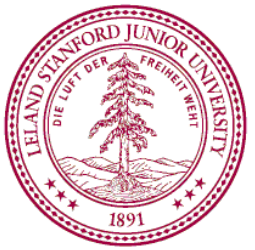
Of Jacquard Looms and Jaccard Coefficients

Multithreading Biomolecular Simulations in a GPU World

Imran Haque
Department of Computer Science
Stanford University

http://cs.stanford.edu/people/ihaque
http://folding.stanford.edu

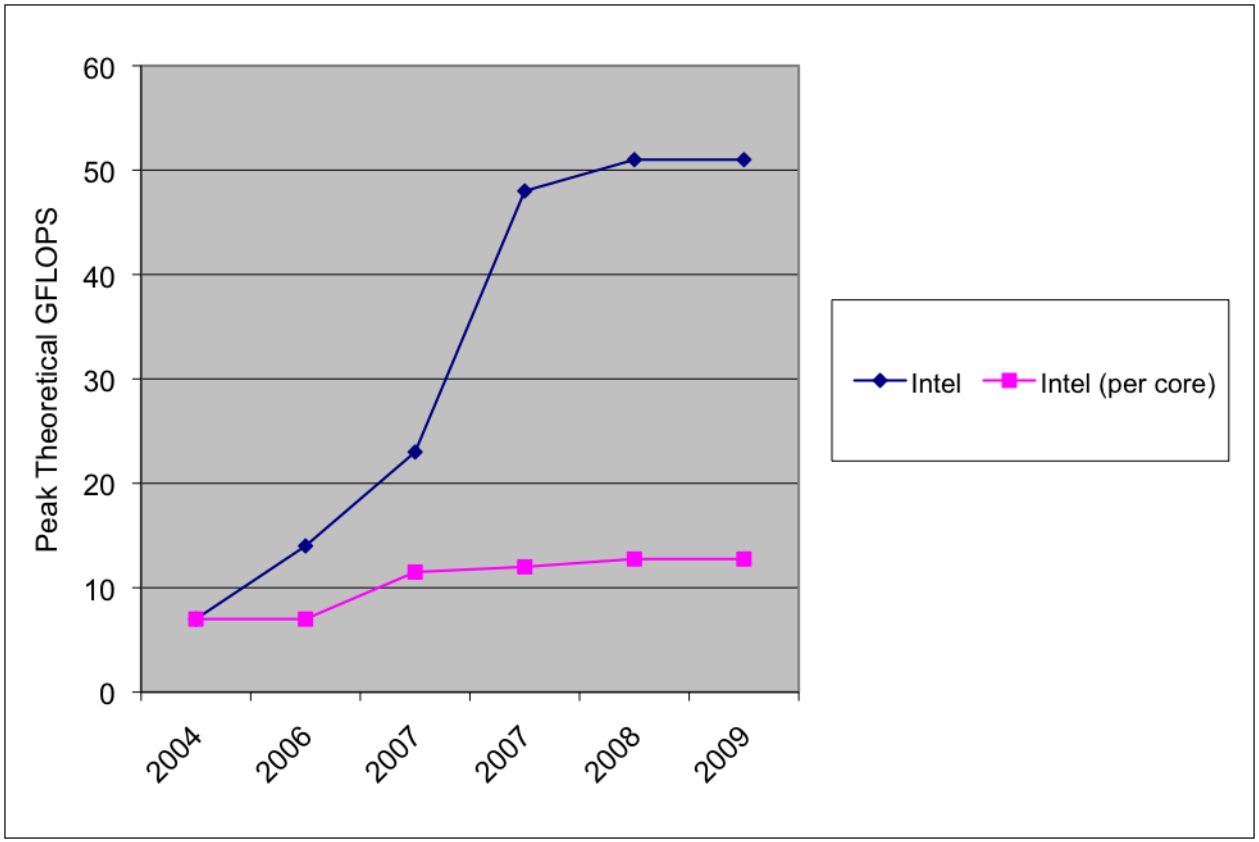


Folding@home distributed computing

NSF-NAIS Workshop on Intelligent Software
20 October 2009

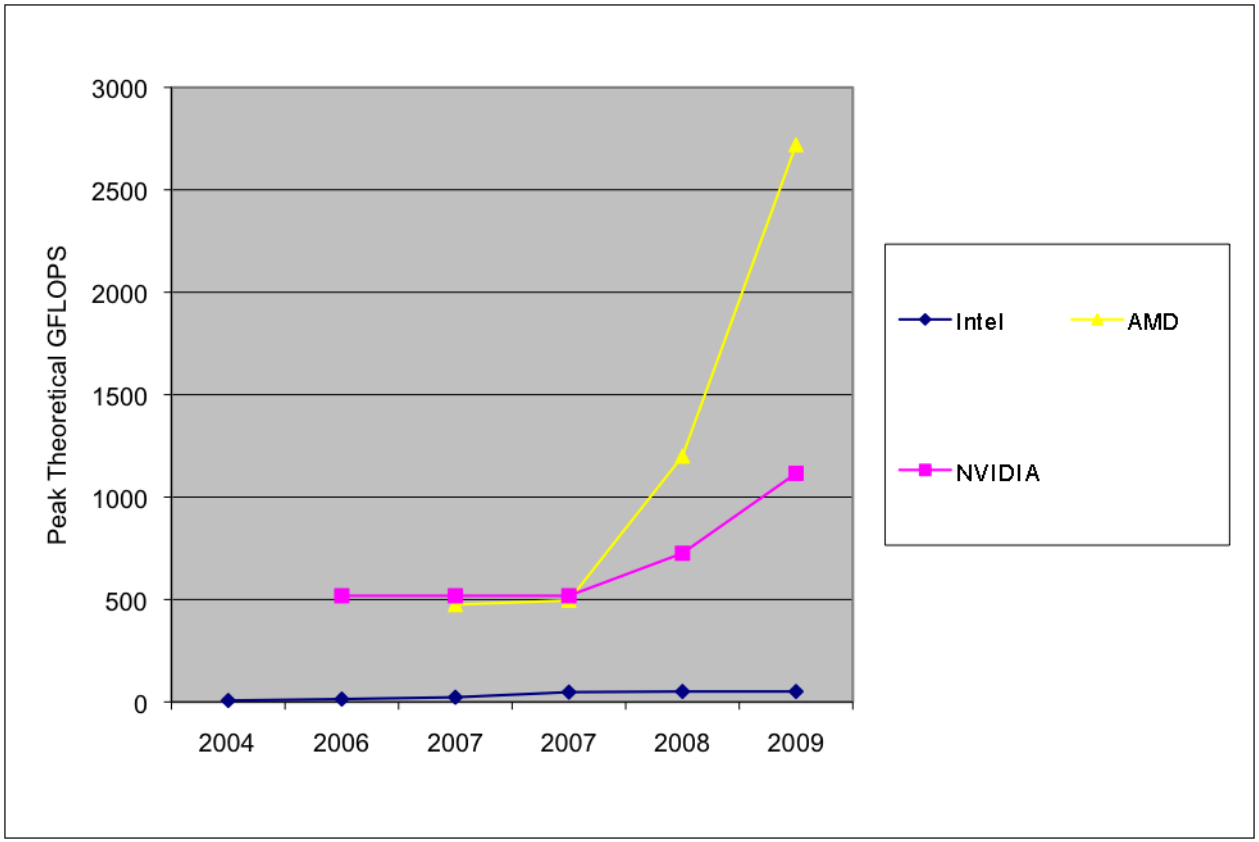
Annotated PDF version

Why bother with this GPU business?



