

# Bioinformatics

## Modeling of biological systems

Adrian E. Roitberg

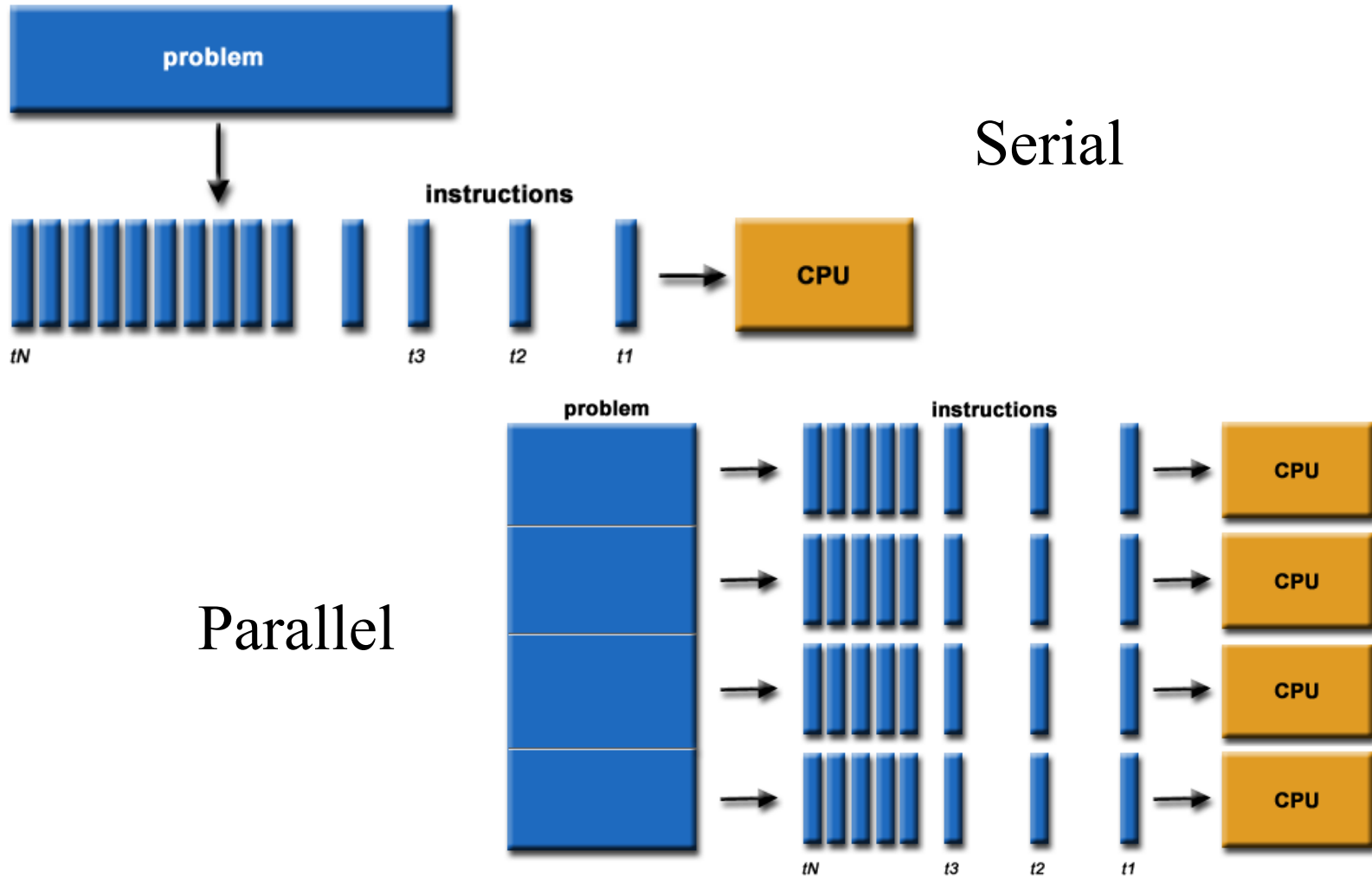
University of Florida

ECAR2012 -  
Escuela de Computacion de Alto Rendimiento

# Why run in parallel or on a GPU?

- Sampling.
  - We can get more simulation done in a given amount of wall clock time.
  - Wall clock time defines how much science you can do before you die.
- Each new generation of processor is getting slower.

# Serial vs Parallel

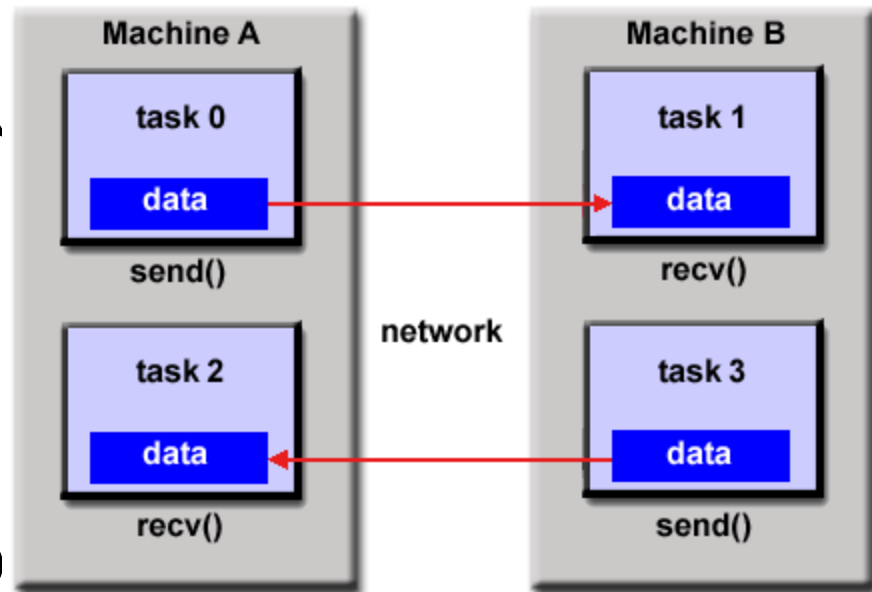


# The Problem

- Each MD step requires knowledge of the previous step.
  - We start step  $N+1$  until we have completed  $N$ ,  $N-1$ ,  $N-2$  etc etc.
- This means we need a fast way to communicate between processors.
  - Bandwidth and latency of the interconnect limits scaling.
  - Performance CAN be improved by running in parallel, however.

# AMBER in Parallel

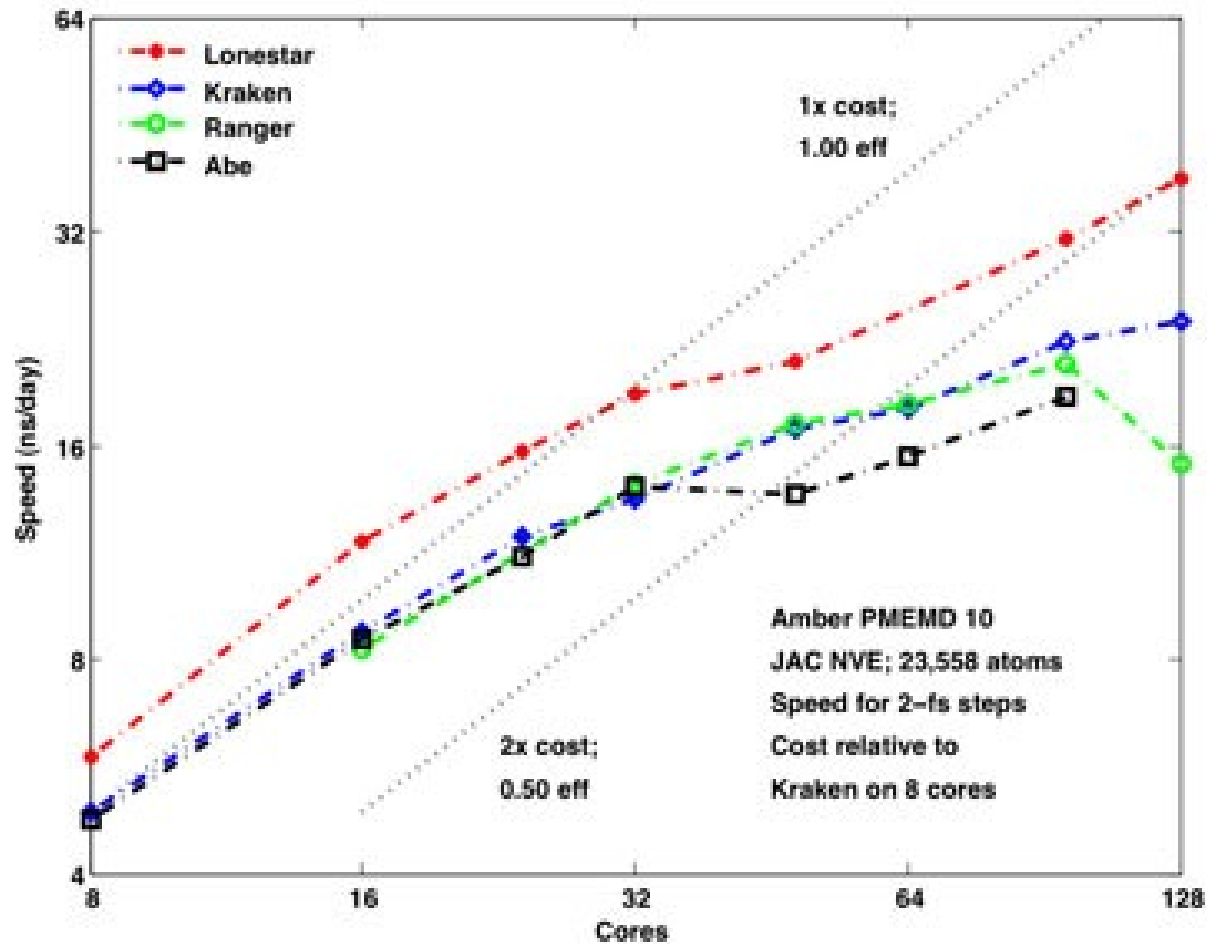
- Uses the Message Passing Interface (MPI)
- Parallel codes run separately.
- Data is sent between nodes on each MD step.



