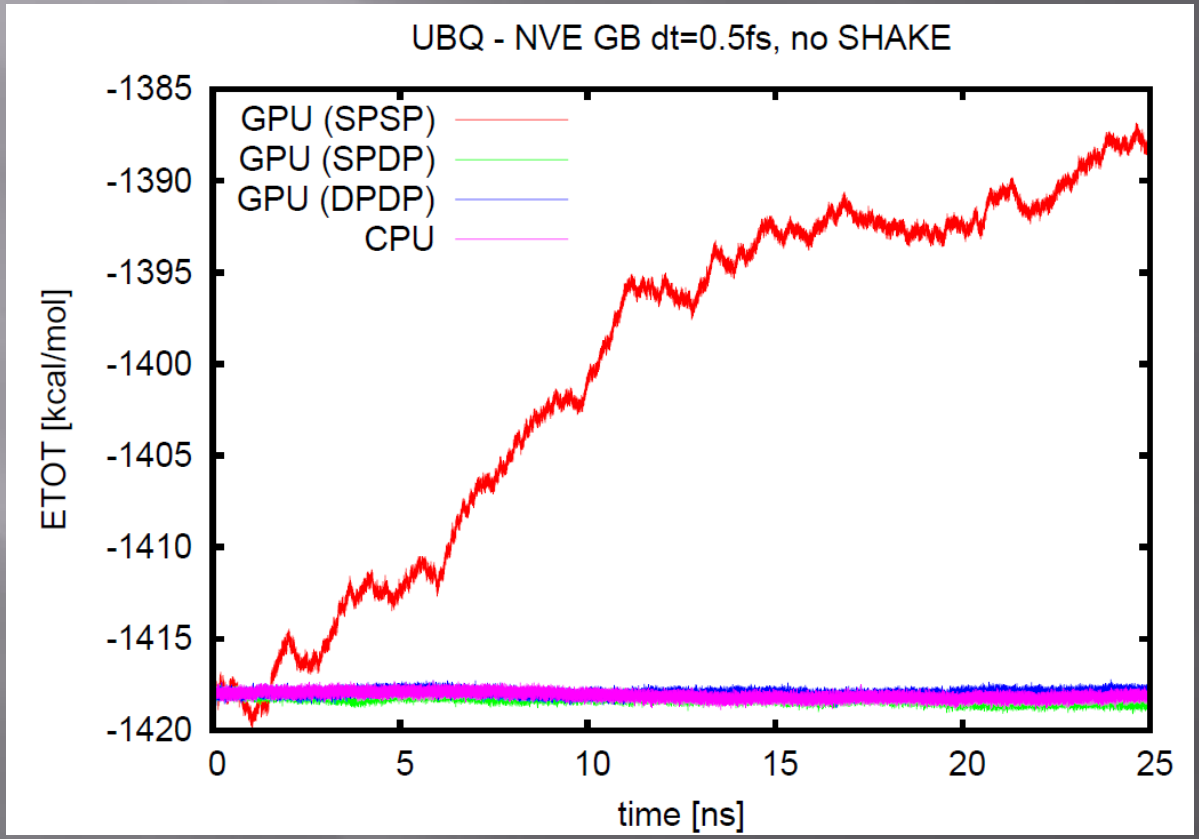
Lost a sig fig

```
  1456702.0000000
+          0.3046714
------------------------
  1456702.0000000
 -1456702.0000000
------------------------
           0.0000000
```

Lost everything.

When it happens: PBC, SHAKE, and Force Accumulation.

Dynamic Range Matters… A Lot…



Reproducible Results Matter… A Lot…

Can you spot the defective GPU?

GPU #1

ETot = -288,718.2326
ETot = -288,718,2325

GPU #2

ETot = -288,718.2326
Etot = -288,718,2326

Let's make it easier…

GPU #1	GPU #2
ETot = -288,718.2326	ETot = -288,718.2326
ETot = -288,718,2325	Etot = -288,718,2326

Non-Deterministic Single-Precision

GPU #1	GPU #2
ETot = -288,456.6774	ETot = -288,458.5931
ETot = -288,453.8133	Etot = -288,454.1539

$!%&? if I know…

But what about ECC?

- Sadly, you're much more likely to encounter a defective GPU than you are to encounter an ECC error
- I've only ever seen one in 8 years myself.
- I've only ever seen one on a CPU.
- I've seen LOTS of defective GPUs

An investigation of the effects of hard and soft errors on graphics processing unit-accelerated molecular dynamics simulations

Robin M. Betz, Nathan A. DeBardeleben and Ross C. Walker

Deterministic Stable MD (using single-precision)

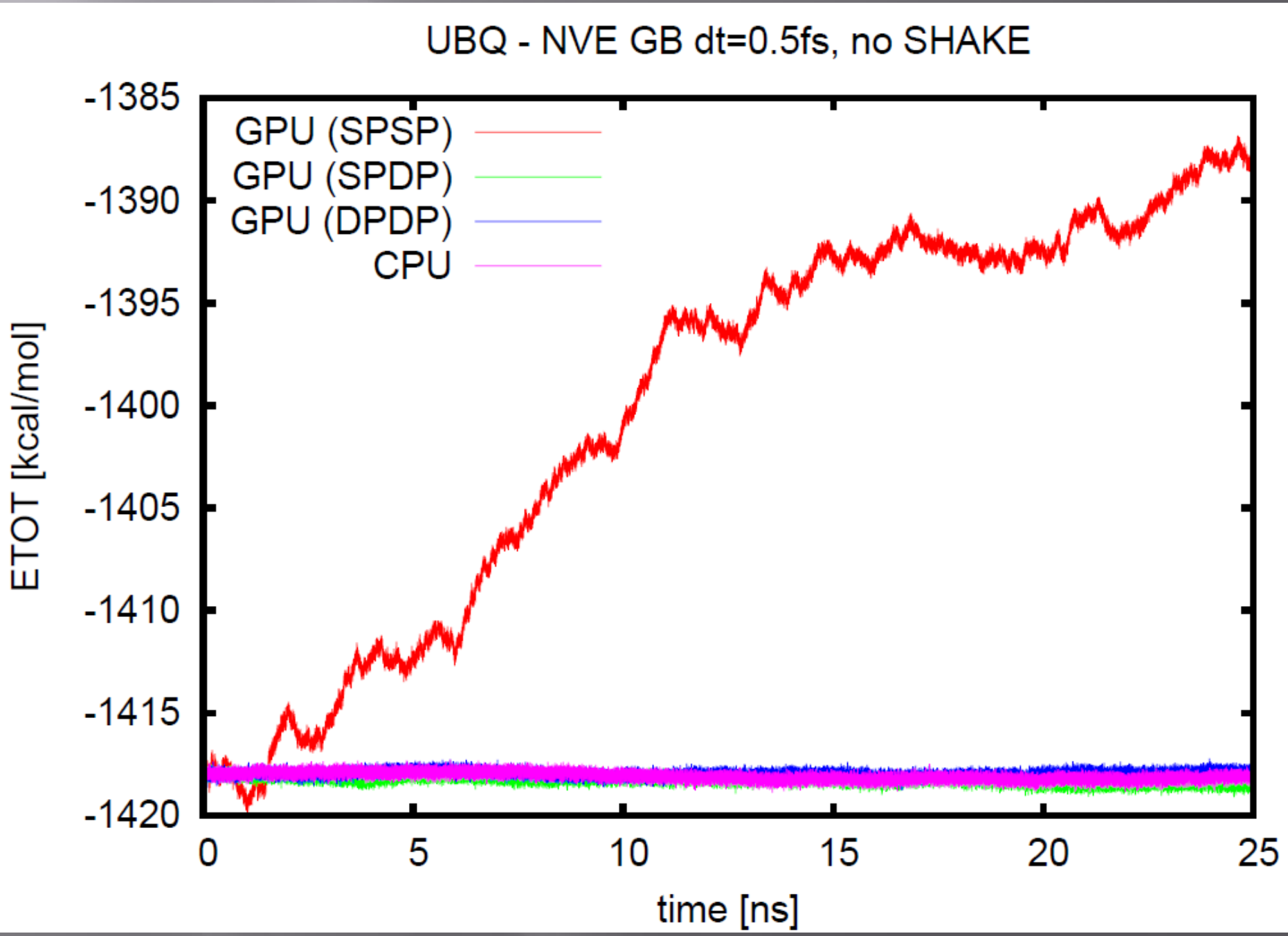
- Acceptable force error is ~10^{-5}
- Single-precision error is ~10^{-7}
- So calculate forces in single precision, but accumulate in extended precision
- Before Kepler, we used double-precision
- GK104 made it necessary to switch to 64-bit fixed point
- But this then allowed us to exploit its fast Atomic Adds as well