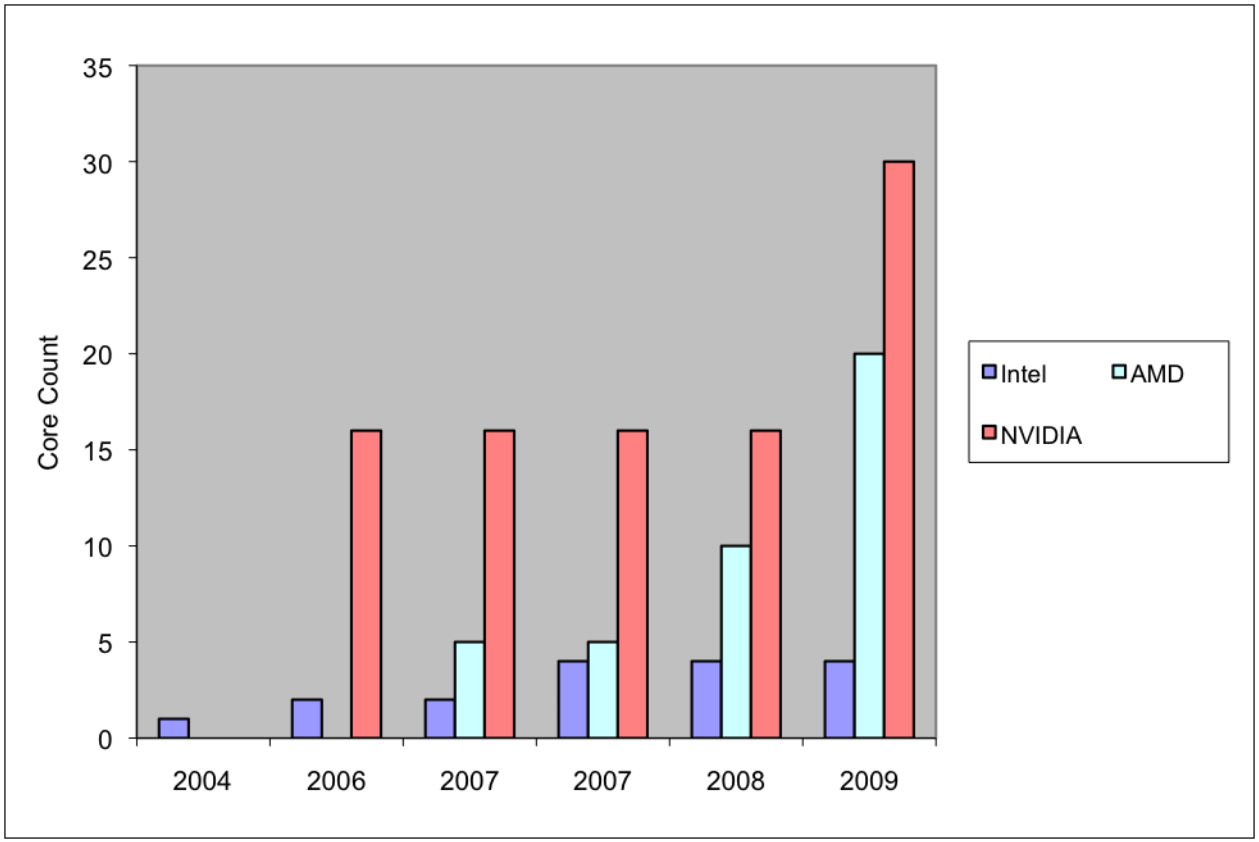
Aggregate performance continues to grow, but per-core performance has stagnated.

Why bother with this GPU business?



GPUs have massively higher peak aggregate performance – largely because of their much higher core counts.

Why bother with this GPU business?



Multicore is the present, not the future –
and it's even *more* present on GPUs.
Go where the FLOPs are!

Why bother with this GPU business?

Client statistics by OS

OS Type	Native TFLOPS*	x86 TFLOPS*	Active CPUs	Total CPUs
Windows	204	204	214873	2864087
Mac OS X/PowerPC	4	4	4956	131305
Mac OS X/Intel	23	23	7272	101836
Linux	60	60	35373	434356
ATI GPU	1092	1152	10708	91510
NVIDIA GPU	2184	4608	18357	147739
PLAYSTATION®3	1036	2186	36746	857841
Total	4603	8237	328285	4628674

Total number of non-Anonymous donators = 1301581

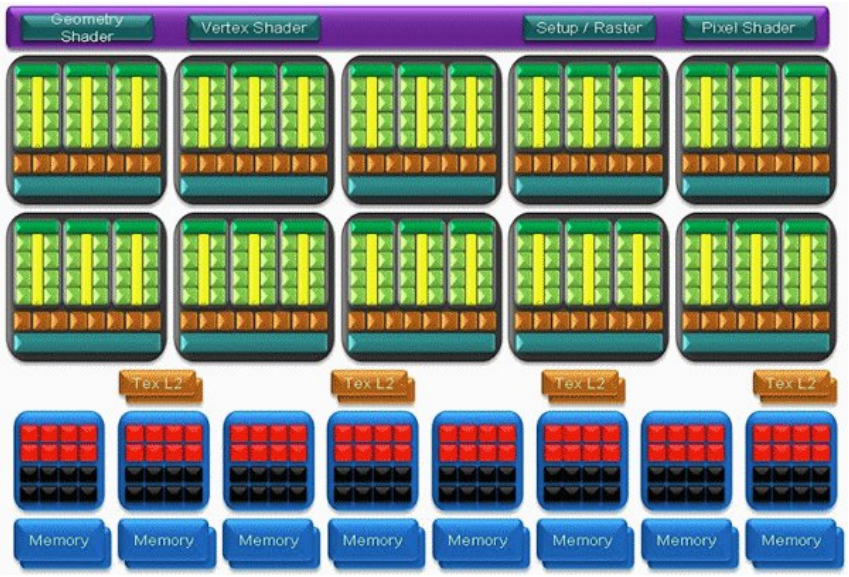
Last updated at Mon, 19 Oct 2009 07:03:20

Multicore is the present, not the future –
and it's even *more* present on GPUs.

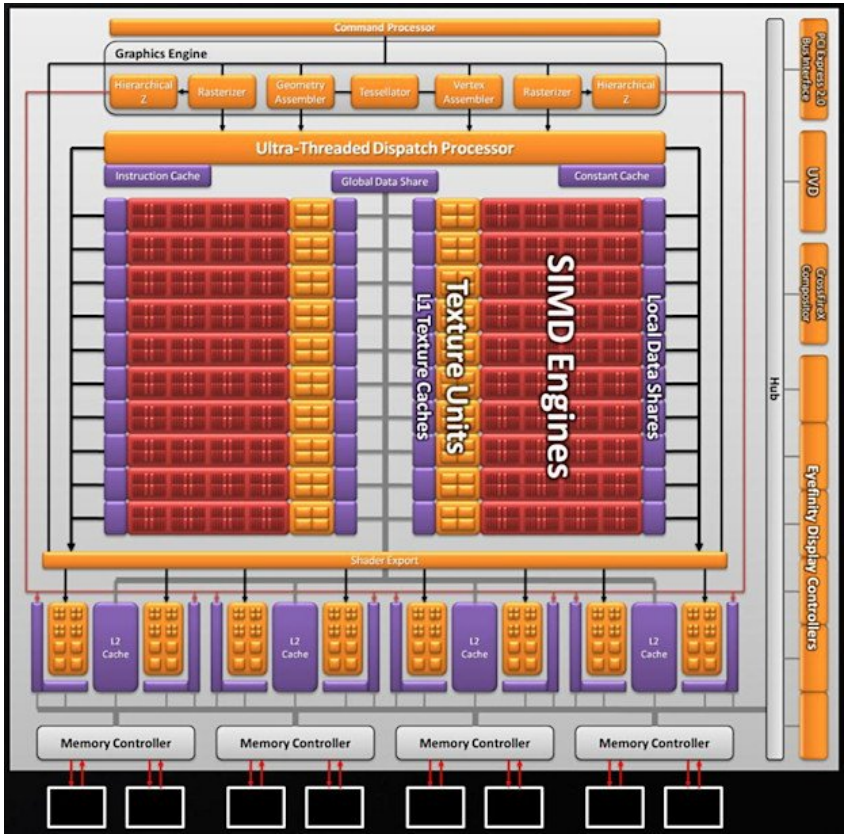
Go where the FLOPs are! Folding@home already does.