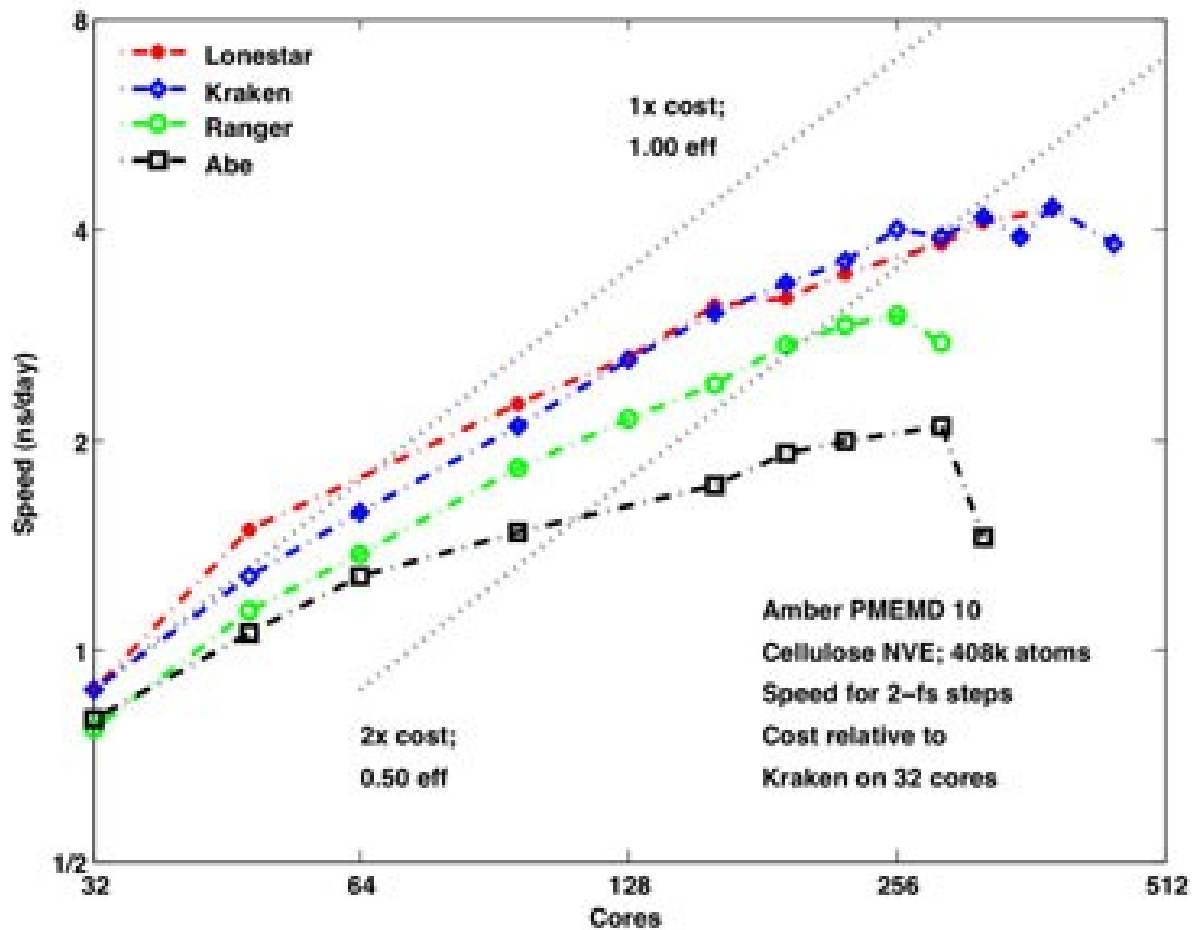
# Always test performance

- Using more CPUs is not always faster.
- Different problem sizes scale differently.
  - GB scales better than PME
  - Larger simulations can typically be run on more CPUs.
  - Writing to mdcrd, mdout or restrt file too often can hurt performance.
  - PMEMD provides better performance than sander
    - If it supports your simulation settings you should use it.

# JAC (23K atoms)



# Cellulose (408K atoms)





# GPU Support

- Collaboration with NVIDIA to produce CUDA version of AMBER.
  - PMEMD Engine
  - Implicit Solvent GB
  - Explicit Solvent PME
- Provides a way to make a desktop very fast.
- More info here:
- [ambermd.org/gpus](http://ambermd.org/gpus)



# Supercomputers in a Desktop



4.44TFlop Peak

ASCII Red (LANL), 3.2TFlop,  
#1 Top 500, June 2000.



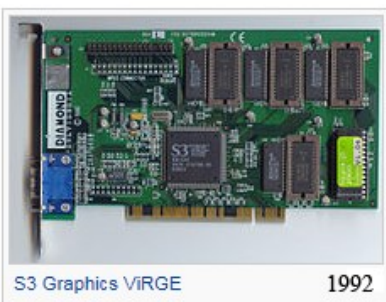
# What is a GPU?

- Graphics Processing Unit

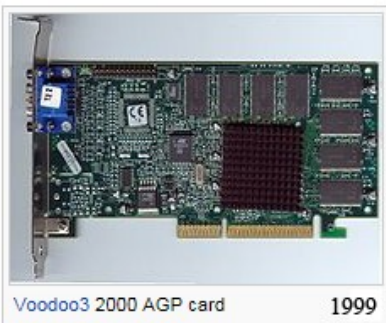
- ‘Specialist’ processor for accelerating the rendering of computer graphics.
- Invented by 3DFX (Later bought by NVIDIA) in 1997.
- Originally fixed function pipelines
  - Invention of OpenGL added programmability.
  - Pixels can be programmed with specific textures.
  - Onboard memory for storing textures.
- Development driven by \$150 billion gaming industry.



