


# MDSimAid: Automatic optimization of fast electrostatics in molecular simulations

*Jesús A. Izaguirre*, Michael Crocker, Alice Ko, Thierry Matthey and Yao Wang

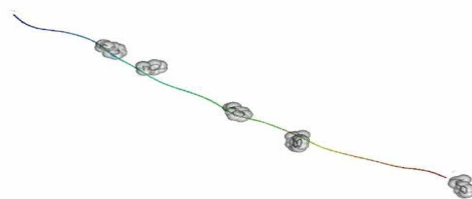
Corresponding author: [izaguirr@cse.nd.edu](mailto:izaguirr@cse.nd.edu)

Department of Computer Science and Engineering  
University of Notre Dame, USA

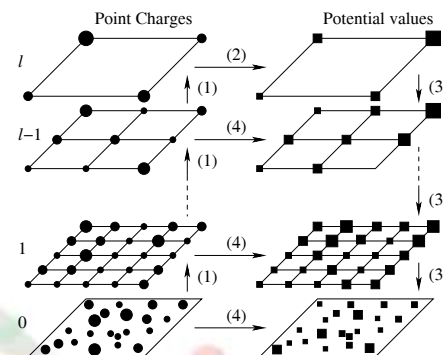


# Talk Roadmap

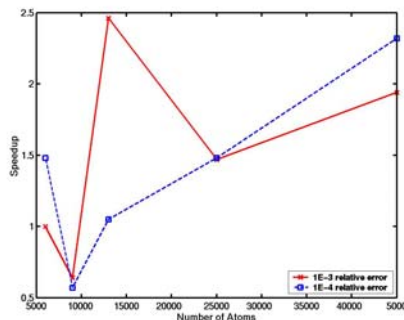
1. Motivation:  
automatic tuning of  
molecular simulations



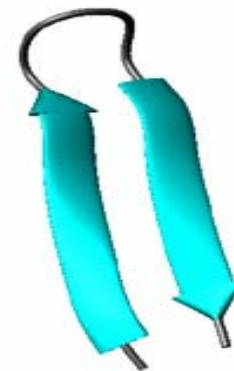
2. Key to performance:  
fast electrostatics



3. Evaluation of MDSimAid  
recommender system



4. Discussion and extensions



# Motivation



- ◆ Create a recommender system/ self-adaptive software for running molecular dynamics (MD) simulations
- ◆ Importance of MD
  - Protein dynamics, e.g., protein folding
  - Sampling and thermodynamics, e.g., drug design
- ◆ Complicated to use effectively
  - Too many algorithms for performance critical sections, with complex parameter relationships

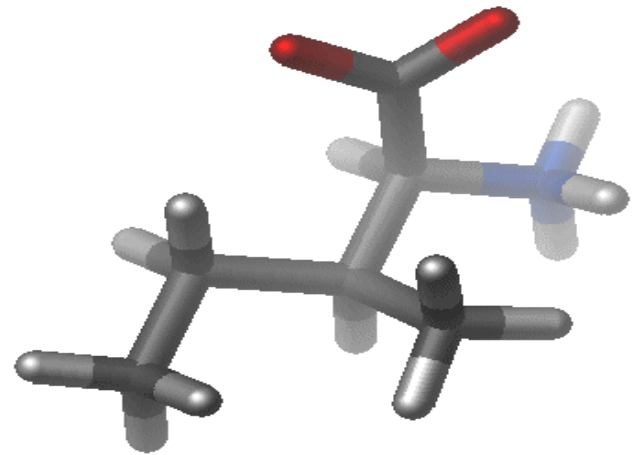
# Classical molecular dynamics

- ◆ Newton's equations of motion:

$$\mathbf{M}\mathbf{q}'' = -\nabla U(\mathbf{q}) = \mathbf{F}(\mathbf{q}). \quad \text{--- (1)}$$

- ◆ Molecules
- ◆ CHARMM force field

(Chemistry at Harvard  
Molecular Mechanics)



Bonds, angles and torsions

# Energy Functions

$$U(\vec{R}) = \underbrace{\sum_{\text{bonds}} k_i^{\text{bond}} (r_i - r_0)^2}_{U_{\text{bond}}} + \underbrace{\sum_{\text{angles}} k_i^{\text{angle}} (\theta_i - \theta_0)^2}_{U_{\text{angle}}} + \underbrace{\sum_{\text{dihedrals}} k_i^{\text{dihe}} [1 + \cos(n_i \phi_i + \delta_i)]}_{U_{\text{dihedral}}} + \underbrace{\sum_i \sum_{j \neq i} 4 \epsilon_{ij} \left[ \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^6 \right] + \sum_i \sum_{j \neq i} \frac{q_i q_j}{\epsilon r_{ij}}}_{U_{\text{nonbond}}}$$