How many cores does it take to get to the center of a GPU?

NVIDIA GT200 (GeForce GTX 285)



AMD Cypress (Radeon HD 5870)



NVIDIA and AMD GPUs have very different internals, but share characteristics – wide SIMD, wide memory interfaces, high FLOP/bandwidth ratios.

Sources: NVIDIA, AMD

The original multithreading challenge

- Jacquard loom (1801)
- Automatic control of loom hooks to make intricate patterns
- **No one wants to be the "drawboy"**

Today's multithreading challenges

- Extracting parallelism from scientific problems
  - Embarrassing parallelism to none at all
- Efficient use of memory bandwidth
  - Von Neumann bottleneck
- Hardware reliability issues?
- Difficulties in rapid prototyping

Build libraries to limit the population of modern drawboys!

3-D Chemical Similarity Calculations on GPUs

FINDING PARALLELISM, WHEREVER IT MAY ROAM

Haque IS and Pande VS. PAPER – Accelerating Parallel Evaluations of ROCS. *J. Comp. Chem.* 2009.

Introduction to Chemical Similarity Search

- S*imilar compounds (may) have similar properties*
- Given a query structure (known drug, screening hit), can you find “similar” compounds in a library?

- Many methods; usual result is a Tanimoto/Jaccard coefficient:

$$T_{A,B} = \frac{\langle A, B \rangle}{\langle A, A \rangle + \langle B, B \rangle - \langle A, B \rangle} = \frac{|A \cap B|}{|A \cup B|}$$

- **Embarrassingly parallel** across a library

PAPER – Introduction

- Model molecular volume as union of atom-centered isotropic Gaussians; consider overlap b/w molecules

$$\langle A, B \rangle = \int d\mathbf{r} \Phi_A(\mathbf{r}) \Phi_B(\mathbf{r}) \approx \sum_{i,j} \left(\int d\mathbf{r} \rho_{Ai} \rho_{Bj} \right)$$

$$\rho_{Ai} = p_i \exp\left(-\alpha_i ||\mathbf{r}||^2\right)$$

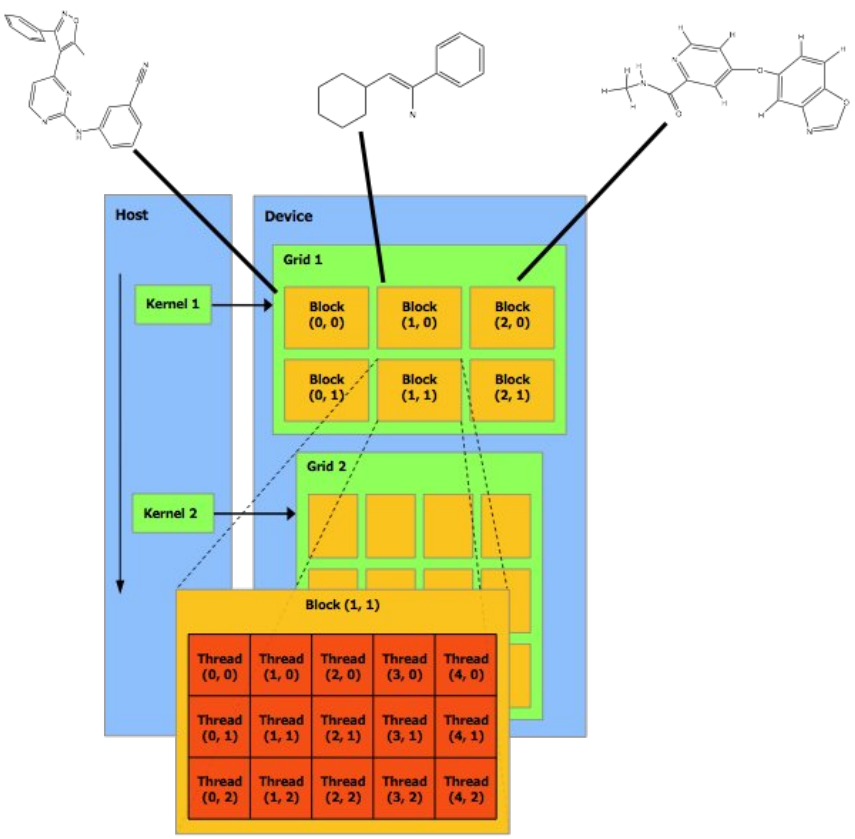
- 100s-1000s atom-atom pairs in inner loop
- Use BFGS local optimizer to maximize overlap
- Use initial conditions + library parallelism to fully load the GPU

https://simtk.org/home/paper

Haque IS and Pande VS. *J. Comp. Chem* 2009.

PAPER – Optimizations

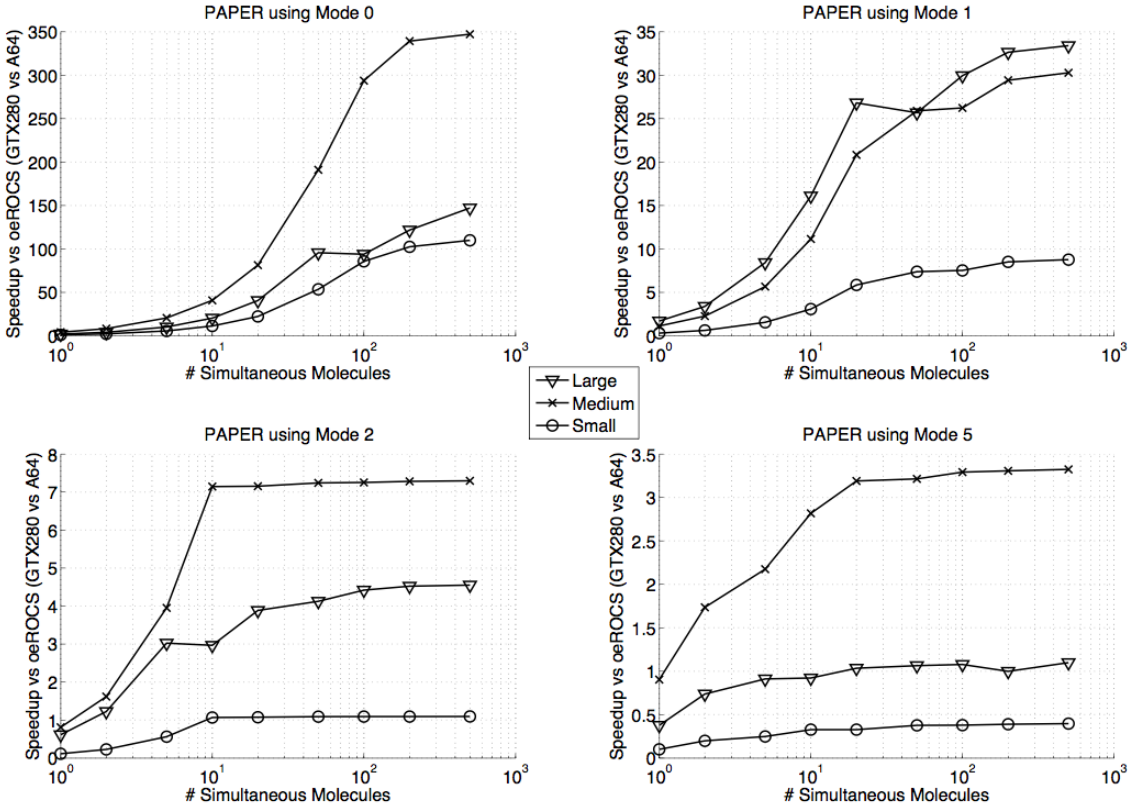
- Each molecule pair handled by a thread block
  (SIMD over inner loops)
- Run multiple molecules simultaneously:
  reduced sync overhead
- Entire optimizer on-chip:
  mitigate transfer latency