1991



1992



1999



2004



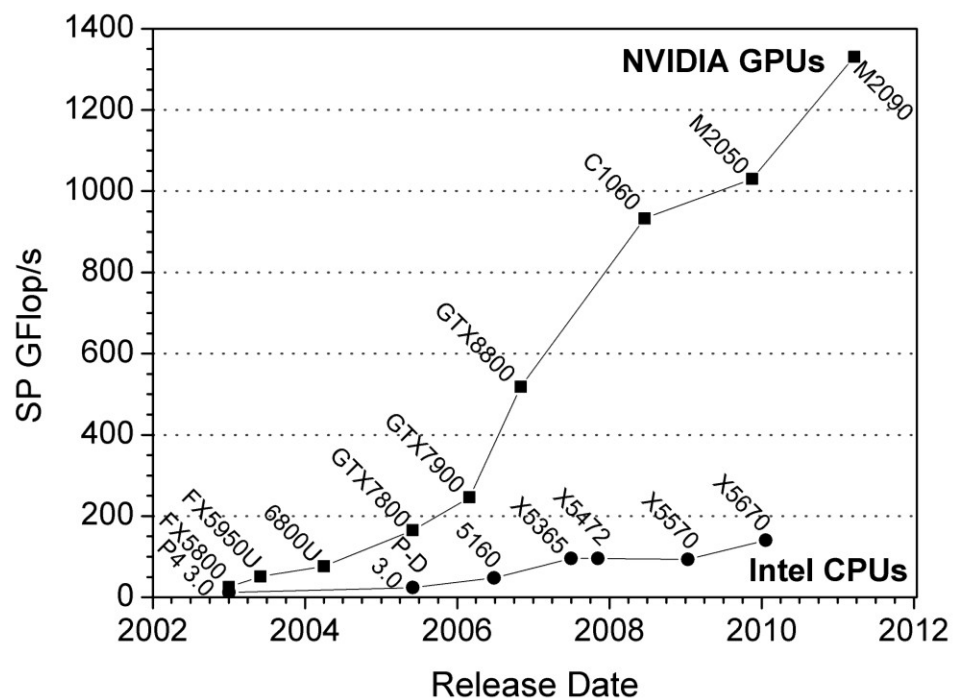
2008

# A Brief History of GPU Computing

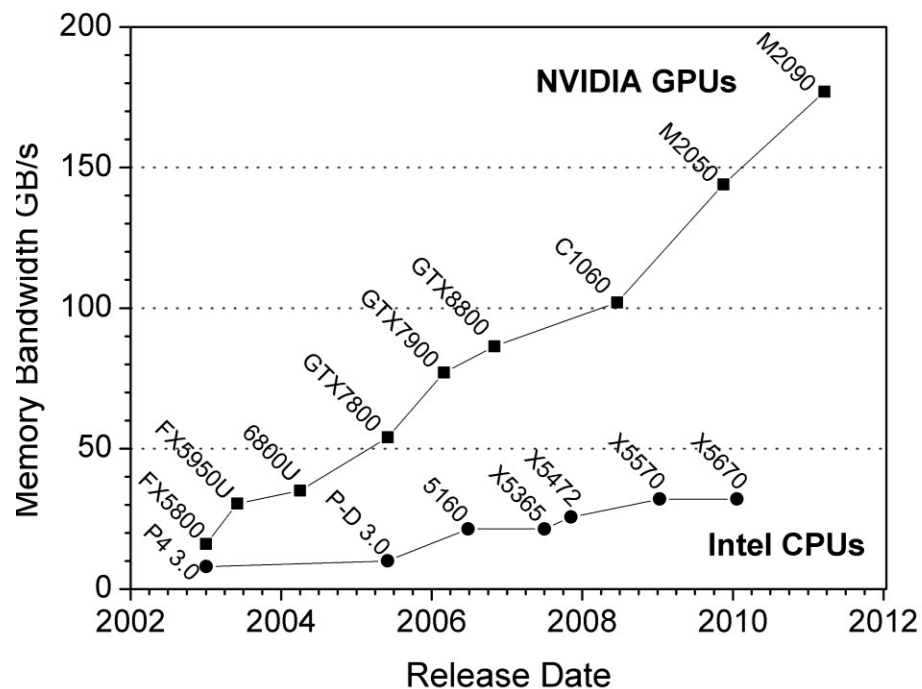
- 2003 - First attempts to use GPUs for general computing.
  - Programmed as graphics primitives (heroic)
  - problems had to be expressed in terms of vertex coordinates, textures and shader programs.
  - Hardware lacking certain ‘features’ – No random reads or writes etc.
- 2004 – ‘Brook’ programming language for GPUs.
- 2007 – NVIDIA announce CUDA at SC07
  - Release GPUs with specific ‘computational’ features.
- 2008 – OpenCL language ratified.
  - Mainly aimed at embedded devices but has features for GPU computation.

# Why Interest GPUs?

## Peak Flops



## Memory Bandwidth

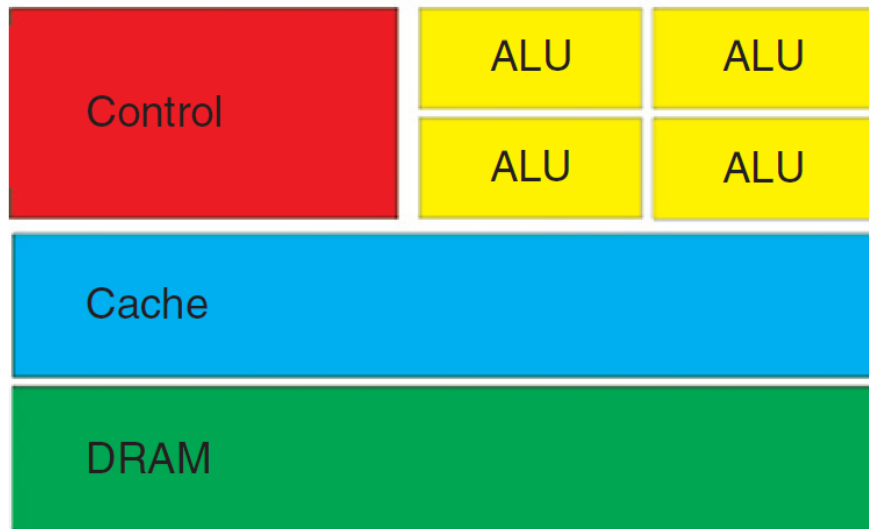


What's the Catch?

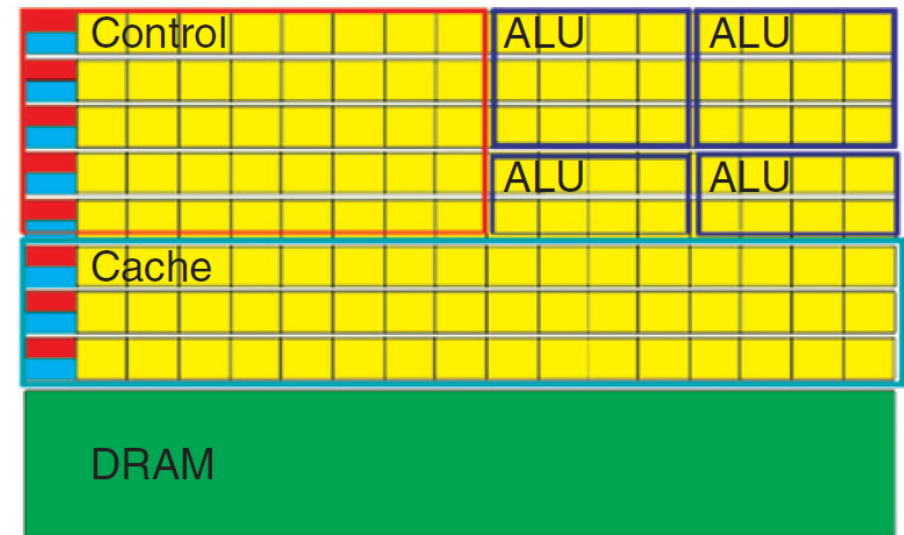


# GPUs Require Rewriting Lots of Code (Is it worth it?)

(a) CPU



(b) GPU



***Spoiler: Yes. But LOTS of work.***

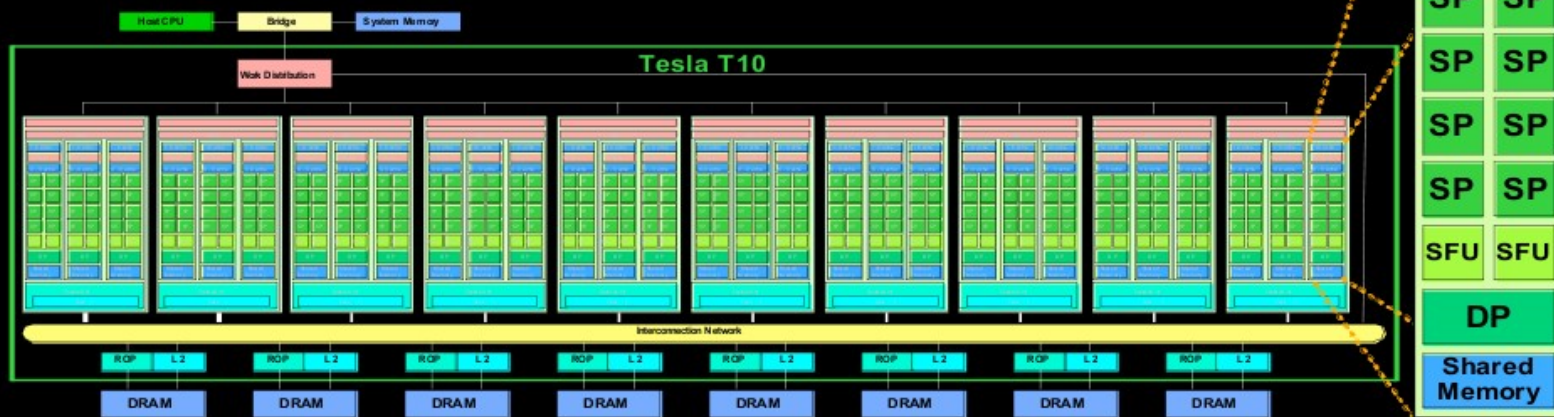
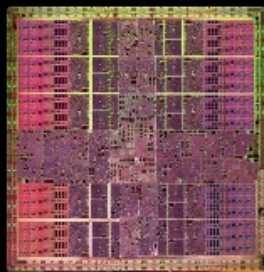
# CUDA Computing with Tesla T10



- 240 SP processors at 1.45 GHz: 1 TFLOPS peak
- 30 DP processors at 1.44GHz: 86 GFLOPS peak
- 128 threads per processor: 30,720 threads total

30 multiprocessors

$$1.45 * 3 \text{flops/cycle} (\text{FMAD} + \text{FMUL}) * 240 \text{ cores}$$



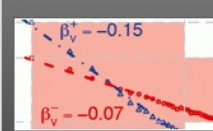


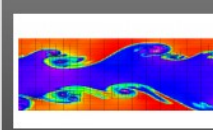
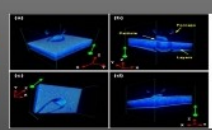

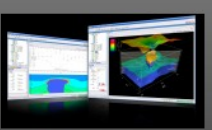
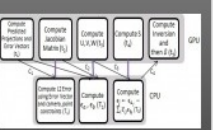
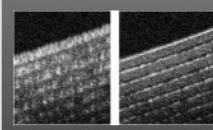






# What GPU Codes Exist?

- Excellent list on:
- <http://www.nvidia.com/cuda/>
- Examples available form almost all fields.