Use 64-bit fixed point for accumulation

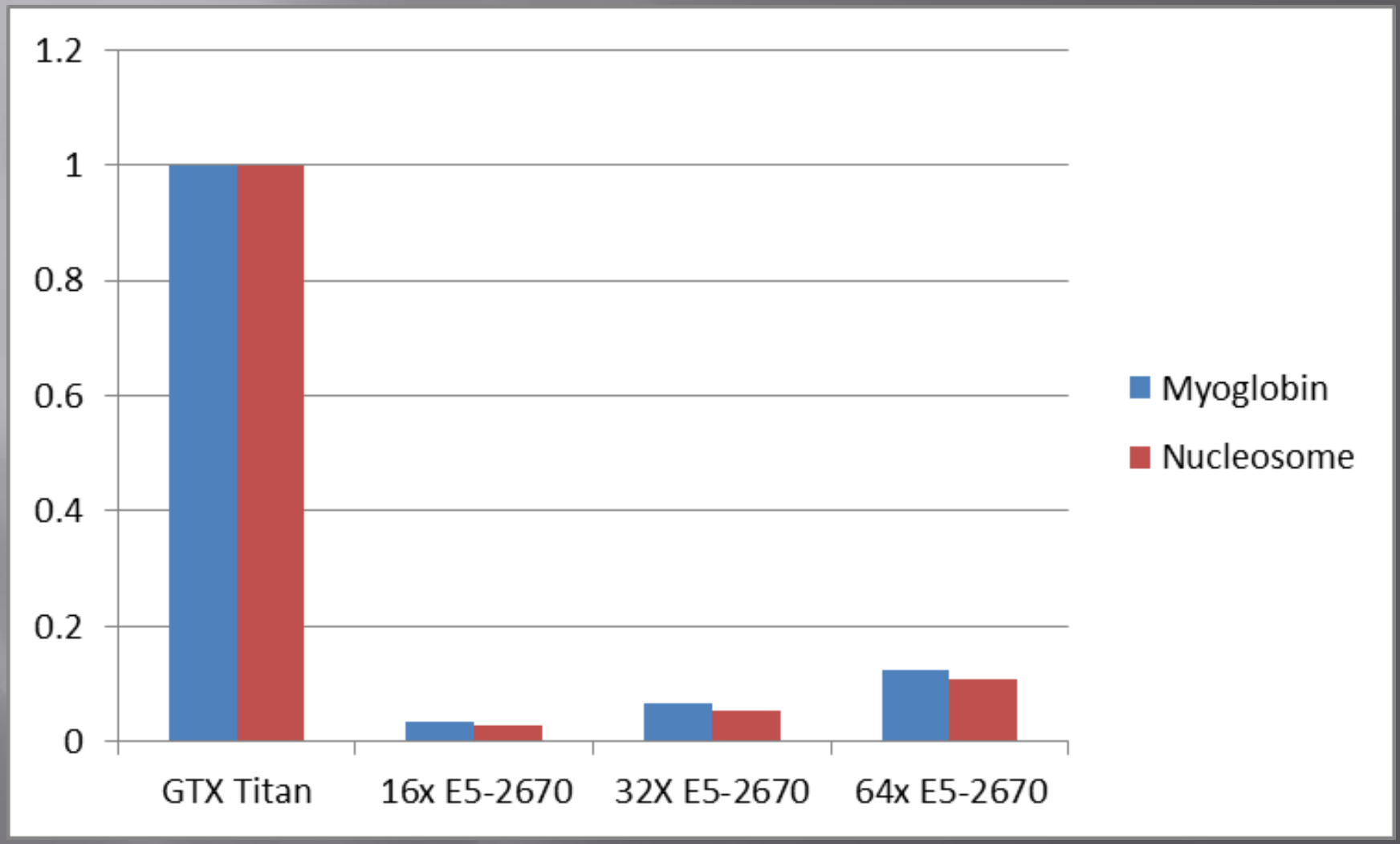
- Each iteration of the main kernel in PMEMD uses 9 double-precision operations
- Fermi double-precision was ¼ to 1/10th of single-precision
- GTX6xx double-precision is 1/24th single precision!
- So accumulate forces in 64-bit fixed point
- Fixed point forces are *perfectly* conserved
- 3 double-precision operations per iteration
- Integer extended math (add with carry) is 32-bit!

Something for Everyone

- DPFP 64-bit everything
- SPFP 32-bit forces, 64-bit everything else
- SPXP Mostly 32-bit, 64-bit system state


UBQ - NVE GB dt=0.5fs, no SHAKE
GPU (SPSP)
GPU (SPDP)
GPU (DPDP)
CPU
-1385
-1390
-1395
-1400
-1405
-1410
-1415
-1420
ETOT [kcal/mol]
0
5
10
15
20
25
time [ns]

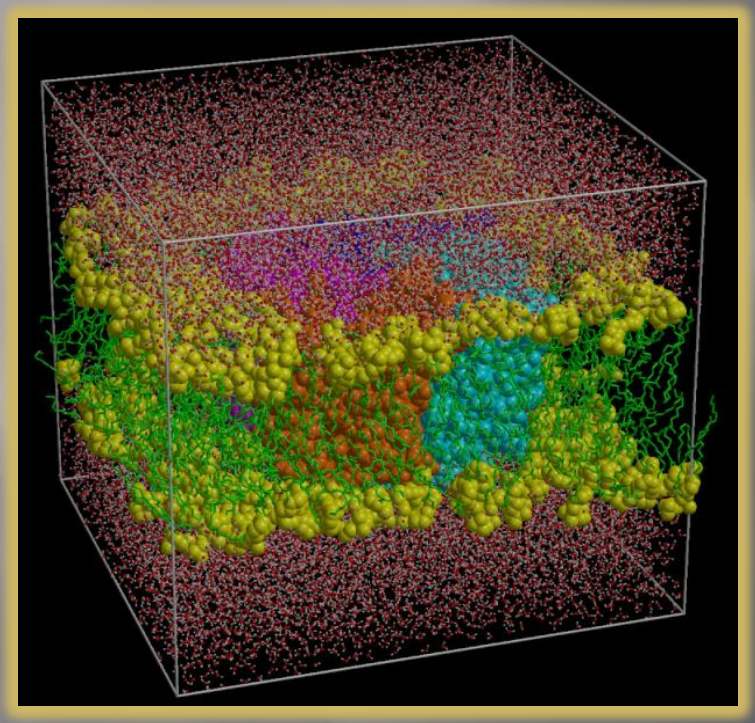
Generalized Born Performance



No surprise…

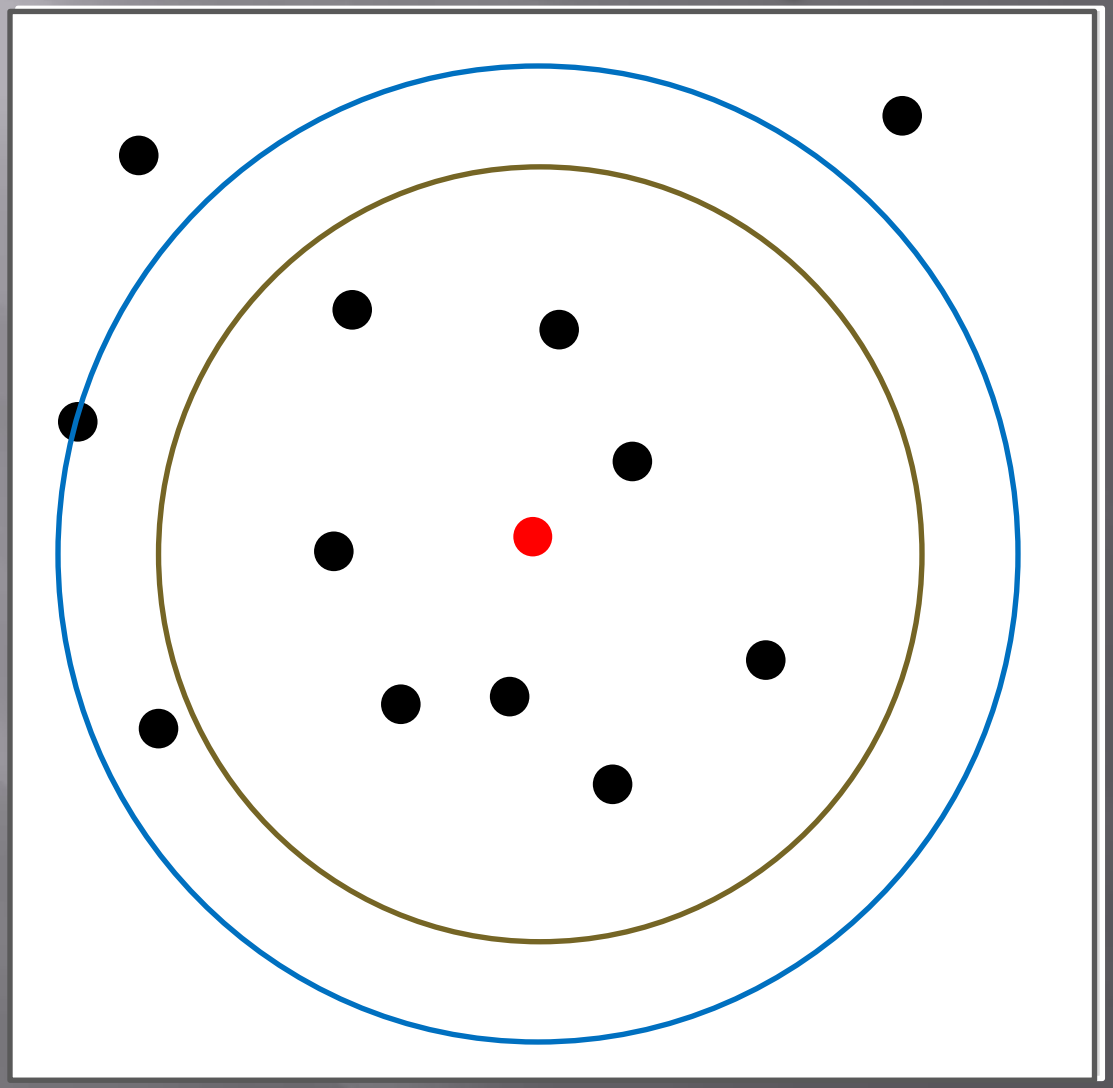
- Generalized Born is $O(N^2)$
- It's embarrassingly parallel
- Calculates all possible interatomic interactions
- If this isn't fast, you're not doing it right…