https://simtk.org/home/paper

Haque IS and Pande VS. *J. Comp. Chem* 2009.

PAPER – Results



- "Small" molecules have insufficient SIMD parallelism in inner loops to load GPUs well
- Need to run many molecules+starting conditions in parallel to achieve peak speedup

https://simtk.org/home/paper

Haque IS and Pande VS. *J. Comp. Chem.* 2009

Algorithmic Redesign for 1-D Similarity Calculations on GPUs

TAKING ANOTHER BRICK OUT OF THE MEMORY WALL

Haque IS and Walters WP. Row-Oriented Fast LINGOs. *In preparation.*

Introduction to LINGOs

- SMILES: graph-based linear molecular notation
  - Benzene -> c1ccccc1, Cyclohexylbenzene -> C1CCC(CC1)c2ccccc2
- LINGOs (Vidal et al. 2005) – compare two molecules by fragmenting SMILES into 4-char substrings, matching counts

$$T_{A,B} = \frac{\sum_{i=1}^{\ell} \left(1 - \frac{|N_{A,i} - N_{B,i}|}{N_{A,i} + N_{B,i}}\right)}{\ell}$$

- Grant et al. 2006 – build DFA from reference string, run query strings through automaton
  - **Branchy, memory-intensive: poor for GPUs!**

GPU LINGO – Algorithm

- Alternative: LINGOs as a multiset problem

$$T_{A,B} = \frac{|A \cap B|}{|A \cup B|}$$

- 4-substrings are identical with 32-bit integers - get 4 comparisons at the same time!
- Treat a molecule as a “bag” of numbers - sorted array of numbers with corresponding array of counts
- Calculate Tanimoto by algorithm like list merge – **easily parallelized, low memory usage**

Haque IS and Walters WP. In preparation.

GPU LINGO – Optimization

M_0L_0	M_0L_1	M_0L_2	M_0L_3
M_1L_0	M_1L_1	M_1L_2	M_1L_3
M_2L_0	M_2L_1	M_2L_2	M_2L_3
M_3L_0	M_3L_1	M_3L_2	M_3L_3

18% peak throughput

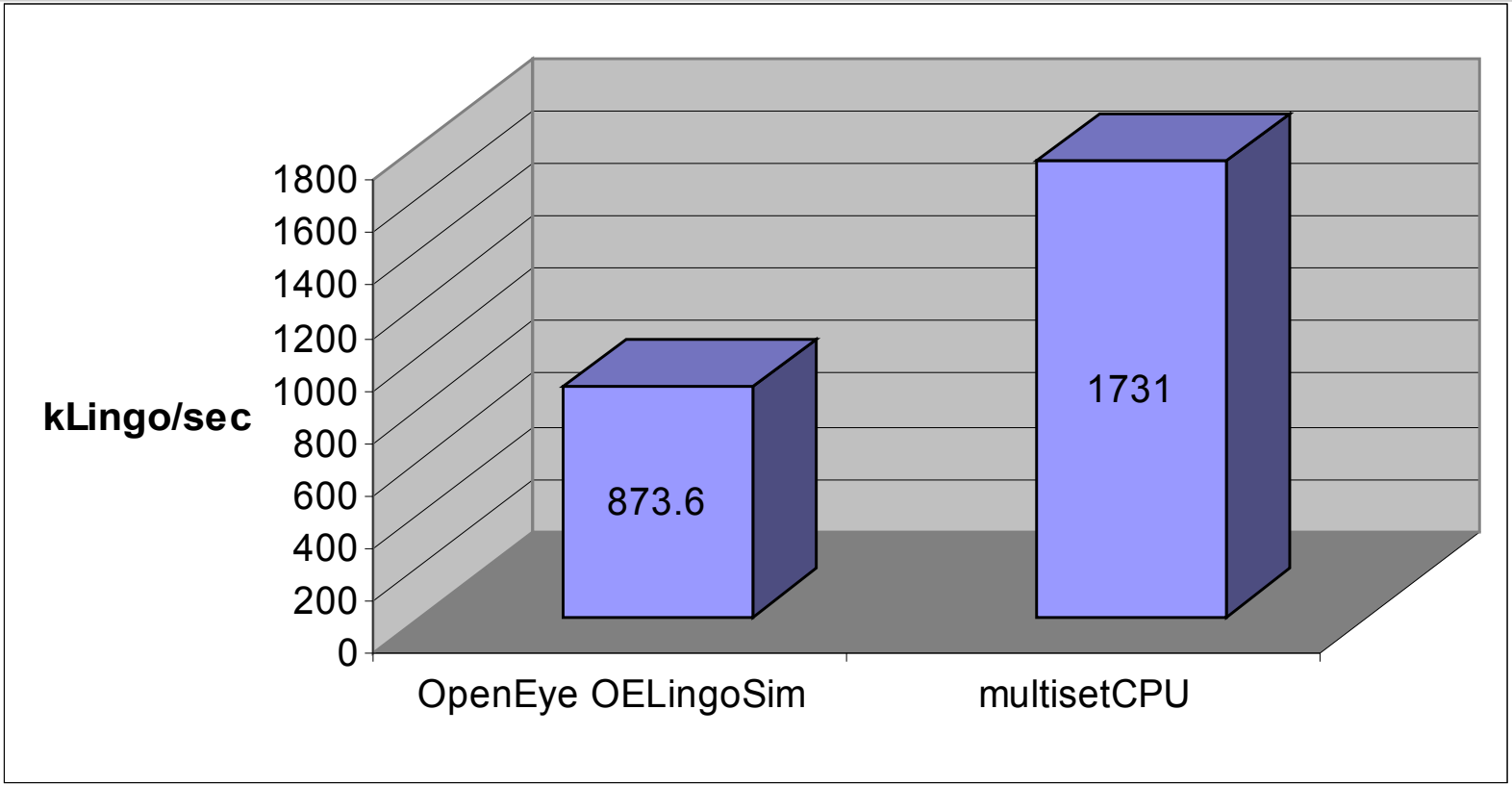
M_0L_0	M_1L_0	M_2L_0	M_3L_0
M_0L_1	M_1L_1	M_2L_1	M_3L_1
M_0L_2	M_1L_2	M_2L_2	M_3L_2
M_0L_3	M_1L_3	M_2L_3	M_3L_3

98% peak throughput

- Normal memory layout (left) imposes strided memory access on SIMD units
- Transposed layout coalesces memory accesses
- 2-D texture cache eliminates a barrier sync