- Computational Chemistry
- Life Sciences
- Astrophysics
- Finance
- Medicine / Medical Imaging
- Natural Language Processing
- Social Sciences

Showing 1 - 15 of 1271

 <p>GPU-computing in econophysics and statistical phys...</p>	 <p>Multicore/Multi-GPU Accelerated Simulations of MUL...</p>	 <p>CUDA Image Mosaic</p>	 <p>Processing and rendering of Fourier domain optical...</p>	 <p>Running the High Performance Linpack (HPL) Benchma...</p>
 <p>Directionally Unsplit Hydrodynamic Schemes with Hy...</p>	 <p>Real-time intraoperative 4D full-range FD-OCT base...</p>	 <p>GPU Vision: Accelerating Computer Vision algorithm...</p>	 <p>IGMAS+</p>	 <p>Practical Time Bundle Adjustment for 3D Reconstruct...</p>
 <p>Real-time numerical dispersion compensation using ...</p>	 <p>Horizon MHD</p>	 <p>CUDA</p>	 <p>Textbook: Programming Massively Parallel Processor...</p>	 <p>A demonstration of Exact String Matching Algorithm...</p>

# Towards routine microsecond molecular dynamics simulations of proteins on commodity hardware: Extreme GPU acceleration of AMBER



# What is AMBER?

## An MD simulation package

12 Versions as of 2012

distributed in two parts:

- *AmberTools*, preparatory and analysis programs, free under GPL

- *Amber* the main simulation programs, under academic licensing

independent from the accompanying forcefields

## A set of MD forcefields

fixed charge, biomolecular forcefields: ff94, ff99, ff99SB, ff03, ff11

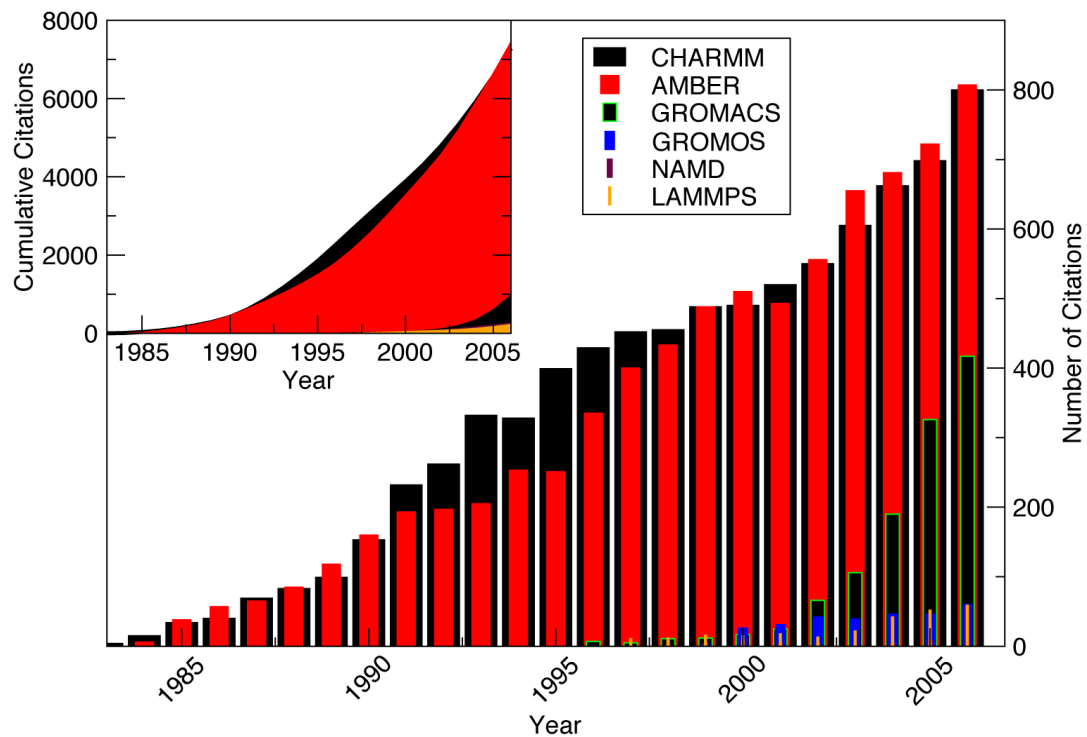
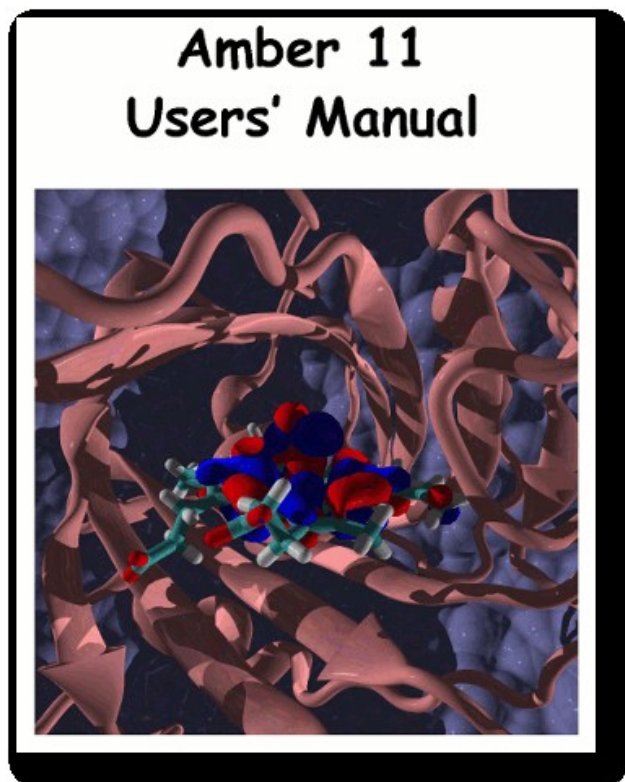
experimental polarizable forcefields e.g. ff02EP

parameters for general organic molecules, solvents, carbohydrates (Glycam), etc.

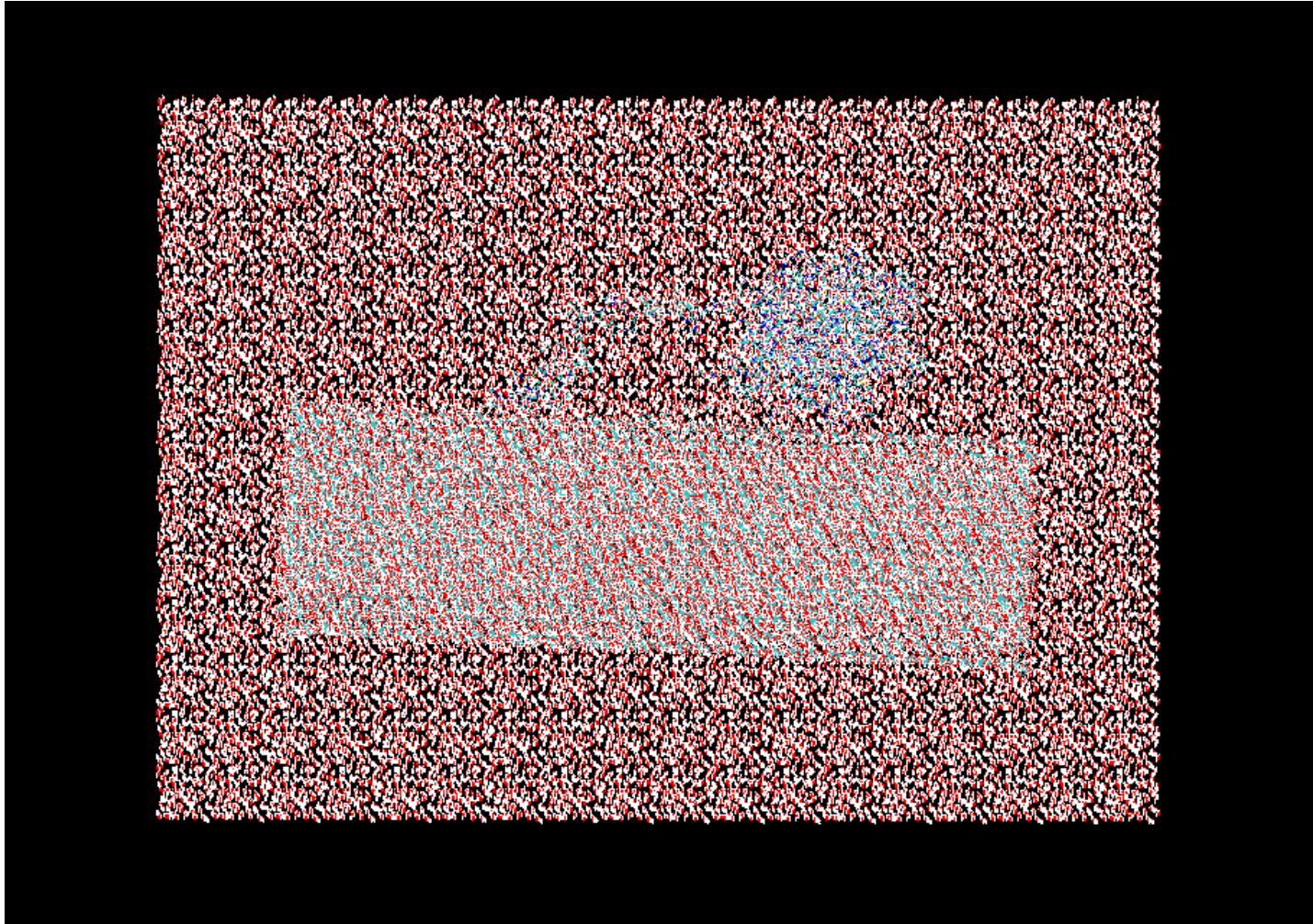
in the public domain

# AMBER Usage

- Approximately 850 site licenses (per version) across most major countries.



