# OpenMM workshop


# Welcome and Introductions

Vijay Pande

OpenMM Workshop


# Introductions

- **Stanford/Simbios team development**
  - Peter Eastman (OpenMM API, GPU code)
  - Yutong Zhou (GPU code, Folding@home)
  - Lee-Ping Wang, Diwakar Shukla, Gert Kiss (Simbios Distinguished Fellows)
  - Vijay Pande (co-PI, protein folding DBP lead PI)

- **Collaborators**
  - Kyle Beauchamp (PyMD)
  - John Chodera (Yank, PyOpenMM, tests)
  - Erik Lindahl (Stockholm)

- **Simbios**
  - Joy Ku (Director of Dissemination)
  - Russ Altman & Scott Delp (Simbios co-PI's)

# Introduction of OpenMM

Vijay Pande

OpenMM Workshop


# OpenMM

http://simtk.org/home/openmm


= rapid development +
rapid execution

OpenMM is an app, API, and library for rapid molecular dynamics.
Easy to modify and incorporate into any code.

# Two competing programming challenges

- **Rapid execution, but slow development**
  - traditional model
  - **Examples:** C, FORTRAN, Assembly

- **Rapid development, but slow execution**
  - new languages often used by scientists
  - **Examples:** Python, Perl

- **We can have it both ways: Domain specific languages (DSLs)**
  - New paradigm: specific language for application area
  - **Examples:** OpenGL, SQL, Matlab

- **A DSL approach for molecular mechanics?**
  - OpenMM hopes to provide **both** rapid execution and rapid development

# Goals

- **A complete library for molecular mechanics**
  - complete = what would need to do to do the most common calculations
  - complete != does everything conceivable

- **Fast and general**
  - **“rapid development *and* rapid execution”**
  - Don’t exposure hardware specifics
  - but optimize for speed underneath
  - broad support for GPUs, multicore, etc

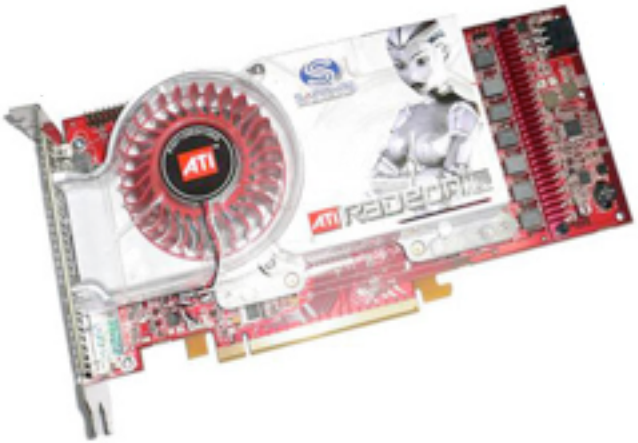
- **Two level API**
  - OpenMM for high level: intended for application developers
  - Low level API: for developers (mainly in-house & accelerator devs)
  - OpenMM can be a nexus for application and low level programmers to meet

# GPUs: very useful for rapid execution

- **Graphics Processing Units (GPUs) are very powerful**
  - Folding@home calculation circa 2003 = 10,000 PC's @ 1 GFLOP/PC = 10,000 GFLOPS
  - Fast GPU today = 1,000 GFLOP
  - Fast GPU cluster today = ~50,000 GFLOPS

- **GPU's are getting faster, faster than CPU's**
  - Moore's law is dead for traditional CPU's
  - we now see more cores per chip, but each core isn't any faster
  - GPU's figured out this trick a long time ago
  - typical GPU's now have 100's of cores
  - GPU's use their cores more efficiently

- **BUT, GPU's are horrible to program**
  - can't just recompile
  - must rethink algorithms
  - must understand the nature of the hardware
  - work closely with vendors (we collaborate closely with AMD/ATI, NVIDIA, and Intel)