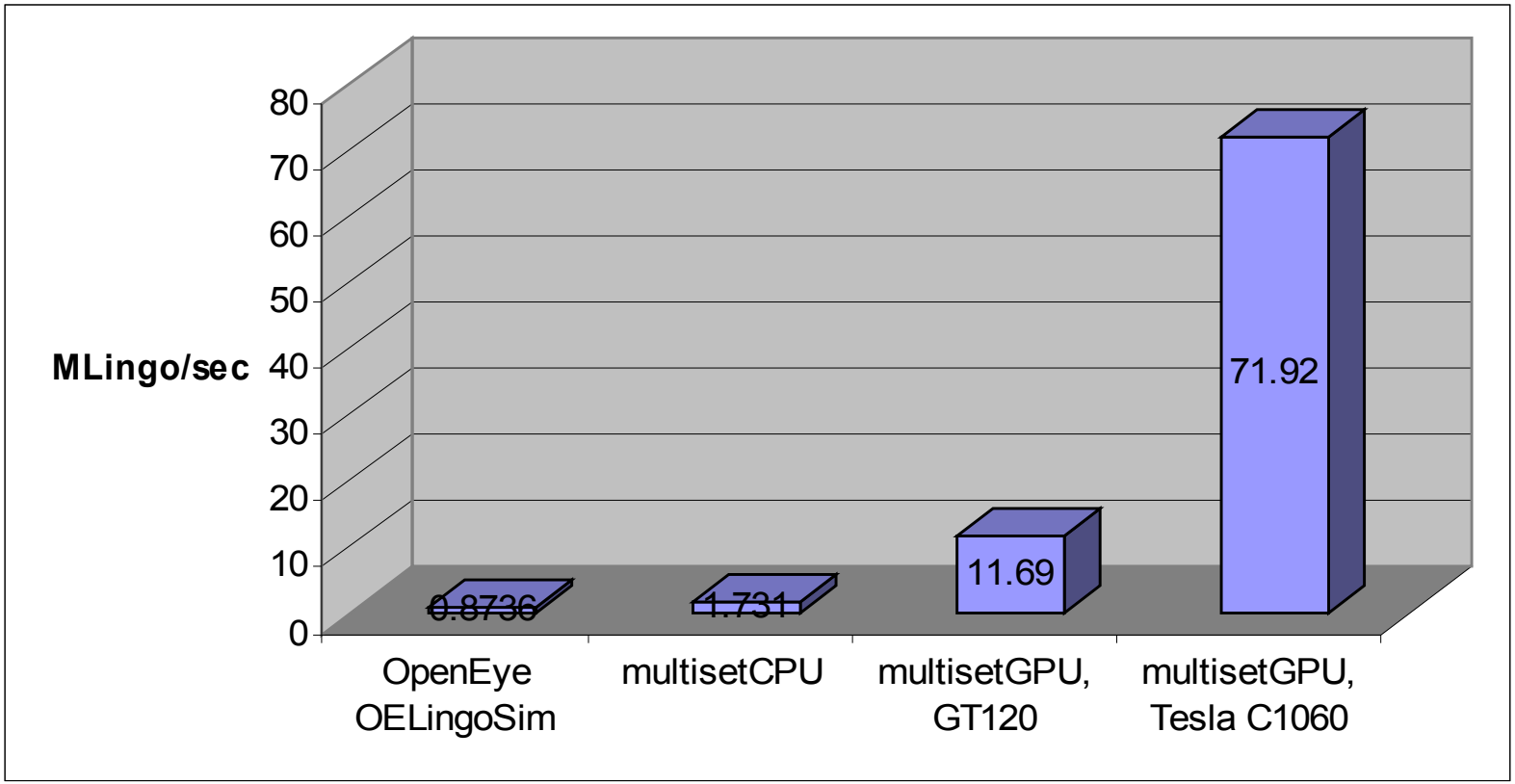
Haque IS and Walters WP. In preparation.

GPU LINGO - Results



Haque IS and Walters WP. In preparation.

GPU LINGO - Results



Redesigned algorithm:
82x faster on a GPU, 2x even on a CPU!

Haque IS and Walters WP. In preparation.

Investigating Memory Soft Errors in GPGPU

GPU MEMORY: ALONE IN THE (COSMIC-RAY) MOONLIGHT

Haque IS and Pande VS. Hard Data on Soft Errors – A Large-Scale Assessment of Real-World Error Rates in GPGPU. *Submitted to J. Comp. Chem; poster at Supercomputing 2009; arXiv:0910.0505v1.*

MemtestG80 – Motivation

- GPUs have their origin in **error-insensitive** consumer graphics
- Neither ECC nor parity on graphics memory

- **How suitable is the installed base of consumer GPUs** (and consumer-GPU derived professional hardware!) **for *error-sensitive* general purpose computing?**

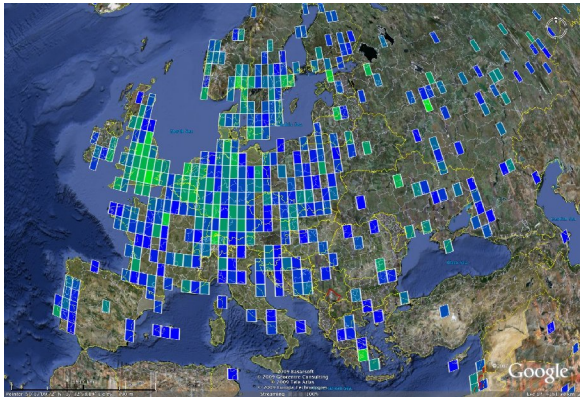
Haque IS and Pande VS. Submitted. arXiv:0910.0505v1

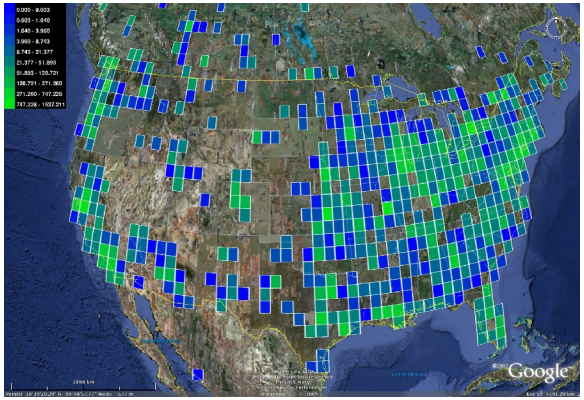
MemtestG80 – Methodology

- Wrote MemtestG80 – custom test software for NVIDIA GPUs, based on Memtest86+ for x86 PCs

https://simtk.org/home/memtest

- Expect a low error-rate and environment sensitivity, so must sample *many* cards in diverse environments
- Ran for ~7 months over 58,000+ NVIDIA GPUs on Folding@home (>800 TB-hr of testing)