# Why do we need Supercomputers? Lots of Atoms





# Why do we need Supercomputers? Lots of Time Steps

- *Maximum time per step is limited by fastest motion in system (vibration of bonds)*  
*= 2 femto seconds (0.0000000000000002 seconds)*  
*(Light travels 0.006mm in 2 fs)*
- *Biological activity occurs on the nano-second to micro-second timescale.*  
*1 micro second = 0.000001 seconds*

**SO WE NEED**

**500 million steps to reach 1 microsecond!!!**

# The Problem(s)

- Molecular Dynamics is inherently serial.
  - To compute  $t+1$  we must have computed all previous steps.
  - We cannot simply make the system bigger since these need more sampling (although many people conveniently forget this).

# GPU Support in AMBER



## Routine Microsecond Molecular Dynamics Simulations with AMBER on GPUs. 1. Generalized Born

Andreas W. Götz,<sup>†</sup> Mark J. Williamson,<sup>†,||</sup> Dong Xu,<sup>†,||</sup> Duncan Poole,<sup>‡</sup> Scott Le Grand,<sup>‡</sup> and Ross C. Walker<sup>\*,1§</sup><sup>†</sup>San Diego Supercomputer Center, University of California San Diego, 9500 Gilman Drive MC0505, La Jolla, California 92093, United States<sup>‡</sup>NVIDIA Corporation, 2701 San Tomas Expressway, Santa Clara, California 95050, United States<sup>§</sup>Department of Chemistry and Biochemistry, University of California San Diego, 9500 Gilman Drive MC0505, La Jolla, California 92093, United States

## Supporting Information

**ABSTRACT:** We present an implementation of generalized Born implicit solvent all-atom classical molecular dynamics (MD) within the AMBER program package that runs entirely on CUDA enabled NVIDIA graphics processing units (GPUs). We discuss the algorithms that are used to exploit the processing power of the GPUs and show the performance that can be achieved in comparison to simulations on conventional CPU clusters. The implementation supports three different precision models in which the contributions to the forces are calculated in single precision floating point arithmetic but accumulated in double precision (SPDP), or everything is computed in single precision (SPSP) or double precision (DPPD). In addition to performance, we have focused on understanding the implications of the different precision models on the outcome of implicit solvent MD simulations. We show results for a range of tests including the accuracy of single point force evaluations and energy conservation as well as structural properties pertaining to protein dynamics. The numerical noise due to rounding errors within the SPSP precision model is sufficiently large to lead to an accumulation of errors which can result in unphysical trajectories for long time scale simulations. We recommend the use of the mixed-precision SPDP model since the numerical results obtained are comparable with those of the full double precision DPPD model and the reference double precision CPU implementation but at significantly reduced computational cost. Our implementation provides performance for GB simulations on a single desktop that is on par with, and in some cases exceeds, that of traditional supercomputers.

## 1. INTRODUCTION

Since the first simulation of an enzyme using molecular dynamics (MD) was reported by McCammon et al.<sup>1</sup> in 1977, MD simulations have evolved to become important tools in rationalizing the behavior of biomolecules. The field has grown from that first 10-ps-long simulation of a mere 500 atoms to the point where small enzymes can be simulated on the microsecond time scale<sup>2–4</sup> and simulations containing millions of atoms can be considered routine.<sup>5,6</sup> However, such simulations are numerically very intensive and using traditional CPU-centric hardware requires access to large-scale supercomputers or well-designed clusters with expensive interconnects that are beyond the reach of many research groups.

Numerous attempts have been made over the years to accelerate classical MD simulations by exploiting alternative hardware technologies. Some notable examples include ATOMS by AT&T Bell Laboratories,<sup>7</sup> FASTRUN by Columbia University and Brookhaven National Laboratory,<sup>8</sup> MIDGRAPE by RIKEN,<sup>9</sup> and most recently Anton by DE Shaw Research LLC.<sup>10</sup> All of these approaches have, however, failed to make an impact on mainstream research because of their excessive cost. Additionally, these technologies have been based on custom hardware and do not form part of what would be considered a standard workstation specification. This has made it difficult to experiment with such technologies, leading to a lack of sustained development or

innovation and ultimately their failure to mature into ubiquitous community-maintained research tools.

Graphics processing units (GPUs), on the other hand, have been an integral part of personal computers for decades, and a strong demand from the consumer electronics industry has resulted in significant sustained industrial investment in the stable, long-term development of GPU technology. In addition to low prices for GPUs, this has led to a continuous increase in the computational power and memory bandwidth of GPUs, significantly outstripping the improvements in CPUs. As a consequence, high-end GPUs can be considered standard equipment in scientific workstations, which means that they either already exist in many research laboratories or can be purchased easily with new equipment. This makes them readily available to researchers and thus attractive targets for acceleration of many scientific applications including MD simulations.

The nature of GPU hardware, however, has until recently made their use in general purpose computing challenging to all but those with extensive three-dimensional (3D) graphics programming experience. However, the development of application programming interfaces (APIs) targeted at general purpose scientific computing has reduced this complexity substantially such that

Received: December 20, 2011

Published: March 26, 2012

Goetz, A. W.; Williamson, M. J.; Xu, D.; Poole, D.; Grand, S.; Walker, R. C. **"Routine microsecond molecular dynamics simulations with amber - part i: Generalized born"**, *Journal of Chemical Theory and Computation.*, 2012, 8 (5), pp 1542–1555, DOI:10.1021/ct200909j

# Supported Features

- Supports ‘standard’ MD
    - Explicit Solvent (PME)
      - NVE/NVT/NPT
    - Implicit Solvent (Generalized Born)
  - Thermostats
    - Berendsen, Langevin, Anderson
  - Restraints / Constraints
    - Standard harmonic restraints
    - Shake on hydrogen atoms
- New in AMBER 12**

  - Umbrella Sampling
  - REMD
  - Simulated Annealing
  - Accelerated MD
  - Isotropic Periodic Sum
  - Extra Points

# Design Goals

- Transparent to the user.
  - Easy to compile / install.
  - AMBER Input, AMBER Output
  - Simply requires a change in executable name.
- Cost effective performance.
  - C2050 should be equivalent to 4 or 6 standard IB nodes.
- Focus on accuracy.
  - Should NOT make any additional approximations.
  - Accuracy should be directly comparable to the standard CPU implementation.

# Precision Models

- Multiple codes have simply used single precision without any ‘REAL’ consideration of accuracy implications.
  - Validation is now the ‘worst’ part of programming.
- We have focused on accuracy first.
  - Get the answers correct and validate!
  - Then improve performance.
- We have implemented several precision models for testing.

# When Does Single Precision Fail?

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.

# Precision Models

*SPSP* - Use single precision for the entire calculation with the exception of SHAKE which is always done in double precision.

*SPDP* - Use a combination of single precision for calculation and double precision for accumulation. (*Default*)

*DPDP* - Use double precision for the entire calculation

# Force Accuracy

Table 5: Deviations of forces (in kcal/(molÅ)) of the AMBER PMEMD GPU implementation using different precision models as compared to reference values obtained with the CPU implementation.

Precision model	TRPCage (304 atoms)	ubiquitin (1,231 atoms)	apo-myoglobin (2,492 atoms)	nucleosome (25,095 atoms)
<i>max deviation</i>				
SPSP	$3.0 \times 10^{-3}$	$4.8 \times 10^{-3}$	$4.2 \times 10^{-3}$	$2.7 \times 10^{-2}$
SPDP	$5.6 \times 10^{-5}$	$3.7 \times 10^{-4}$	$1.6 \times 10^{-4}$	$1.1 \times 10^{-3}$
DPDP	$1.1 \times 10^{-8}$	$7.3 \times 10^{-8}$	$3.4 \times 10^{-8}$	$8.0 \times 10^{-8}$
<i>RMS deviation</i>				
SPSP	$5.0 \times 10^{-4}$	$6.1 \times 10^{-4}$	$4.1 \times 10^{-4}$	$1.5 \times 10^{-3}$
SPDP	$7.0 \times 10^{-6}$	$1.5 \times 10^{-5}$	$8.1 \times 10^{-6}$	$3.0 \times 10^{-5}$
DPDP	$1.5 \times 10^{-9}$	$3.6 \times 10^{-9}$	$2.6 \times 10^{-9}$	$3.2 \times 10^{-9}$

# Energy Conservation: Implicit Solvent (kT/ns/d.o.f)

UBIQUITIN GB	dt = 0.5fs	dt = 1.0fs
CPU	-0.000008	-0.000835
DPDP	-0.000001	-0.000780
SPDP	-0.000008	-0.000631
SPSP	0.000589	0.001139
OpenMM	---	0.005411

0.000008 kT/ns/dof = 0.01798 kcal/mol/ns.