*ATI X1900XT (500 GFlops peak, ~$100 + of a cost computer)*

*Sony PS3 (Cell processor: 220 GFlops peak, ~$400 total)*

# Unique aspects of comp biology on GPUs

- **Design algorithms that are GPU friendly**
  - FLOPS are free, memory is expensive
  - low lying fruit: algorithms which map well to GPUs

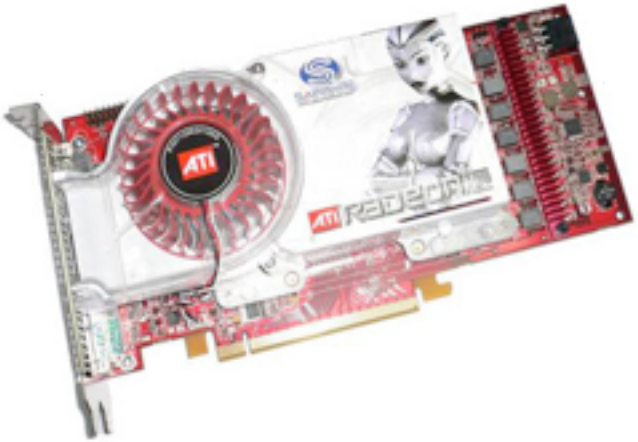
- **Code everything on the GPU**
  - If the original bottleneck is 90% of the calc, that's still only a 10x speed up at best
  - to get 100x to 1000x, one needs to have the whole calculation on the GPU (in our experience)

- **Centralized libraries, open source (eg OpenMM)**
  - avoid reinventing the wheel
  - build on others' work


- **Next steps**
  - not just speeding existing algorithms, but new methods
  - code methods which we wouldn't even dare to try now

# The opportunity for a common library

- **The molecular mechanics community has become fragmented**
  - tens of different MD codes
  - hardware acceleration (via multi-core, SSE, MPI, GPU's, and math coprocessors) is a critical element
  - much like the graphics community in the 1980's