Haque IS and Pande VS. Submitted. arXiv:0910.0505v1

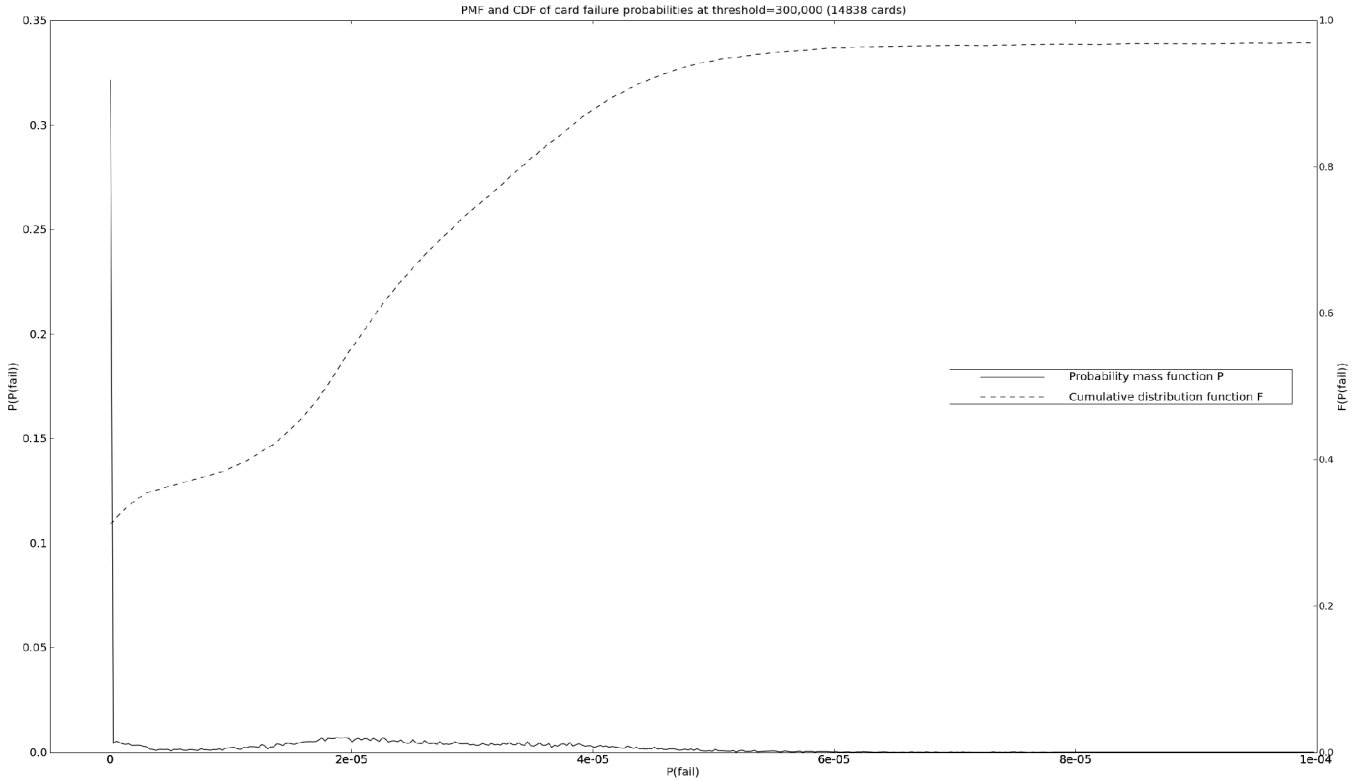
MemtestG80 – Results

- Negative control run on GeForce 8800 GTX and 8x Tesla C870 (consumer and GPGPU G80 cards)
  - Controlled environment, power, host hardware
  - 925,000 FAH-equivalent iters/Tesla, >1M on GeForce

- **No errors observed in control test**
  - Possible that environmental effects (better power, cooling) are in effect
  - Possible that Tesla cards are actually made of more reliable hardware

Haque IS and Pande VS. Submitted. arXiv:0910.0505v1

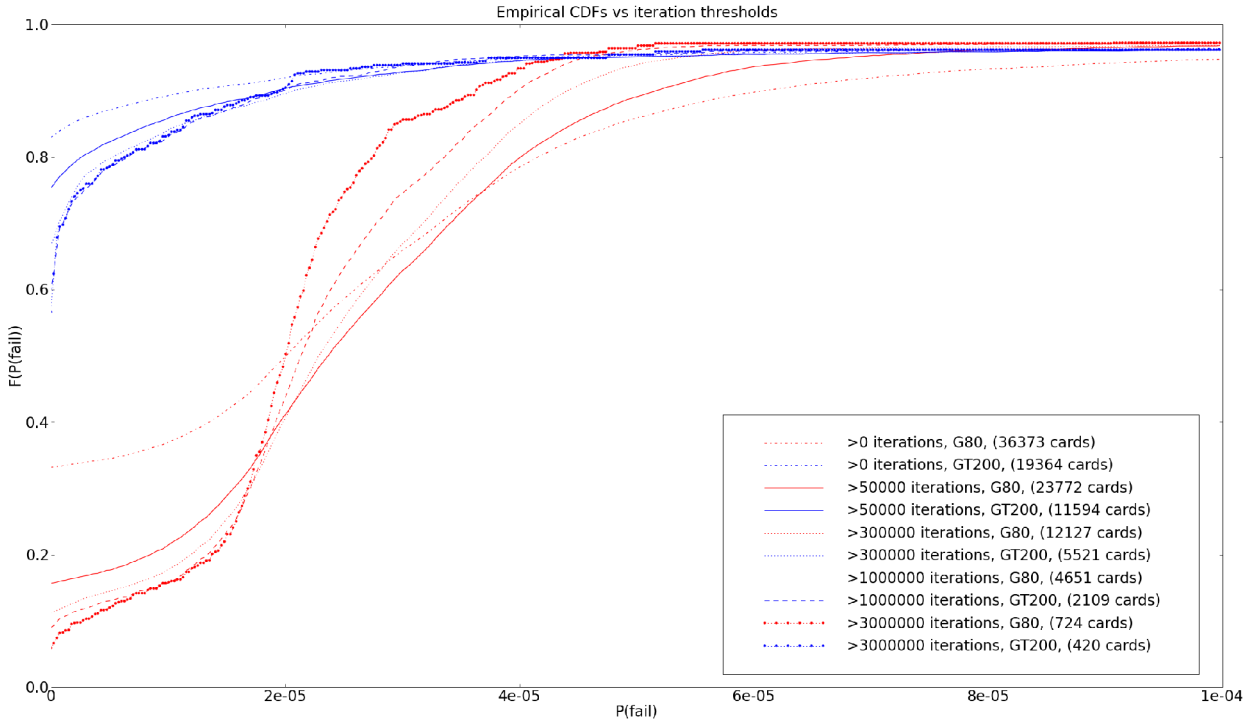
MemtestG80 – Results



- 2/3 of NVIDIA GPUs “in the wild” on Folding@home showed measurable rate of memory errors
- Mode of error distribution around probability = $2x10^{-5}$ error/test iteration = ~1-2 error/week for an “average” board
- Additional modes @ 0, ~$2x10^{-6}$- **why?** Not overclocking or time of day (proxy for temperature).

Haque IS and Pande VS. Submitted. arXiv:0910.0505v1

MemtestG80 – Results



- Newer GT200 GPU has a much lower error generation rate
- GT200 generates fewer memory transactions on most sensitive test (13x fewer than G80) – lines up well with GT200 error generation rate (~10x lower than G80)
- *Possible* that both G80 and GT200 have an inherent nonzero probability of error per-transaction – both architectures have larger fraction of failing boards as you consider more test iterations
- Hopefully Fermi's ECC will fix this.