# Energy Conservation: Explicit Solvent

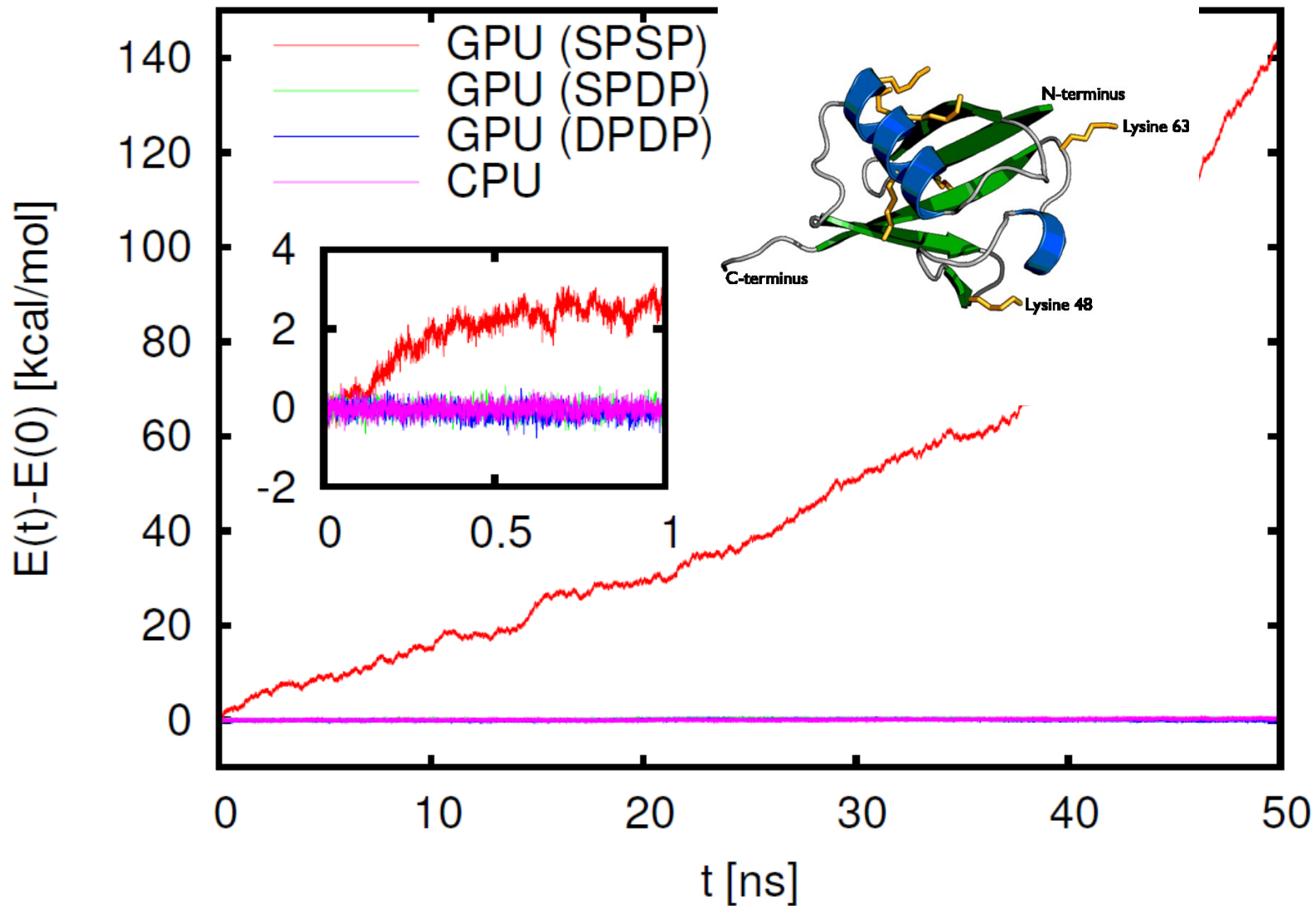
(kT/ns/d.o.f)

DHFR	dt = 0.5fs	dt = 1.0fs	dt = 2.0fs*
CPU	0.000000	0.000001	-0.000047
DPDP	0.000007	0.000024	-0.000101
SPDP	-0.000052	0.000050	-0.000066
SPSP	0.001969	0.001171	3.954495
Gromacs 4	---	0.011xxx	0.005xxx
Desmond	---	0.017xxx	0.001xxx
NAMD	---	0.023xxx	---

+ = J. Chem. Theory Comput., Vol. 4, No. 3, 2008

\* = shake on H bonds. 0.000008 kT/ns/dof = 0.01798 kcal/mol/ns.

# Energy Conservation



# Single Precision Issues?

- NVE does not work, but we can use a thermostat no?

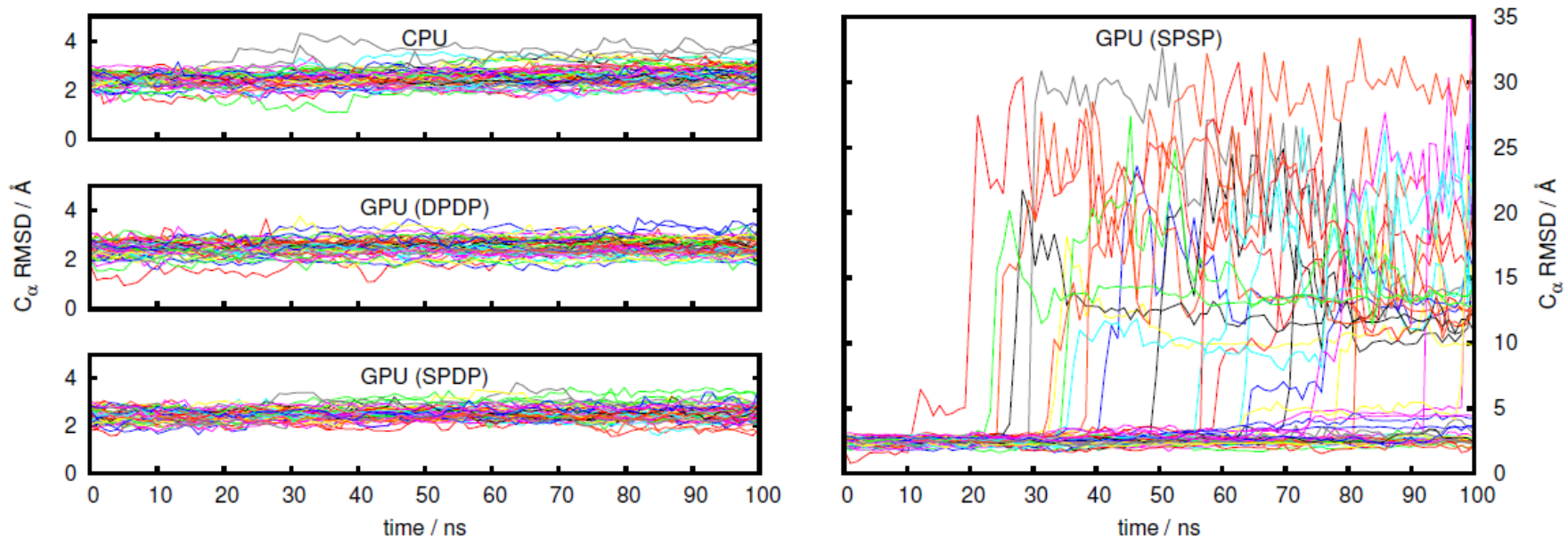


Figure 4: Root-mean-square deviations (RMSDs) of the C<sub>α</sub> backbone carbon atoms of ubiquitin (excluding the flexible tail, residues 71 to 76) with respect to the crystal structure for 50 independent trajectories as obtained with the CPU implementation and the GPU implementation of PMEMD using different precision models.