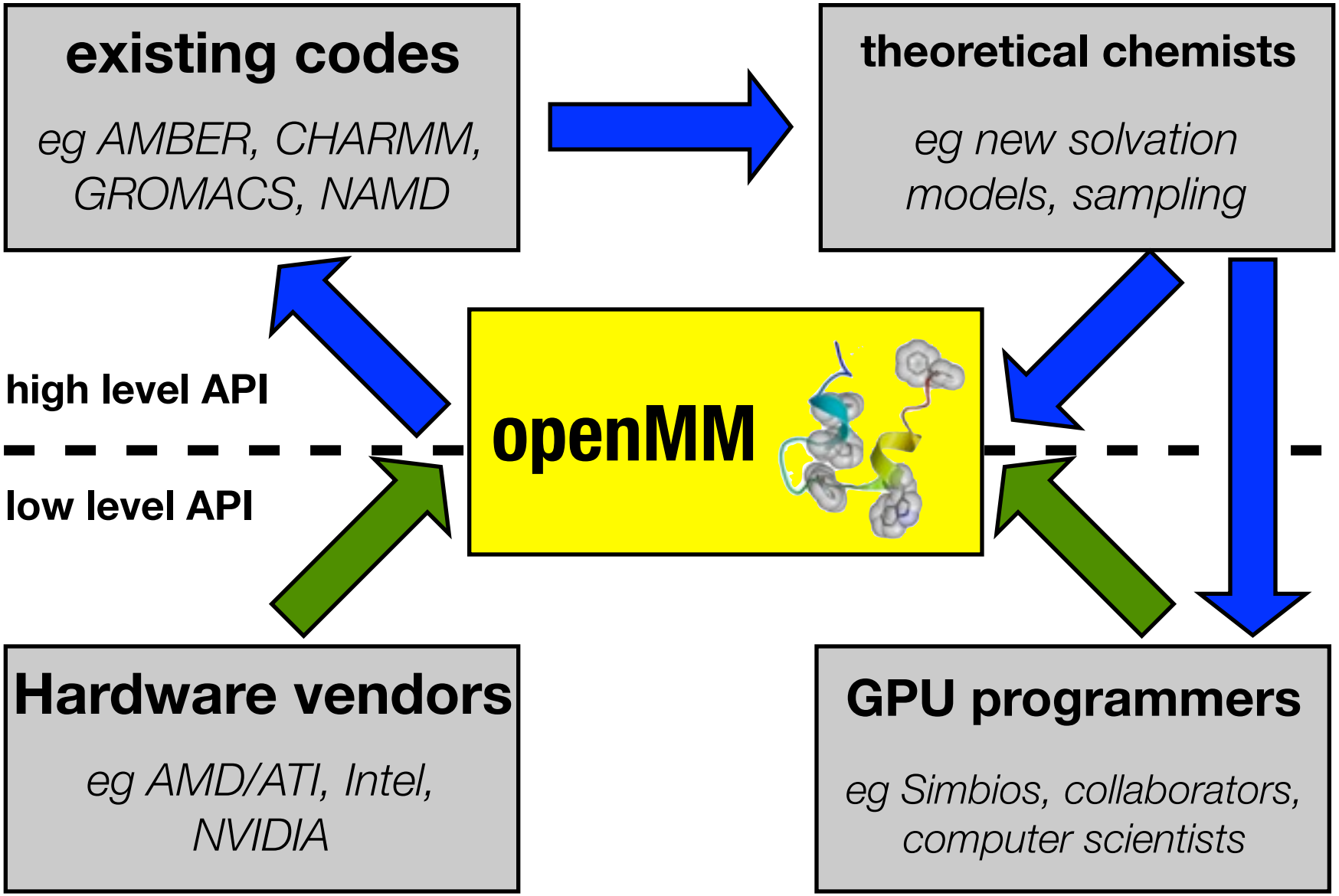



*ATI X1900XT (500 GFlops peak, ~$100 + of a cost computer)*

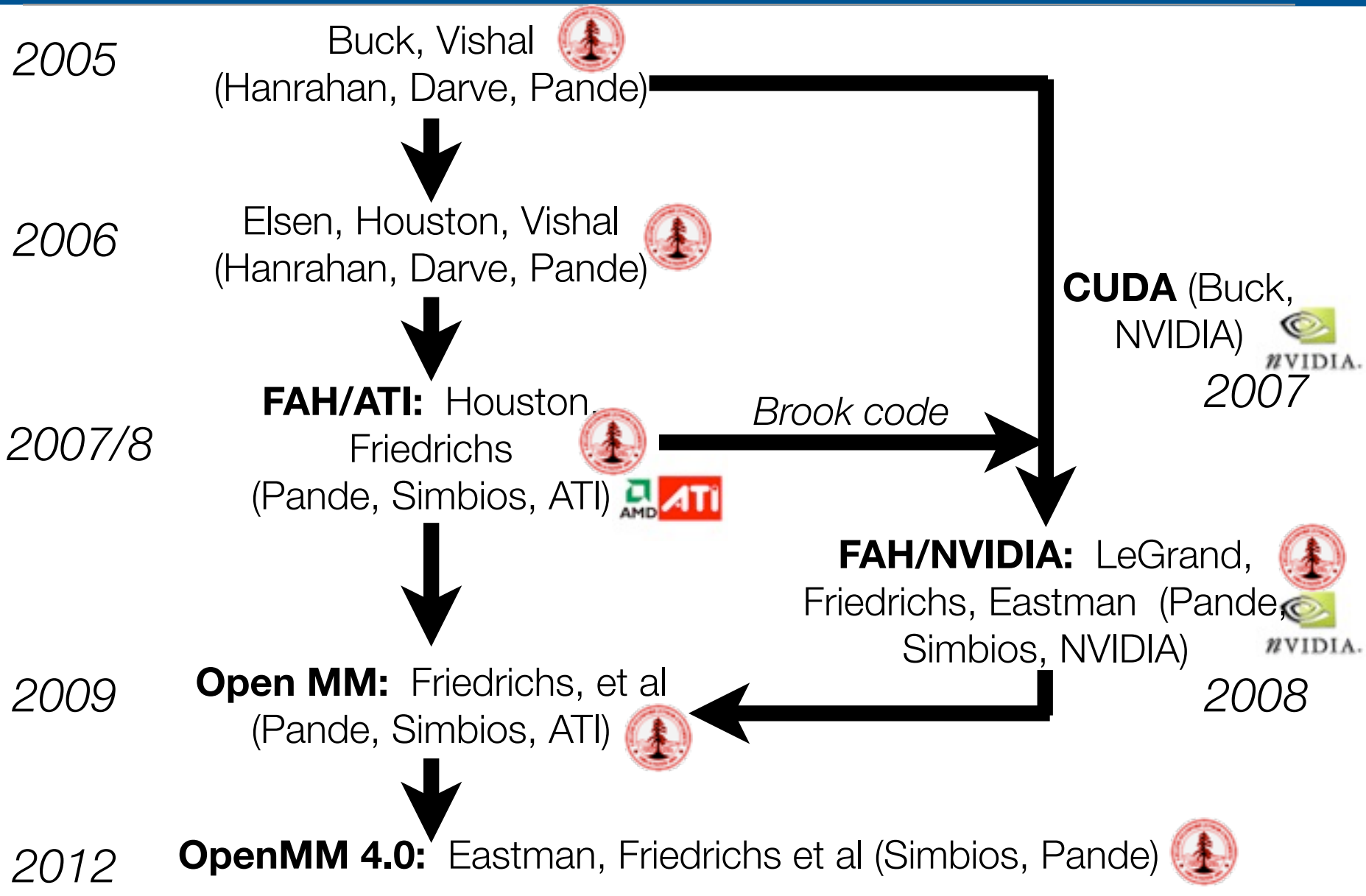
- **Hardware acceleration is a great unifying factor**
  - unifying API the way OpenGL unified graphics
  - incorporates hardware acceleration in its base design
  - this API would be used as the backend to existing codes, allowing for all to benefit from hardware acceleration