Haque IS and Pande VS. Submitted. arXiv:0910.0505v1

OpenMM and the future of user-level GPU libraries

YOU DON'T HAVE TO LIVE LIKE A (GPU) REFUGEE

Friedrichs MS et al. Accelerating Molecular Dynamics Simulation on Graphics Processing Units. *J. Comp. Chem.* 2009

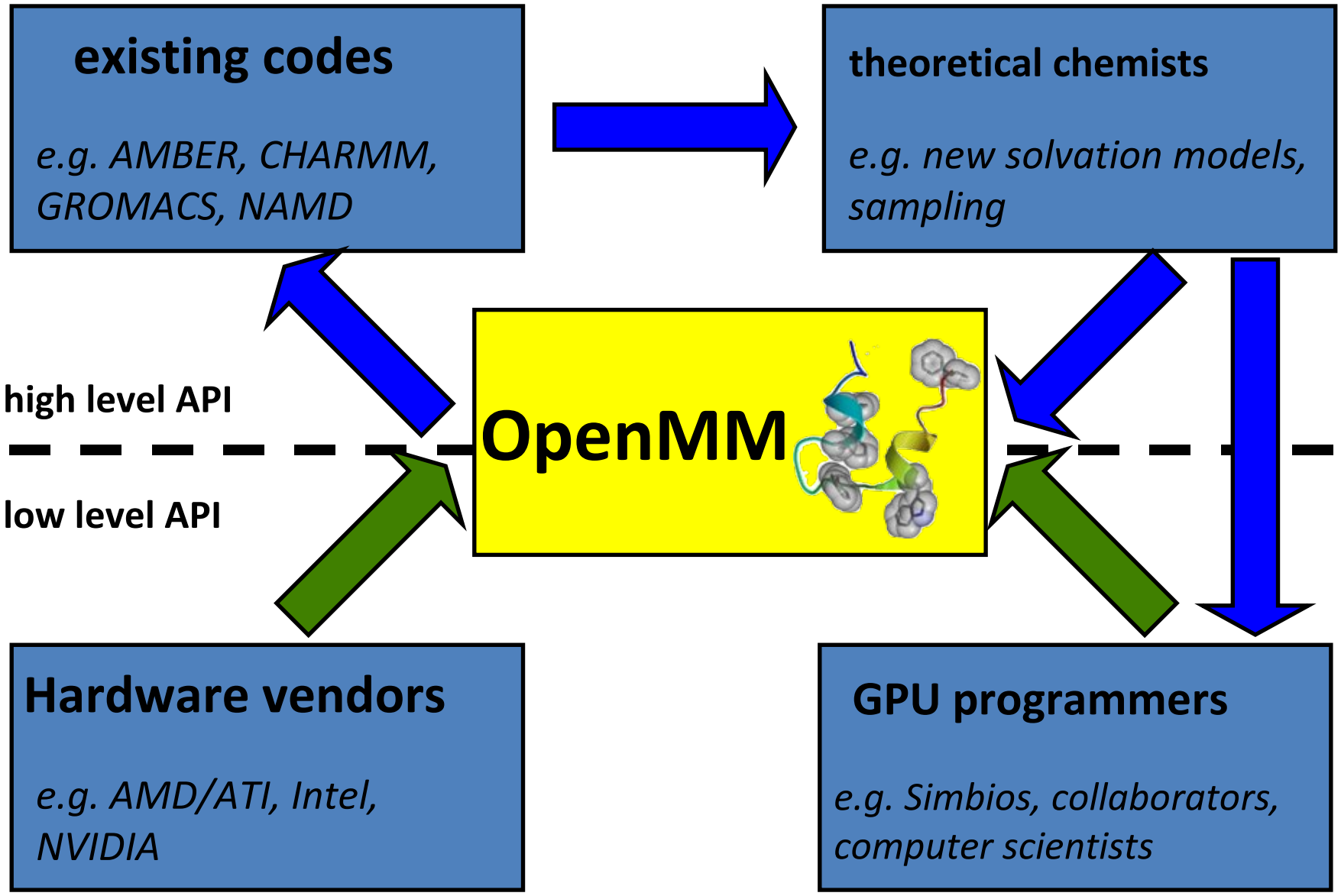
The OpenMM Opportunity

- MD community is fragmented – tens of codes with overlapping functionality and differing interfaces
- New advances (algorithms, **hardware acceleration**) must be ported individually to all these codes

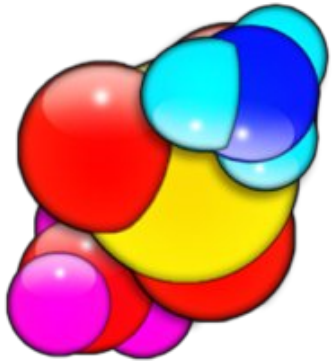
- We propose **OpenMM**, an extensible molecular mechanics API to unify MM like OpenGL for graphics
- Incorporates hardware acceleration in base design
- *Use this API as backend for existing MD packages*

https://simtk.org/home/openmm

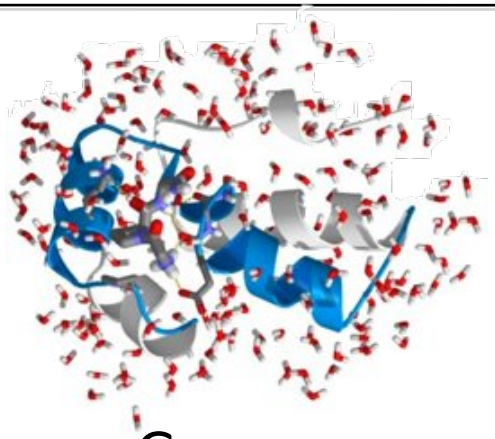
Connections to OpenMM



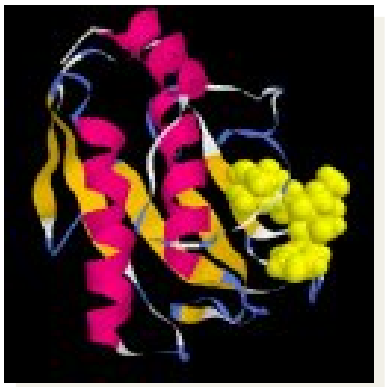
OpenMM-enabled Applications



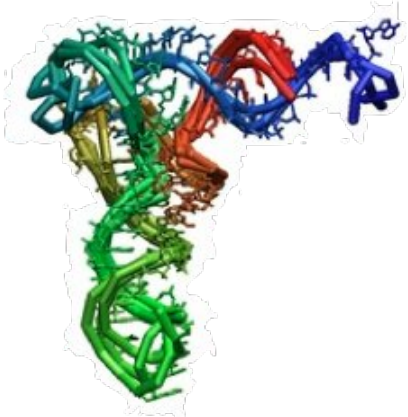
Folding@home
http://folding.stanford.edu