Particle Mesh Ewald (PME)

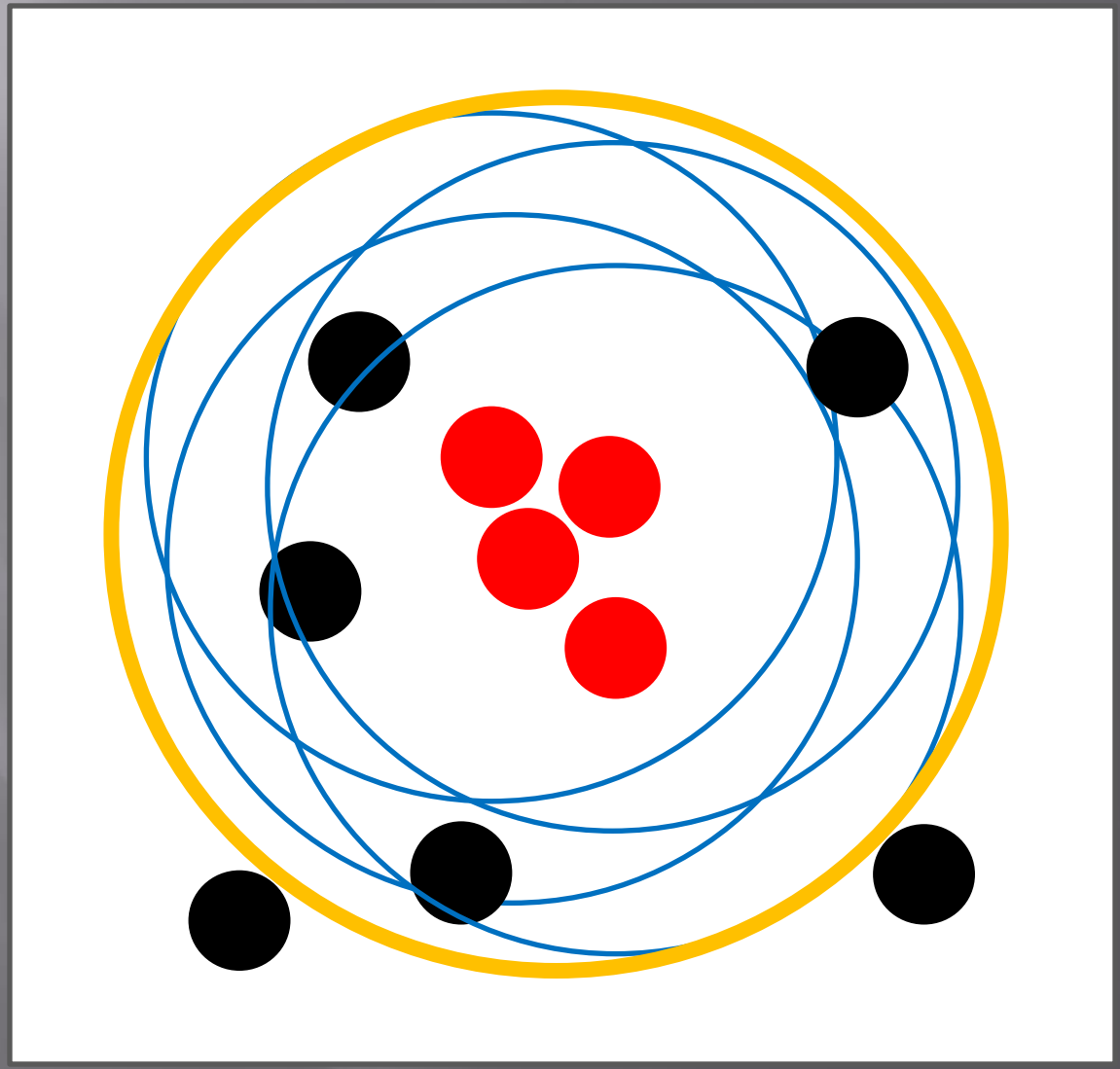


- O(N log N) Annoyingly Parallel
- More relevant than Generalized Born
- Rate-limited by a 3D FFT
- Approximates long-range interactions