*Sony PS3 (Cell processor: 220 GFlops peak, ~$400 total)*

# Connections to OpenMM



# History of OpenMM



# OpenMM 5.1 speed improvements: JAC benchmark

| | CUDA (GTX 680) | OpenCL (GTX 680) | OpenCL (HD 7970) |
|---|---|---|---|
| **Implicit hbonds** | 148 | 134 | 120 |
| **Implicit hangles** | 240 | 209 | 104 |
| **RF hbonds** | 90.3 | 78.1 | 83.5 |
| **RF hangles** | 141.1 | 113.0 | 90.2 |
| **PME hbonds** | 61.0 | 41.5 | 49.3 |
| **PME hangles** | 99.9 | 66.9 | 63.0 |

Joint AMBER-CHARMM DHFR Benchmark in ns/day

# OpenMM is under active development

- 4.1 (~April 2012): Modeling tools, bugfixes
- 4.0 (January 2012): Python based app, multi-GPU support, Ring Polymer MD (RPMD)
- 3.1.1 (Aug. 11, 2011): bug fixes
- 3.1 (Aug. 1, 2011): performance improvements, especially to the AMOEBA plugin; support for parallelizing computations across multiple GPUs; support of direct polarization model by AMOEBA plugin; GB/VI force fully supported
- 3.0 (Mar. 30, 2011): supports AMOEBA force field; provides an energy minimizer; CMAP torsions; improved performance, especially for running on CPUs; Python API wrappers
- 2.0 (June 24, 2010): supports pressure coupling; provides custom Hbond forces; major performance improvements; works on ATI GPUs
- 1.1.1 (Mar. 4, 2010): bug fixes
- 1.1 (Feb. 12, 2010): provides custom torsion forces; bug fixes and other improvements
- 1.0 (Jan. 20, 2010): provides new custom forces, including bonds, angles, external, and GB; improved OpenCL support
- 1.0 Beta (Oct. 30, 2009): supports Particle Mesh Ewald (PME); custom nonbonded interactions, including algebraic and tabulated forces; preliminary OpenCL support
- Preview Release 4 (Aug. 20, 2009): supports Ewald summation; GPU accelerated energy calculations; C and Fortran API wrappers; a faster constraint algorithm; and many minor enhancements
- Preview Release 3 (May 19, 2009): provides support for explicit solvent on NVIDIA GPUs; includes periodic boundary conditions, cutoffs on non-bonded interactions, and new constraint algorithms.
- Preview Release 2 (Jan. 26, 2009): provides support for accelerating molecular modeling simulations on ATI and NVIDIA graphics processor units (GPUs)