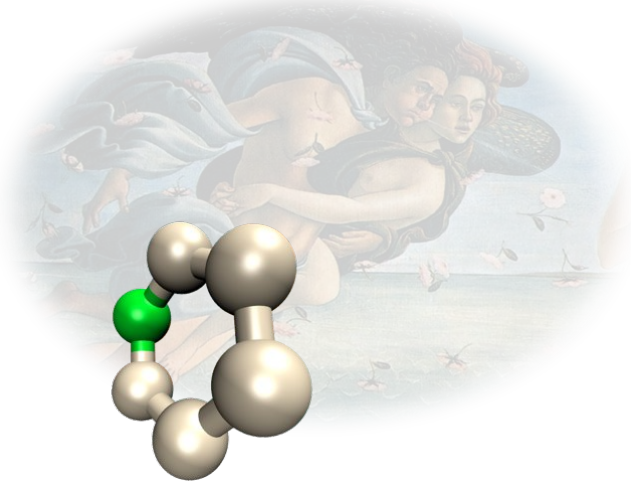
Gromacs
http://www.gromacs.org



Protomol
http://protomol.sourceforge.net



NAST
http://simtk.org/home/nast



Zephyr
http://simtk.org/home/zephyr

Yank
http://simtk.org/home/yank

OpenMM – Performance

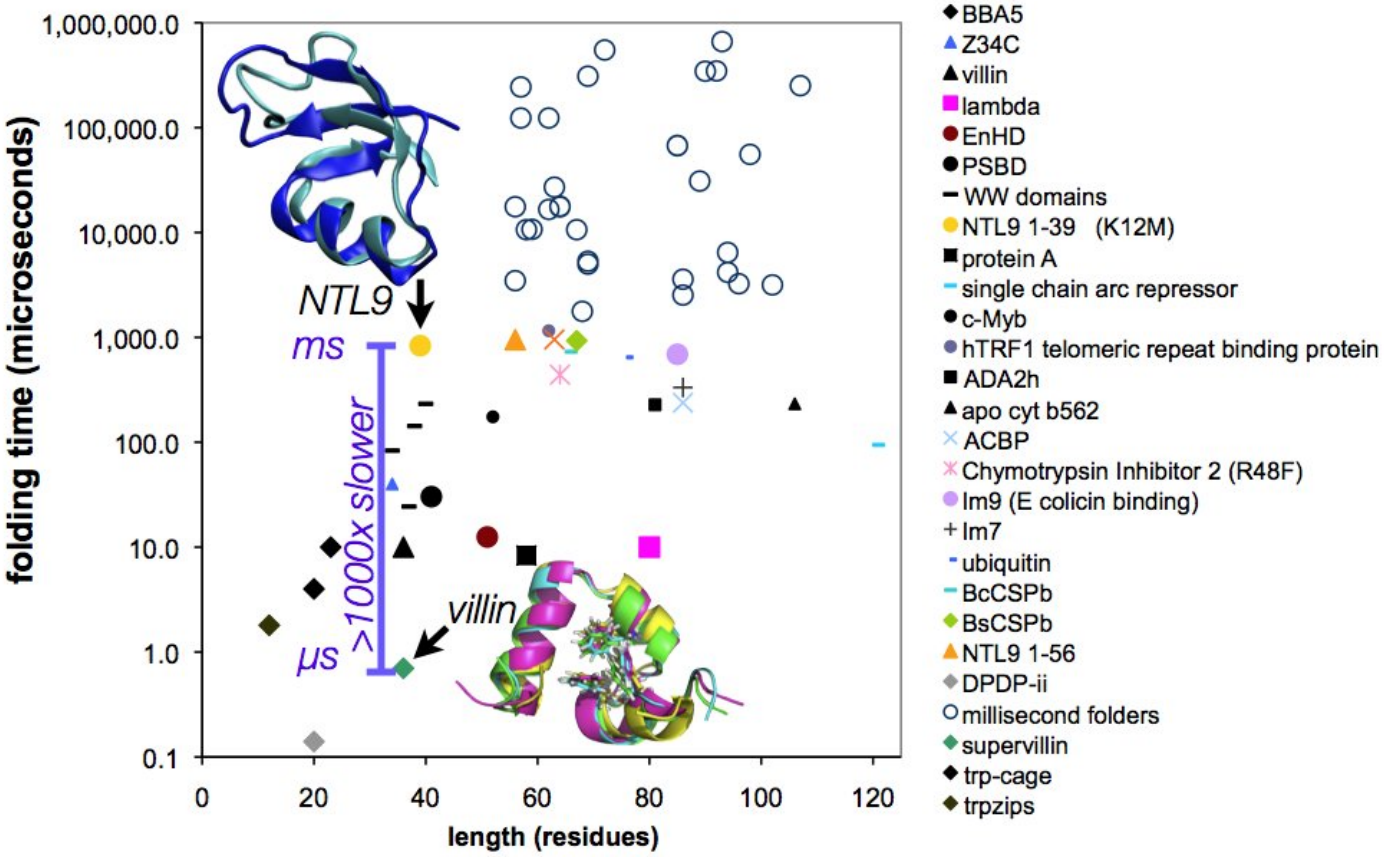
Molecule	# atoms	ns/day	Speedup*	GFLOPS (GPU native)	GFLOPS§ (x86-equiv)
fip35	544	576	**128x**	311	657
villin	582	529	**136x**	328	692
lambda	1254	202	**255x**	547	1153
α-spectrin	5078	17	**735x**	805	1702

(*) OpenMM on a GTX 280 vs. AMBER on one core of a 3GHz Core 2 Duo

(§) GPUs evaluate some transcendentals more efficiently than x86, so equivalent FLOP counts are included for each architecture

https://simtk.org/home/openmm

OpenMM – Performance



Using OpenMM on the GPU, we have folded NTL9, the slowest-folding protein yet computationally folded – a 1000x harder problem than folding villin! (Vince Voelz)

From Sprinter to Wide Receiver

How can we turn a *sprinter* – a high-performance, but inflexible scientific code – into a *wide receiver* – a code that does more than just run fast in a straight line?

https://simtk.org/home/openmm

AP Photo/Thomas Kienzle

OpenMM Lepton

- A domain-specific language for MD – optimizations are easier; no “magic compiler” needed
- Describe code in equations
  - Very flexible, custom nonbonded code
  - Ease of coding: automatic derivative evaluation, etc.

Subclass CustomFunction, implement:

```
int getNumArguments()
double evaluate(const double* arguments)
double evaluateDerivative(const double* arguments,const int* derivOrder) const
CustomFunction* clone() const
```

Provide custom functions when parsing:

```
map<string,CustomFunction*> functions;
functions[“foo”] = new MyCustomFunction();
ParsedExpression exp = Parser::parse(“10*foo(x/2)”,functions);
```

https://simtk.org/home/openmm

Example: PyMD

- **Interface to Python**
  - 9 lines of code to customizable, **high-performance** MD

```
import FF
import Simulation

FField = FF.ForceField.LoadFromHDF("./Amber99.h5")
Conf = FF.Conformation.LoadFromPDB("Test","./state0.pdb")
Topo = FF.Topology.CreateTopologyFromConformation(Amber99,Conf)
Sim = Simulation.Simulation.CreateSimulation(FField, Topo, Conf,
      Temp=300., Friction=1.0, TimeStep=0.002, GBSA=True, BondConstr=True)

Sim.Step(50000)
Conf["XYZ"]=Sim.GetXYZ()
Conf.SaveToPDB("Traj2.pdb")
```

https://simtk.org/home/openmm

Acknowledgments

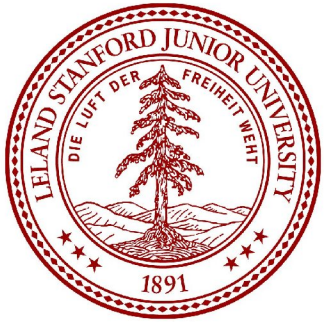
Stanford

- Vijay Pande (PI)
- Kyle Beauchamp
- Vince Voelz
- Christopher Bruns
- Peter Eastman
- Mark Friedrichs
- Kai Kohlhoff
- Michael Sherman

Collaborators

- Pat Walters
- Kim Branson
- John Chodera
- Erik Lindahl
- Michael Houston
- Scott LeGrand

- **Folding@home users**



Conclusions

- Need to choose applications that are parallel.
- Redesign algorithms both for parallelism and access – helps CPUs too!
- Trust, but verify, hardware.
- High-level libraries are the way to go.
- Questions? **ihaque@cs.stanford.edu**

Package	URL (Software/Publication)
PAPER (3-D chemical similarity)	**https://simtk.org/home/paper** http://dx.doi.org/10.1002/jcc.21307
MemtestG80 (GPU hardware test)	**https://simtk.org/home/memtest** http://arxiv.org/abs/0910.0505
OpenMM (Molecular mechanics)	**https://simtk.org/home/openmm** http://dx.doi.org/10.1002/jcc.21209
gpuLINGO (1-D chemical similarity)	*OpenCL port on its way – stay tuned!*