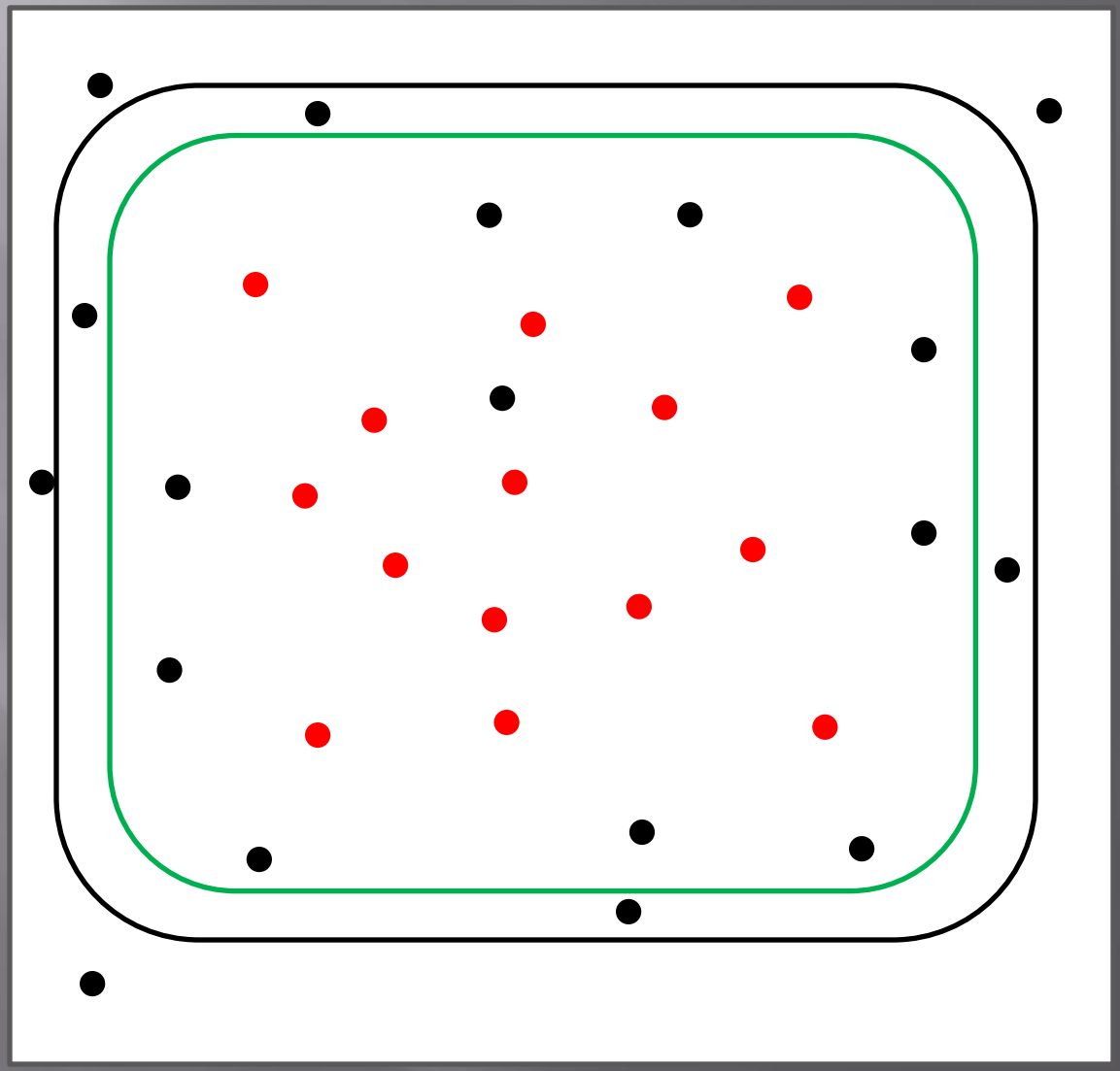
Nonbond Cutoff plus Skin



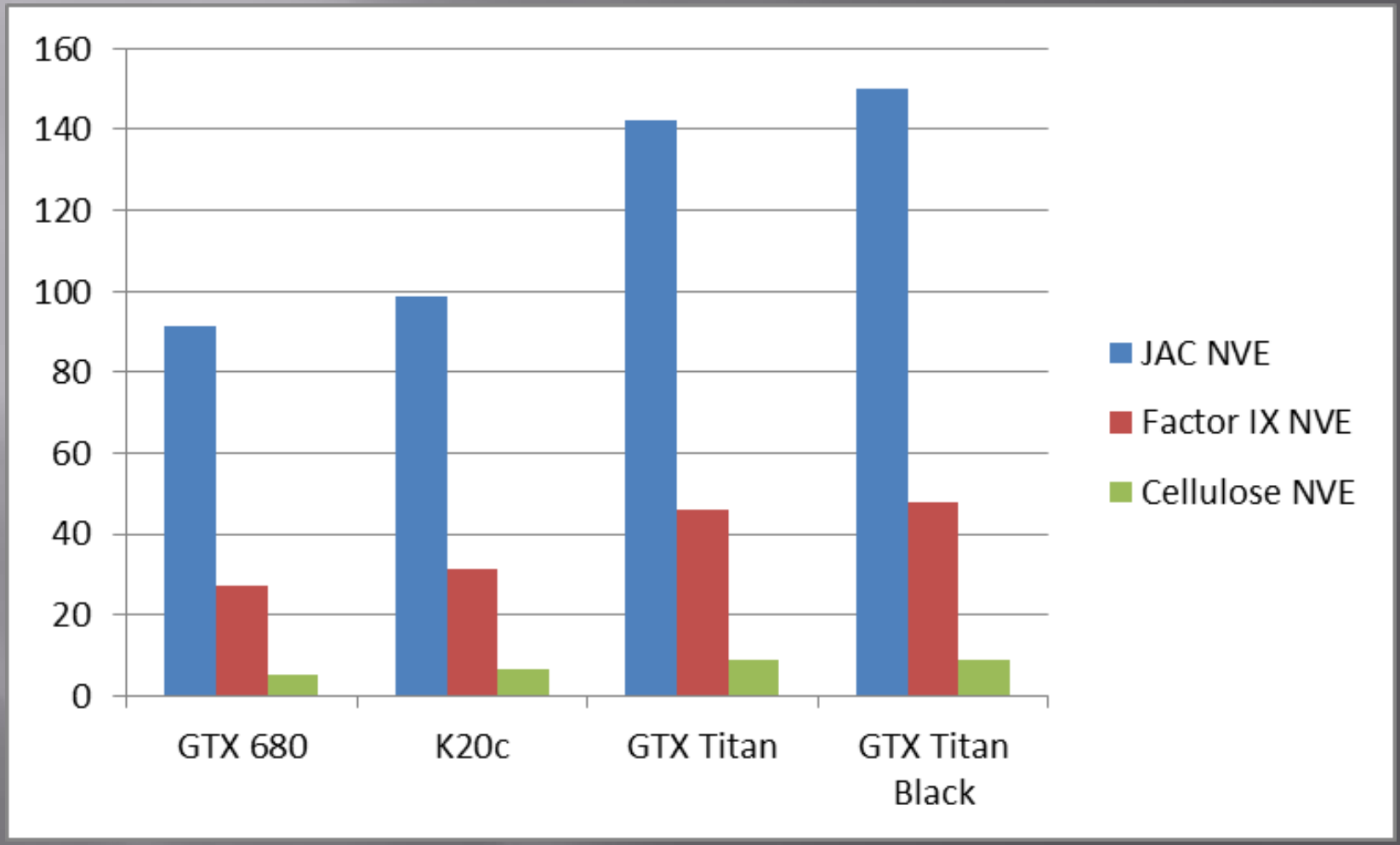
In Parallel



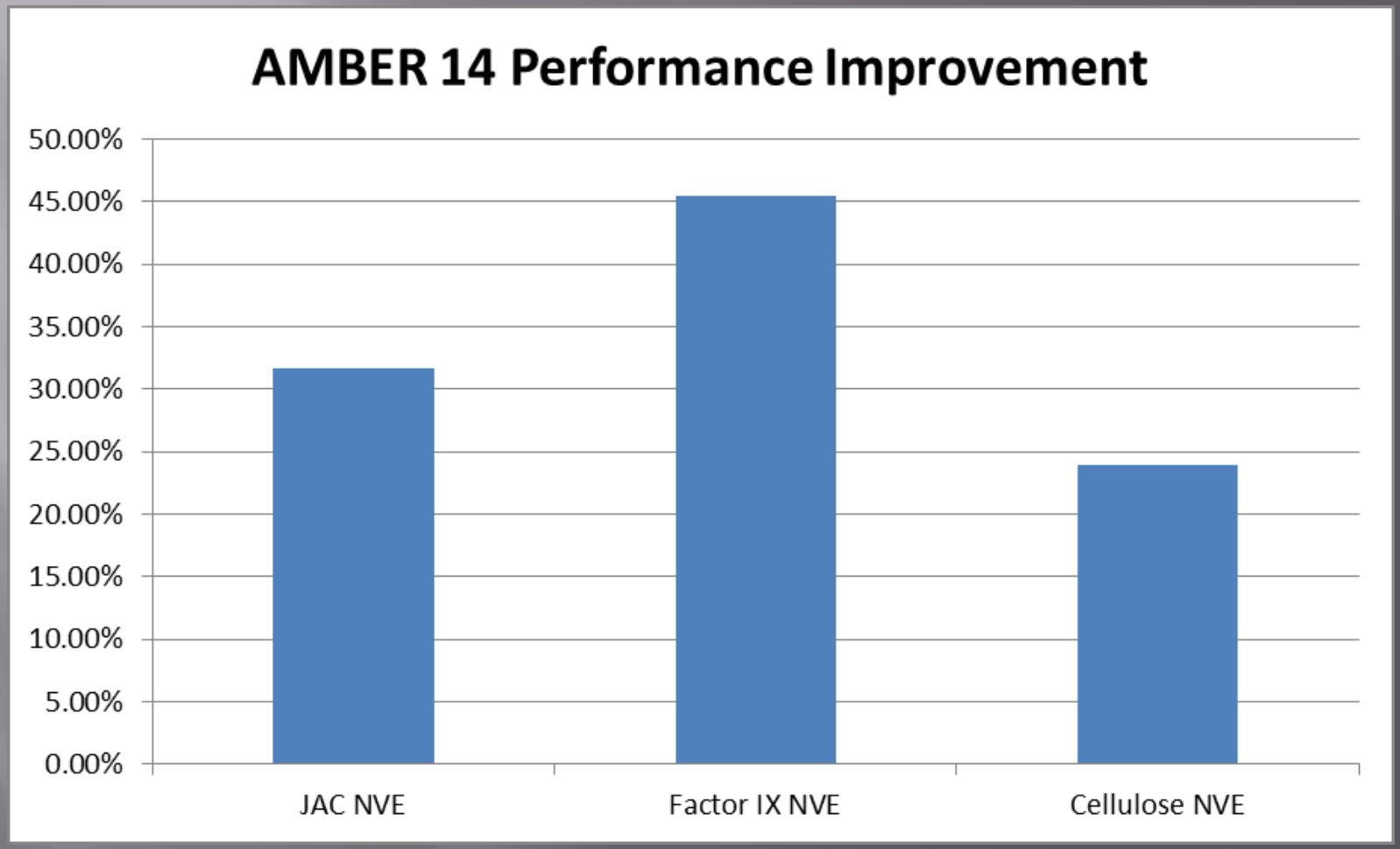
Bounding Boxes



PME Performance (AMBER 14)