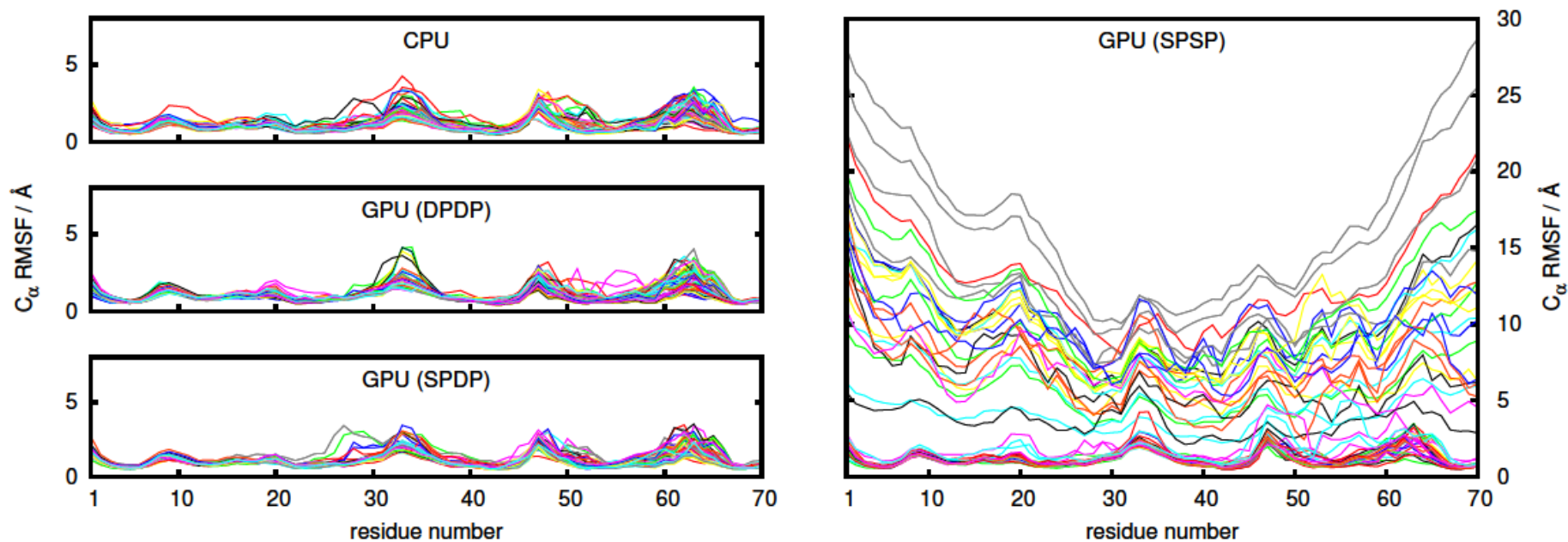


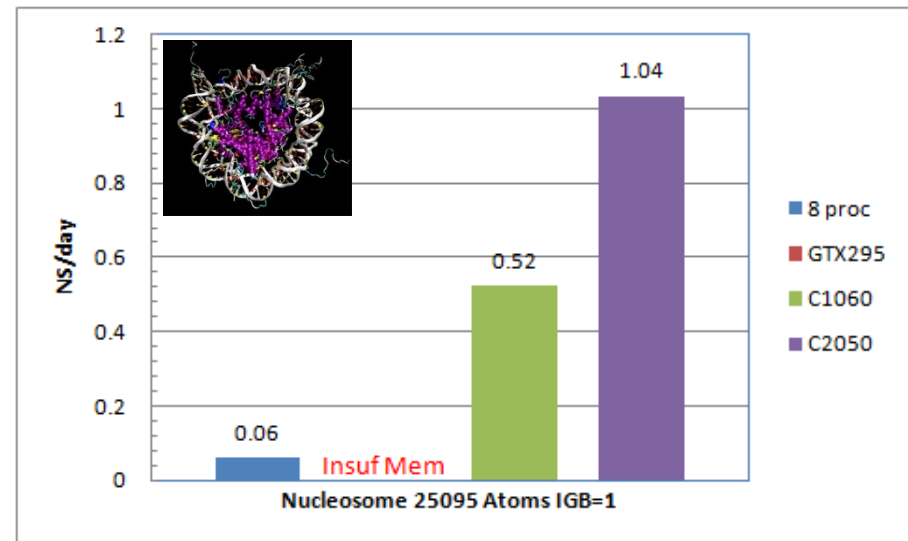
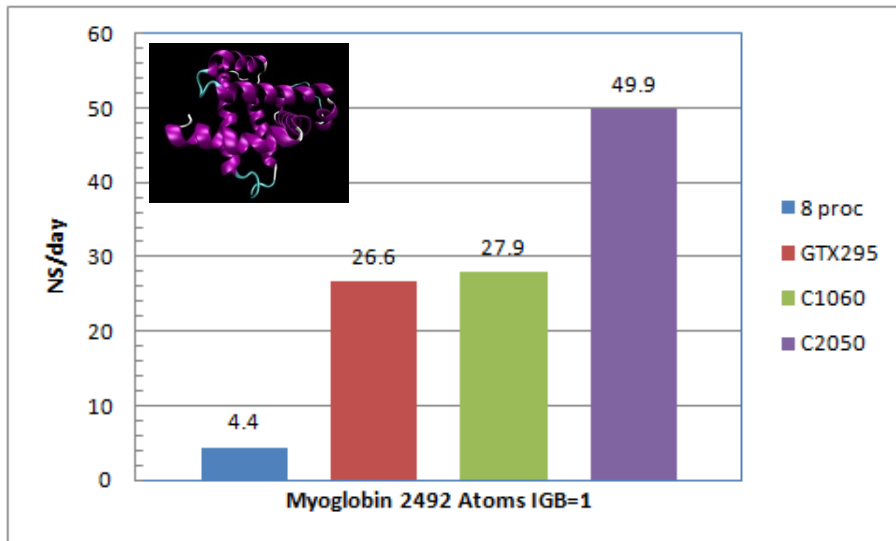
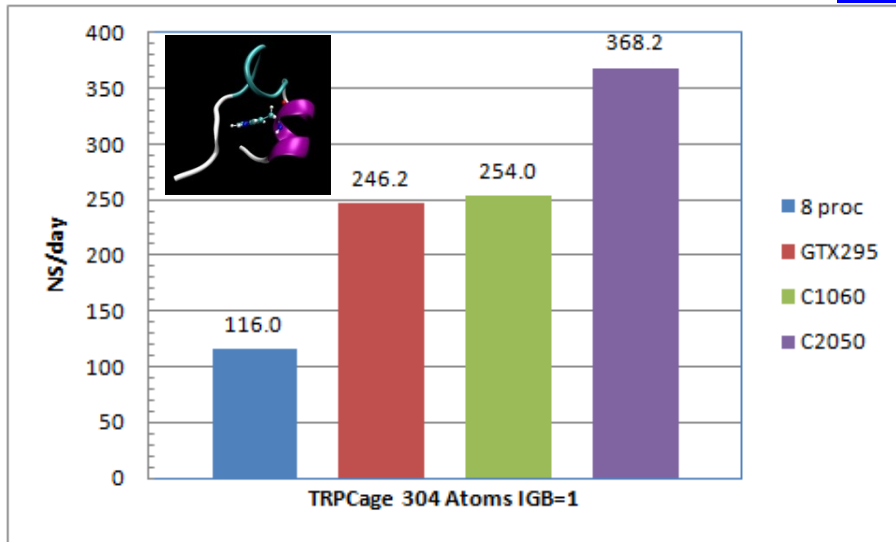
Figure 5: Root-mean-square fluctuations (RMSFs) of the  $C_{\alpha}$  backbone carbon atoms of ubiquitin residues 71 to 76 with respect to the crystal structure for 50 independent trajectories of 100 ns length as obtained with the CPU implementation and the GPU implementation of PMEMD using different precision models.

# Performance

AMBER 11

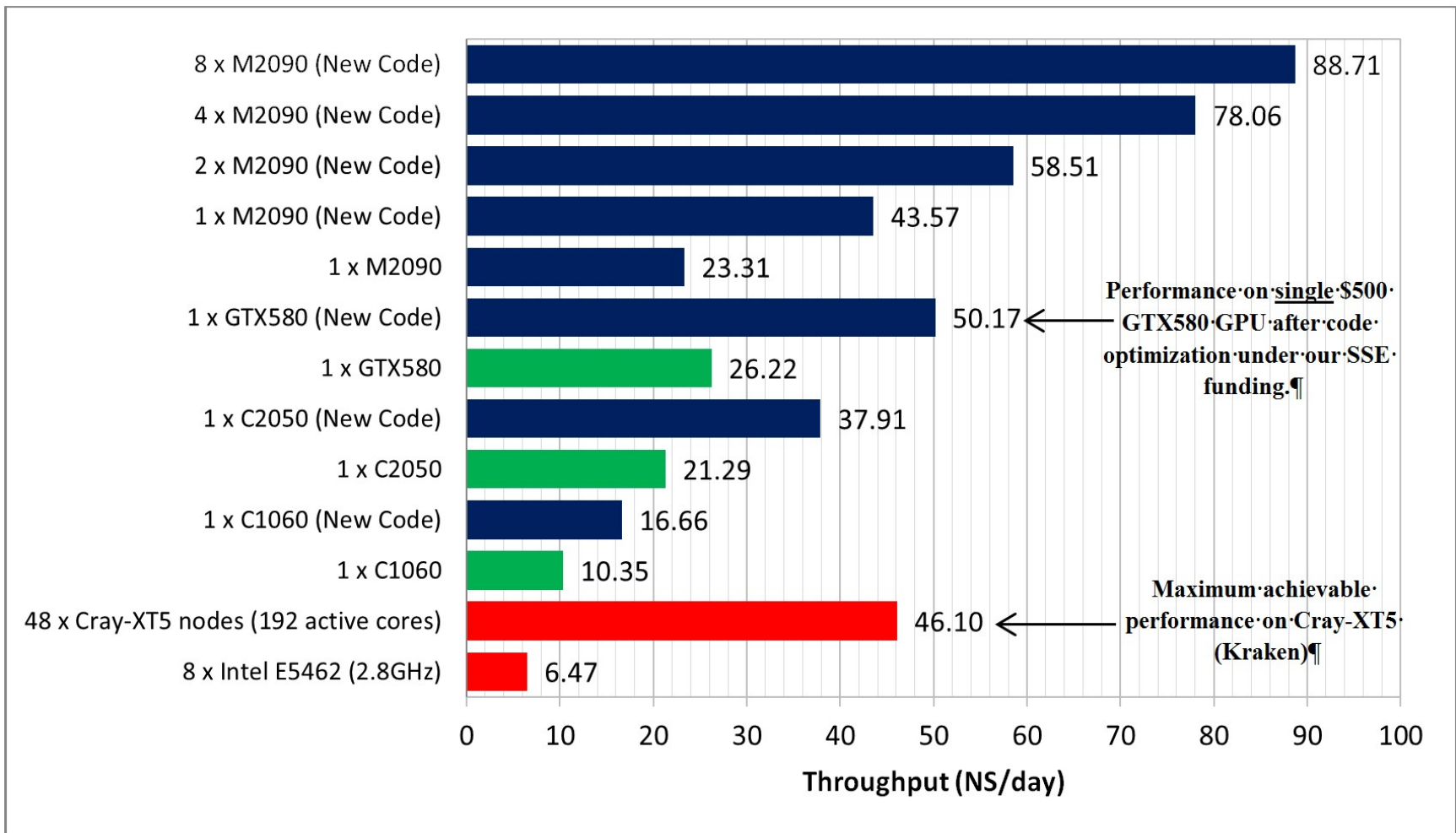
# Implicit Solvent Performance

As expected the performance differential is larger for bigger systems.