# OpenMM roadmap: 2013

- **OpenMM 5.1**
  - Big speed increases over 5.0 (2x faster)
  - Integration with CHARMM
  - Accelerated MD
- **OpenMM 5.2**
  - ABSINTH implicit solvent
  - AMOEBA OpenCL implementation
  - Colored noise integrator
  - LTMD
  - CHARMM* force fields
- **Longer term**
  - CUDA JIT (when CUDA supports it)
  - More optimizations
  - Custom compound bonded force
  - Finalize Rosetta force field
  - Triclinic boxes
  - A more accurate SASA calculation for use with GB models
  - QM/MM

http://wiki.simtk.org/openmm/RoadmapTimeline

# Looking further to the future: FAQ

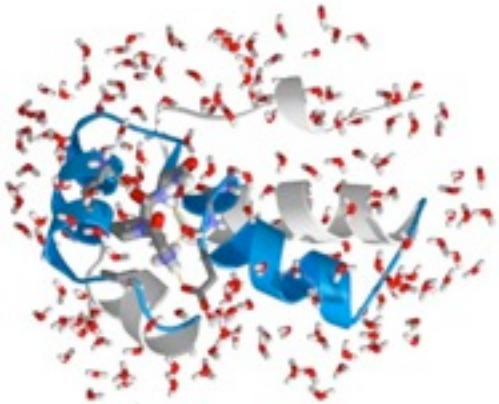
- **Scheme to manage API for years to come**
  - design decisions were made to address trade-off between generality/ performance: goals are performance and simplicity, features come 2nd
  - speed/simplicity/elegance were the motivations for these decisions

- **How will the API be updated in the future?**
  - As OpenMM has progressed, we have brought in other key developers to advise us, including Erik Lindahl (Gromacs), Jesus Izaguirre (Protomol), & Ross Walker (AMBER) as well as the Simbios SAB to advise us for useful directions for the API
  - We plan to formalize this to have an advisory board for OpenMM specifically

- **How difficult is it for hardware vendors to produce mature and performant OpenMM "drivers"?**
  - Strong success working with NVIDIA, ATI, and Sony and are making connections/ collaborations with other key players (OpenCL, Larrabee)
  - Due to its design and the existence of key OpenMM apps (eg Folding@home, Gromacs), OpenMM will be able to stay current

# Examples of current OpenMM enabled applications

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



Gromacs
http://www.gromacs.org

CHARMM

CHARMM
http://www.charmm.org

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

Protomol
http://protomol.sourceforge.net


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

# Licensing and distribution

- **API & reference BSD license, GPU kernels are LGPL**
  - free & open
  - we want LGPL to have a community owned set of GPU kernels
  - we're looking for collaborations for new features

- **But, please cite us**
  - Any work that uses OpenMM should cite the following paper: M. S. Friedrichs, P. Eastman, V. Vaidyanathan, M. Houston, S. LeGrand, A. L. Beberg, D. L. Ensign, C. M. Bruns, V. S. Pande. "Accelerating Molecular Dynamic Simulation on Graphics Processing Units." J. Comp. Chem., (2009)
  - http://www3.interscience.wiley.com/journal/121677402/abstract

# Summary

- **What is it**
  - API, library, and application for core molecular dynamics / molecular mechanics applications
  - emphasis on rapid development *and* rapid execution speed (eg hardware acceleration)
  - dual APIs (one for applications and one for low level hardware)
  - open source (LGPL) software

- **What is it not**
  - a general solution for all possible molecular mechanics tasks
  - a compiler which can turn a generic MD code into accelerated code