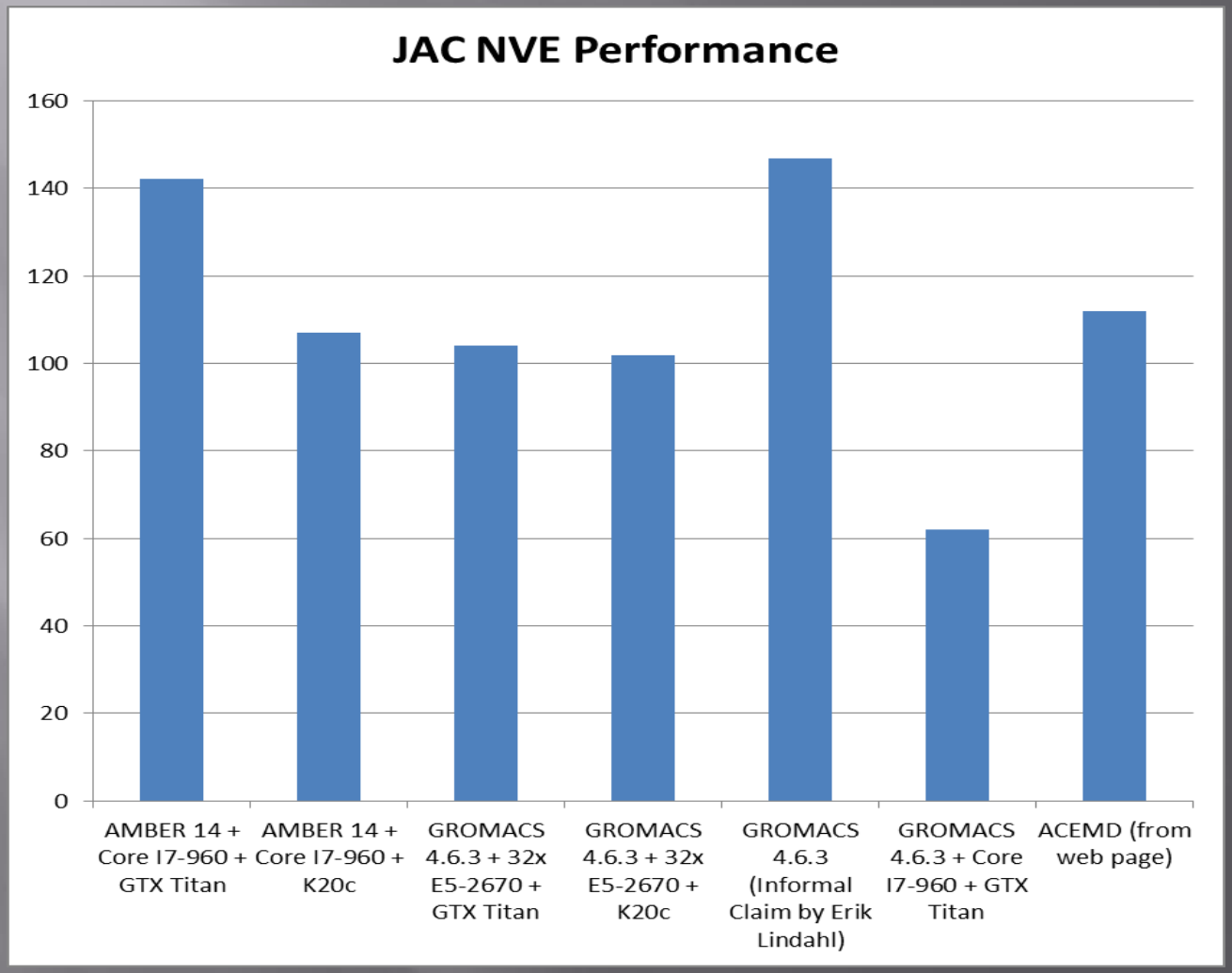
Compared to AMBER 12



Relative Performance

- No one wants to just run JAC NVE
- Lots of fascinating “optimizations” and under the hood hacks
- Run on a couple really high-end CPUs
- Not deterministic, not numerically stable

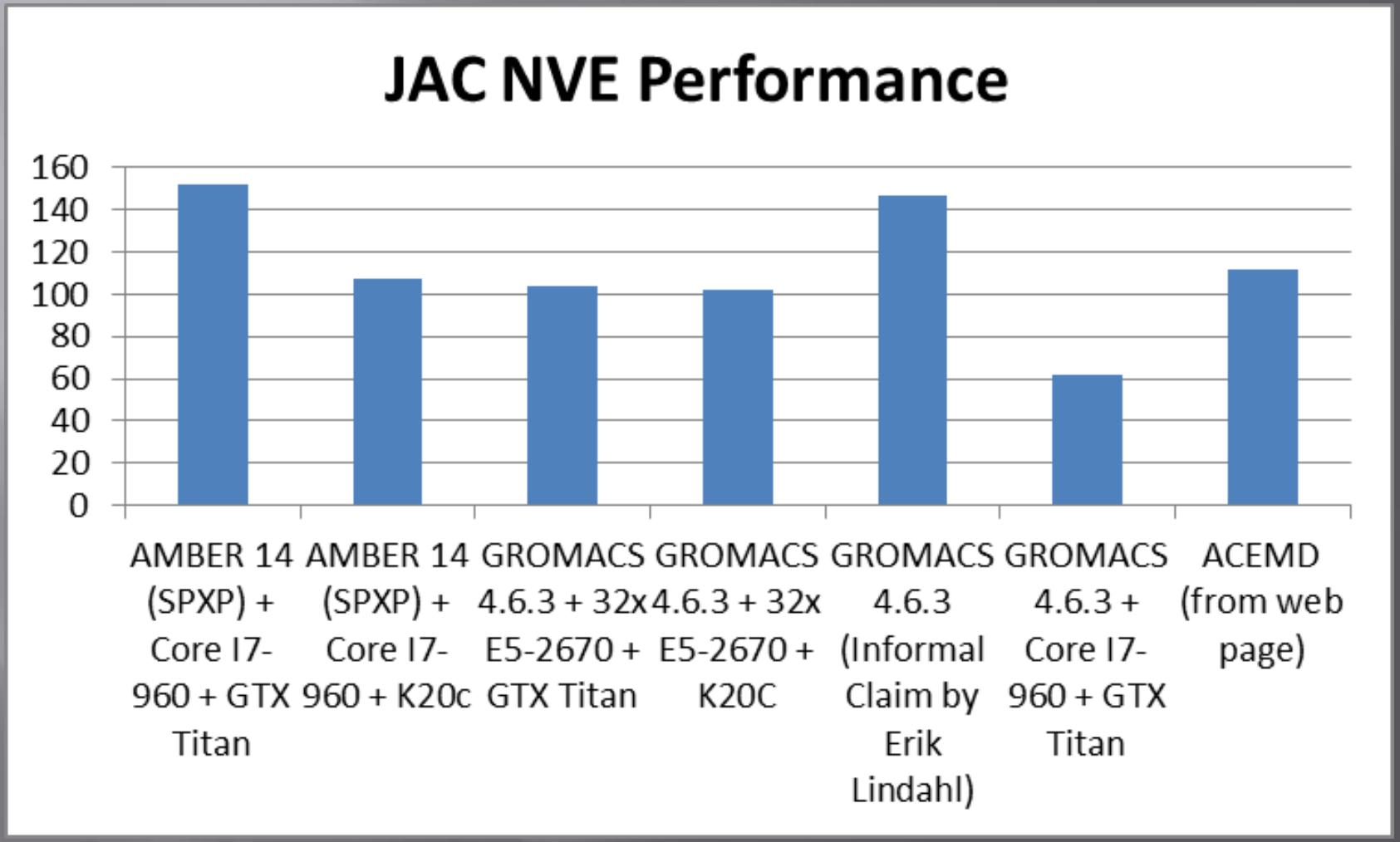
Straight up Relative Performance



Not so fast…

- GROMACS isn't deterministic
- GROMACS isn't numerically stable
- SPXP isn't numerically stable, but it *is* deterministic
- Let's fix that plot…