8 proc = Intel E5462 @ 2.8GHz

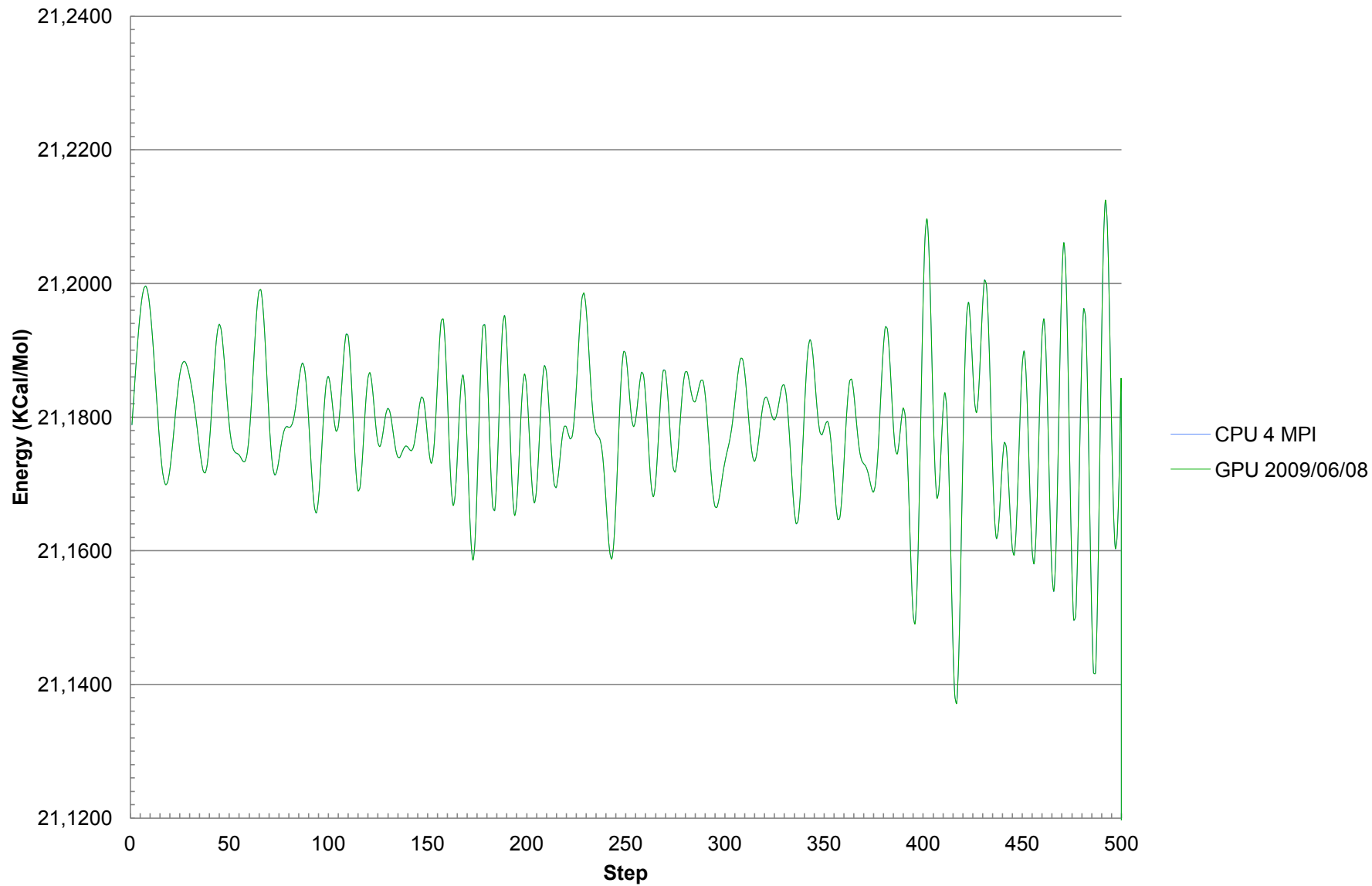
# Explicit Solvent Performance (JAC DHFR Production Benchmark)





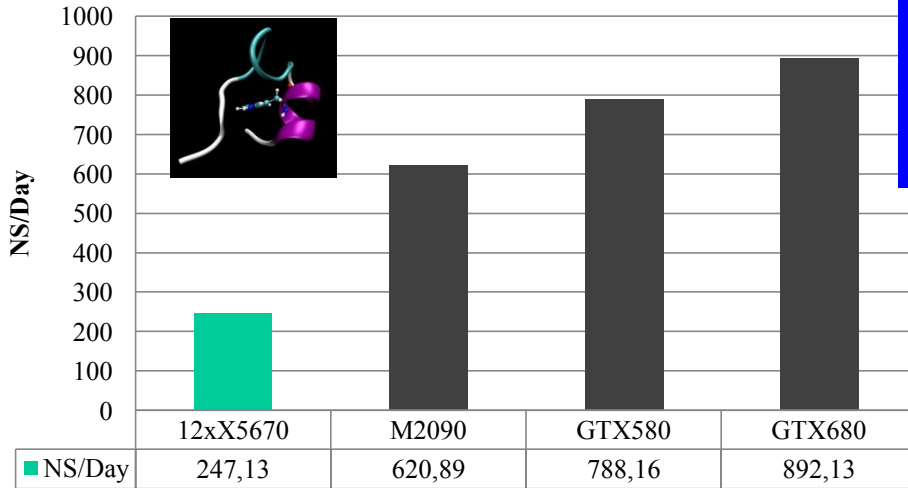


# IGB1 NVE No Shake Total E 0.5fs Time Step (NSCM = 1000) [DPDP]



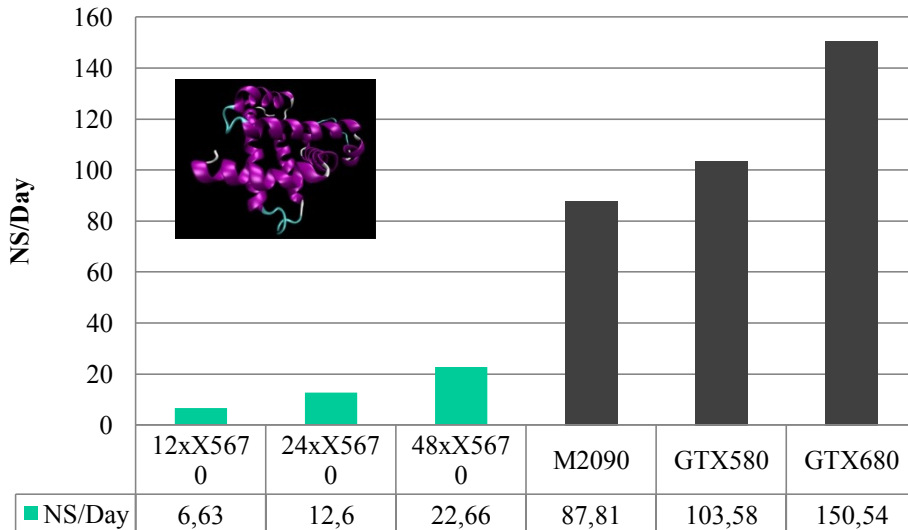
# Implicit Solvent Performance (SPFP)

TRPCage, 304 atoms, igb=1, 2fs, no cut

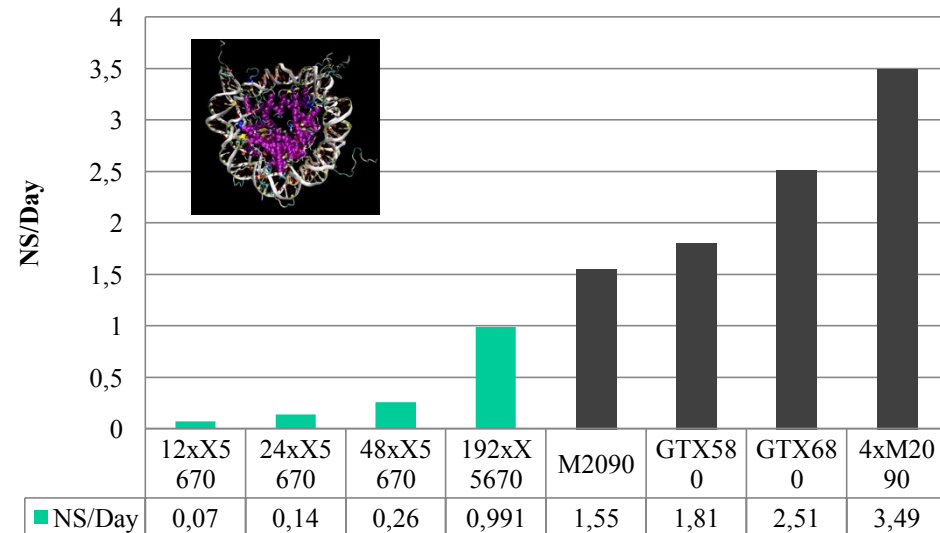


As expected the performance differential is larger for bigger systems.

Myoglobin, 2492 atoms, igb=1, 2fs, no cut



Nucleosome, 25095 atoms, igb=1, 2fs, no cut



# Explicit Solvent Performance (JAC DHFR Production Benchmark)