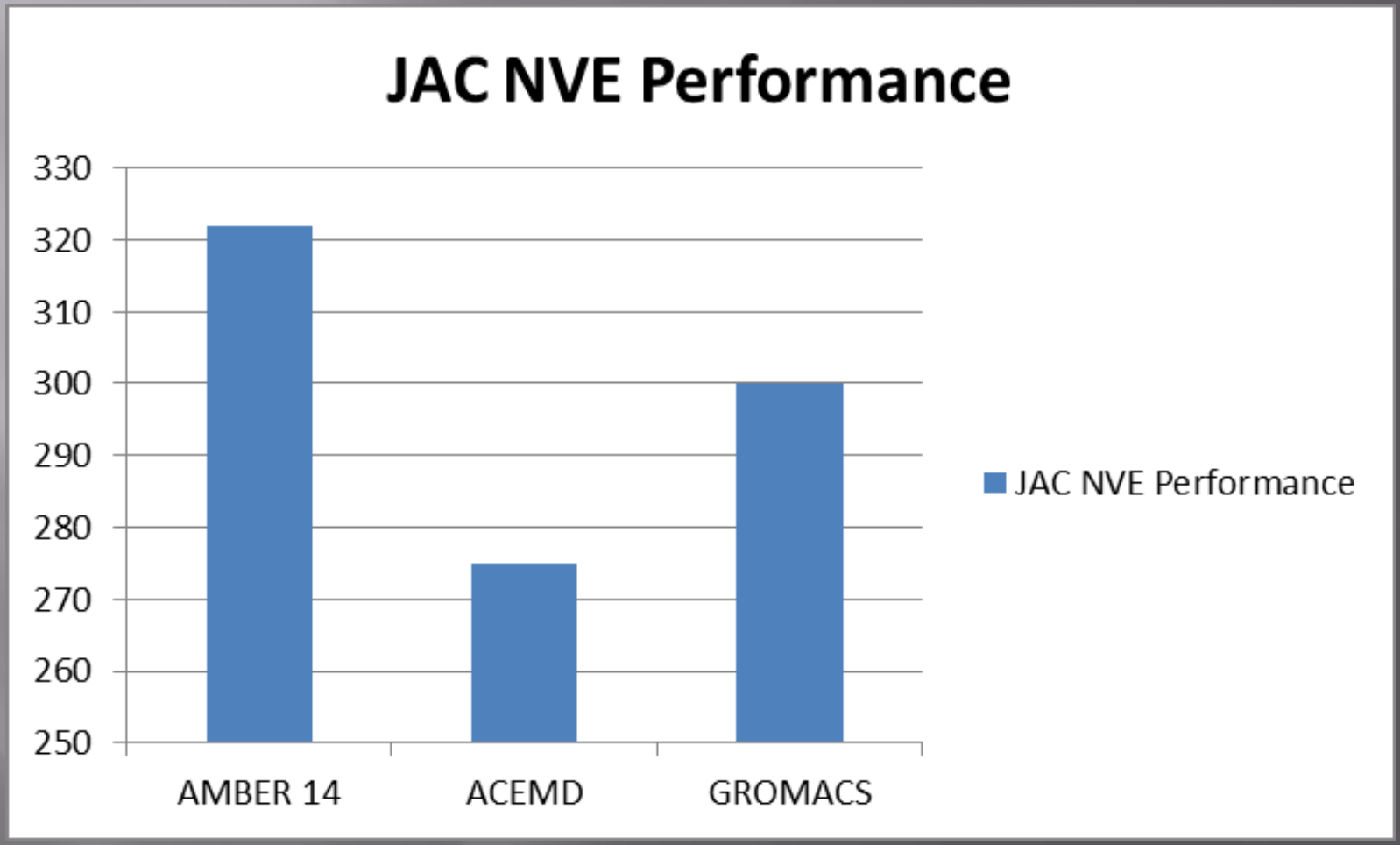
The Fast And The Spurious



Down the rabbit hole…

- GROMACS rebuilds its neighbor list every 10 iterations in my tests, every 20 iterations to hit 147 ns/day (so let's do that too! Must be OK, right?)
- 4 or 5 fs timesteps are used to "double" performance
- So we reluctantly added support for Hydrogen Mass Repartitioning
- While we're at, nrespa==2 just 'cuz

AND AMBER 14 WINS AGAIN!*

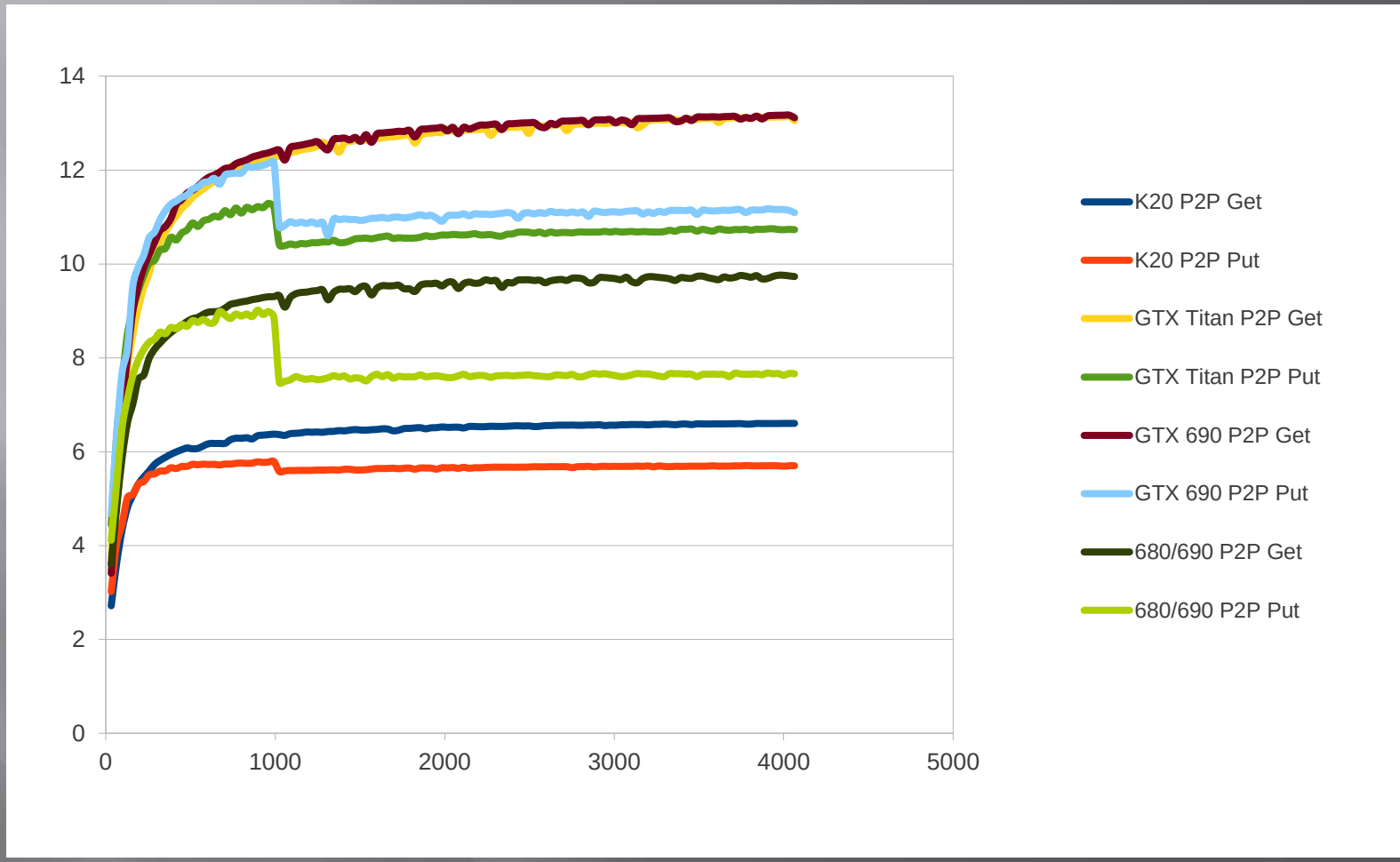


*Soon to be ~480 ns/day as we continue down this silly path

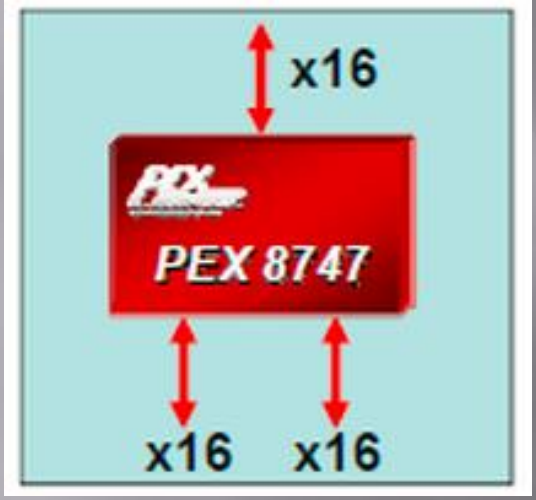
There's a better way…

- Instead of blowing gigabucks on CPUs, might I suggest buying several GPUs?
- You can use them for throughput *or* performance
- But make sure you buy a good motherboard
- Asus P9X79E-WS is my MB of choice

Peer to Peer (P2P) Copies

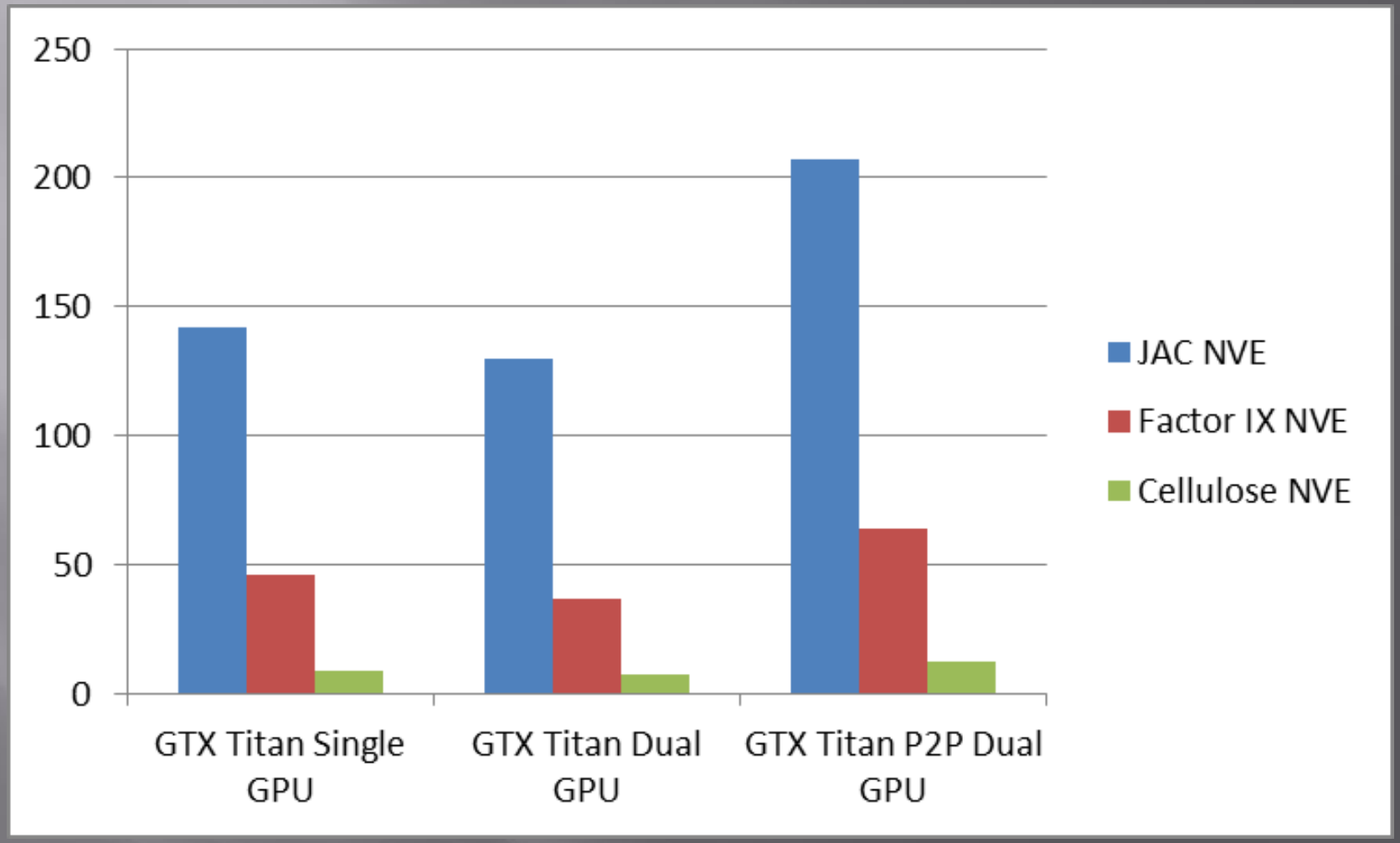


Works best with PLX Technology switches

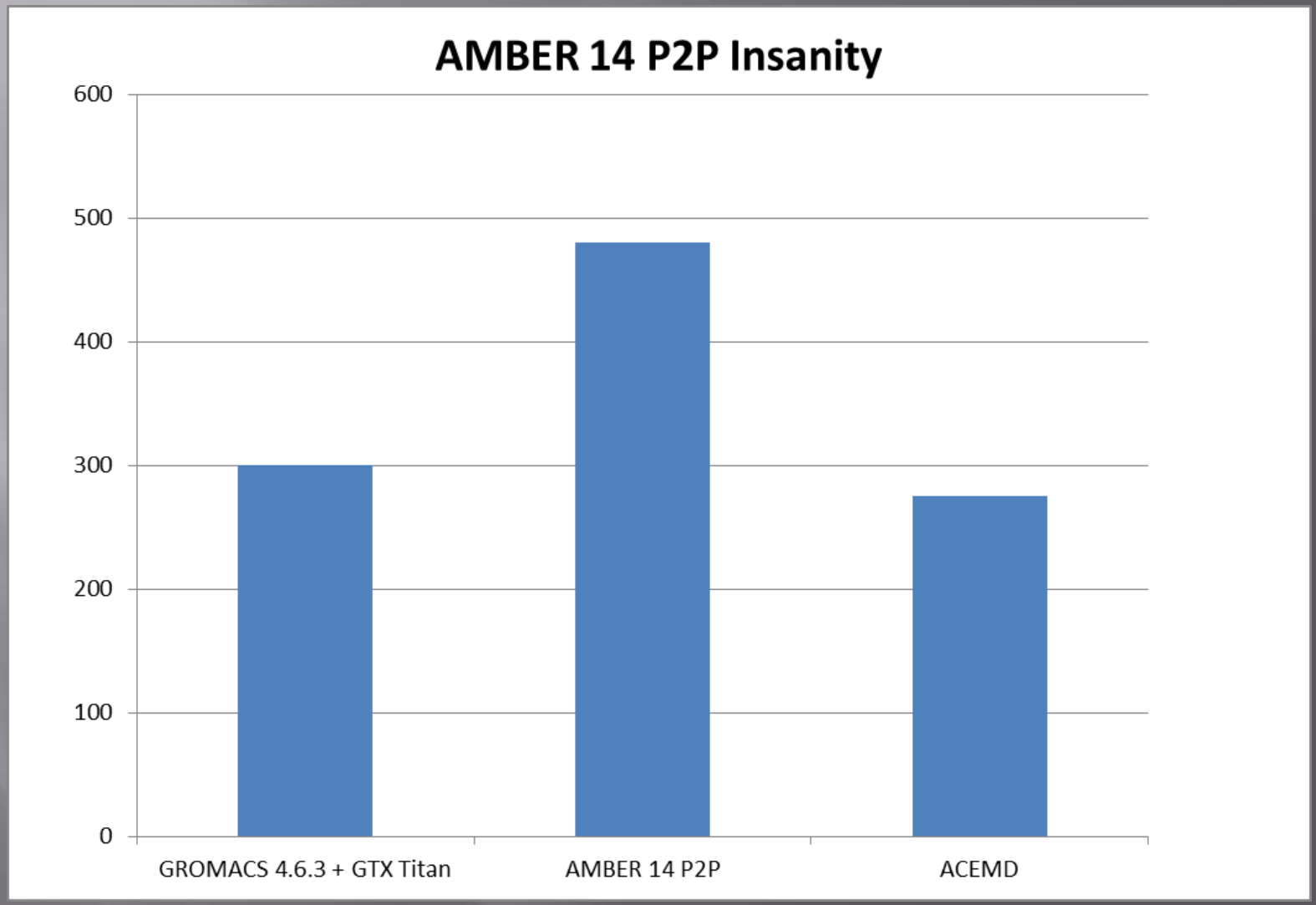


- Intel Chipset slows things down by ~20%
- PEX 8747 Allows pairs of GPUs to talk directly to each other (Asus P9X79-E WS motherboard)
- PEX 8780 and PEX 8796 allow up to 4 GPUs to talk to each (but no shipping motherboard has one)

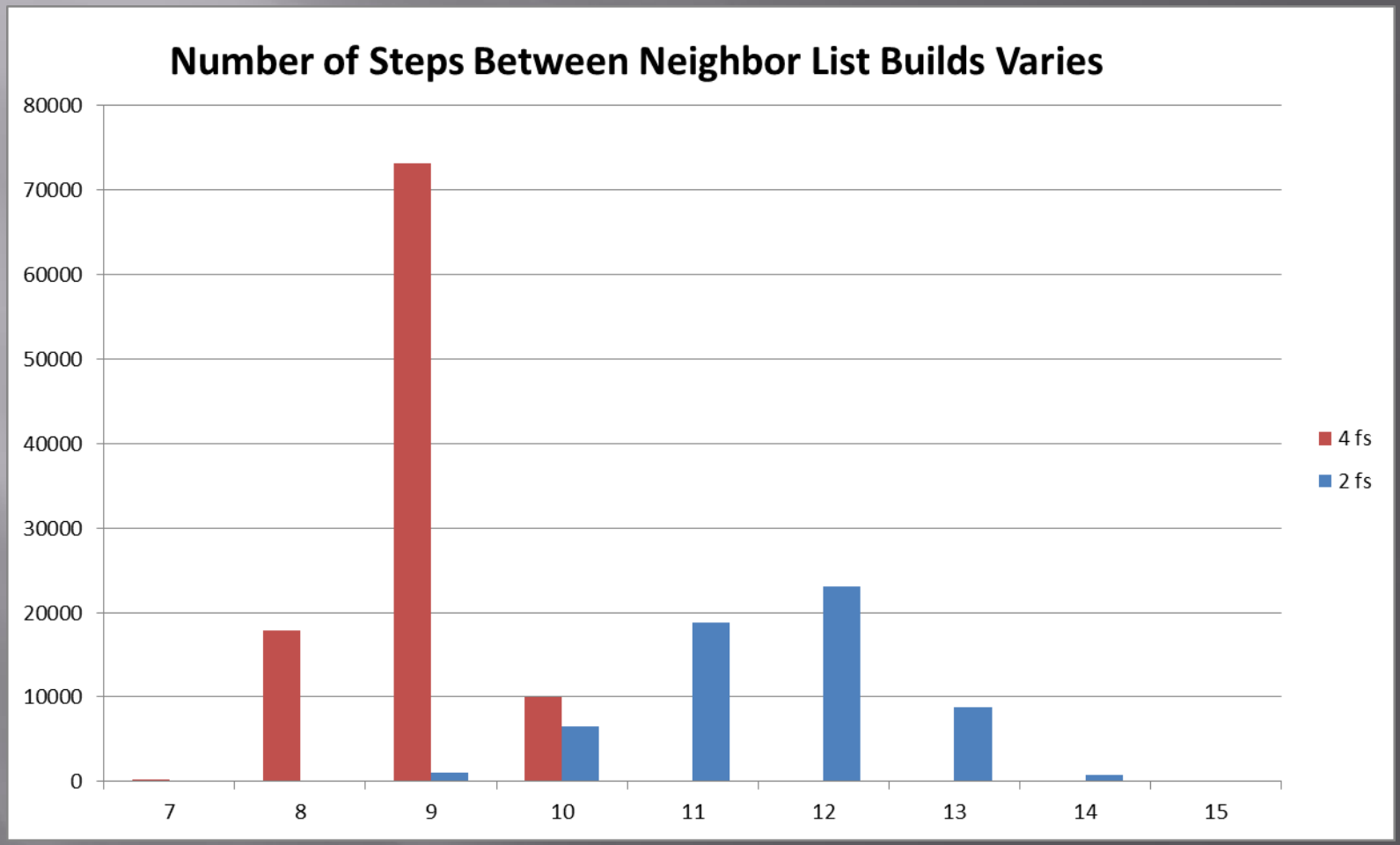
P2P Molecular Dynamics



P2P Science F(r)iction



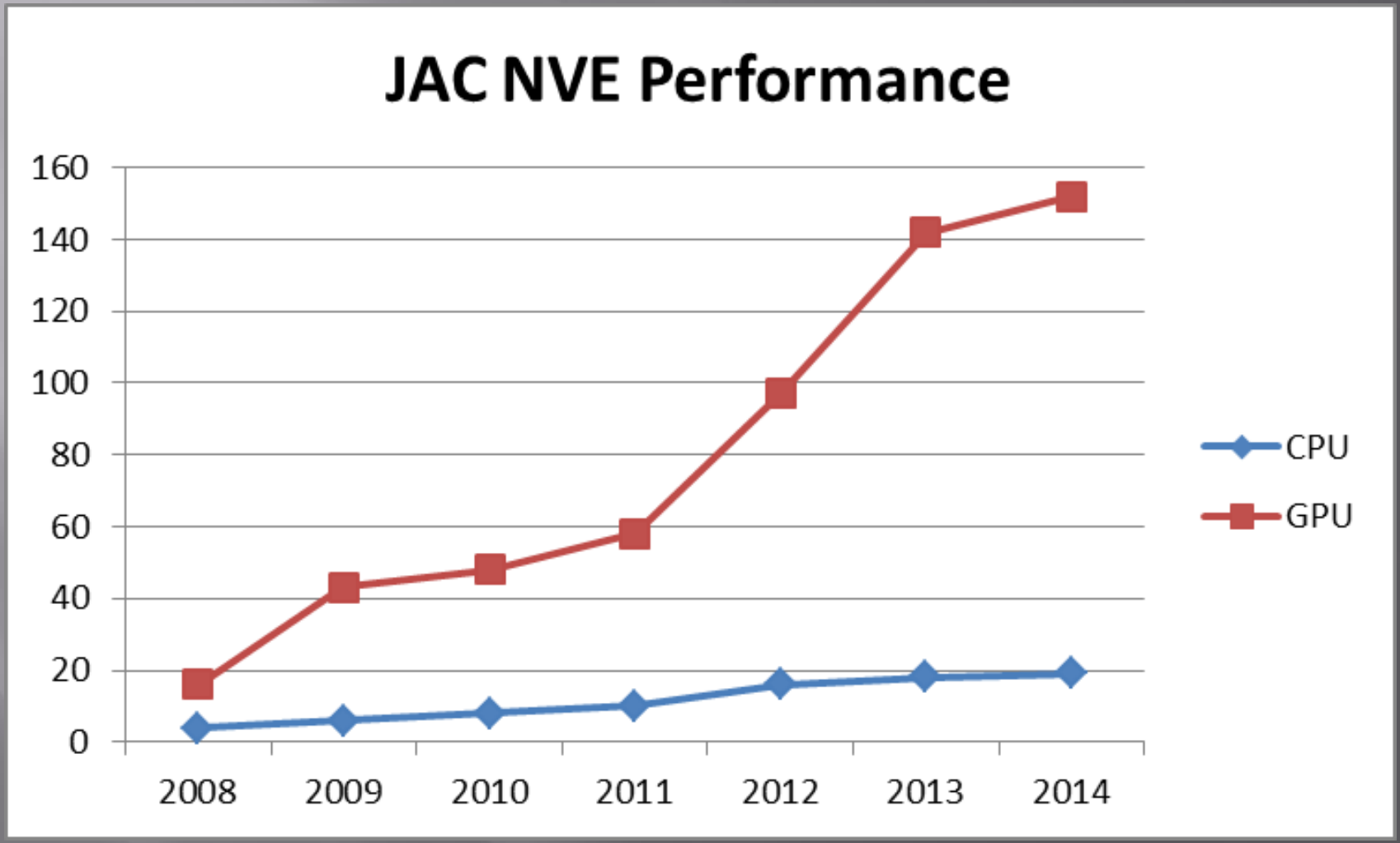
Science Fact



Are GPUs worth the work?

- Why didn't I just use SSE/AVX/OpenMP/ACC/Trendy WTF du jour?
- Without a ground up redesign around them, it just doesn't work
- Don't believe me? Go ahead, make my day…

AMBER Performance



Some Cold Water

- GPUs will never beat ANTON at this rate
- ANTON calculates an entire timestep in 15 microseconds
- Each iteration of PMEMD launches 23 kernels which consumes 76 microseconds
- SOL is ~2.2 microseconds/day
- Realistically, I think we'll hit the wall at 1 microsecond/day without driver improvements

The Cloud™

yum install make
yum install gcc-gfortran
download AMBER
cd amber/src
./configure –cuda gnu
make

And away you go…

Acknowledgments

AMBER: Ross Walker, David Case, Adrian Roitberg, Tom Cheatham, Andreas Goetz, Jason Swails, Ben Madej, Grace Liu

NVIDIA: Mark Berger, Duncan Poole, Sarah Tariq

AWS: Rejith Joseph, Teng-Yok Lee, Saurabh Baji, Peter Sirota, Deepak Singh