$U_{\text{bond}}$  = oscillations about the equilibrium bond length

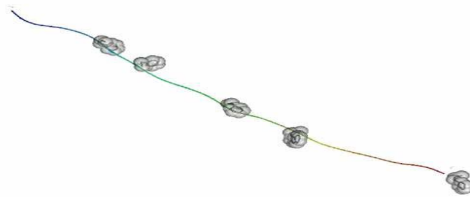
$U_{\text{angle}}$  = oscillations of 3 atoms about an equilibrium angle

$U_{\text{dihedral}}$  = torsional rotation of 4 atoms about a central bond

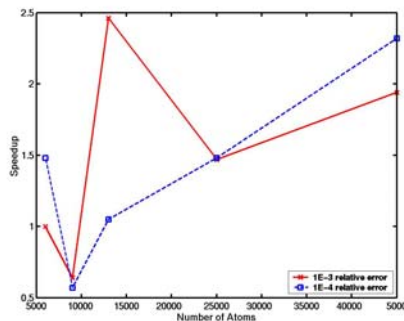
$U_{\text{nonbond}}$  = non-bonded energy terms (electrostatics and Lennard-Jones)

# Talk Roadmap

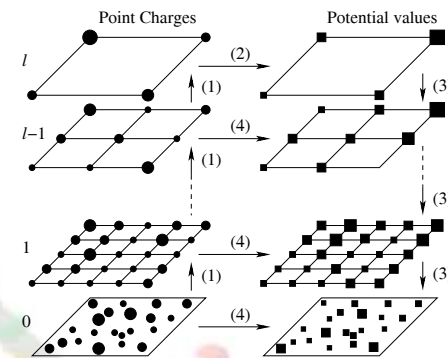
1. Motivation:  
automatic tuning of  
molecular simulations



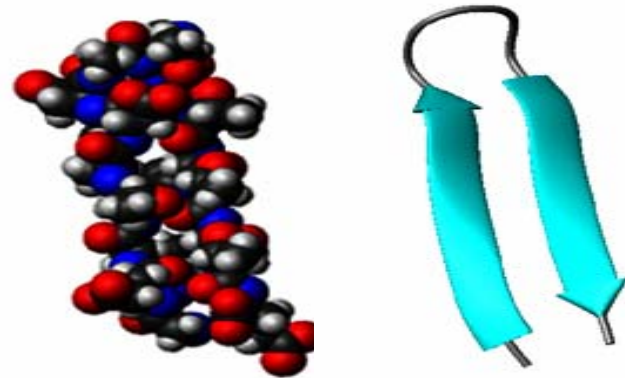
3. Evaluation of MDSimAid  
recommender system



2. Key to performance:  
fast electrostatics



4. Discussion and extensions



# Performance critical section: Electrostatics computation

$$U(\vec{r}_1, \vec{r}_2, \dots, \vec{r}_m) = \frac{1}{2} \sum_{\vec{n}' \in \mathbb{Z}^3} \sum_i \sum_j \frac{q_i q_j}{|\vec{r}_j - \vec{r}_i + \vec{n}'L|} \quad (1)$$

This is a conditionally convergent sum. Ewald (1921) figured how to *split* this sum into two rapidly convergent sums:

$$\frac{1}{r} = \underbrace{\frac{f(r)}{r}}_{\text{Short range}} + \underbrace{\frac{1-f(r)}{r}}_{\text{Smooth}}$$

# Derivation of fast electrostatics methods I

Ewald used the following function:

$$f(r) = \operatorname{erfc}(\alpha r) = \frac{2}{\sqrt{\pi}} \int_{\alpha}^{\infty} \exp(-s^2) ds$$

The short range part is usually solved directly. The smooth part may be solved by a Fourier series, giving rise to *Ewald* methods:

$$\frac{1}{2} \sum_{\vec{k} \neq 0} \frac{1}{k^2} e^{-\frac{k^2}{4\alpha^2}} \left[ \underbrace{\sum_j q_j \exp(-i\vec{k} \cdot \vec{r}_j)}_{\hat{\rho}(\vec{k})} \right]^2$$

# Derivation of fast electrostatics methods II

Ewald chooses splitting parameter so that work is evenly spaced as  $O(N^{3/2})$

Particle Mesh Ewald (PME) chooses splitting parameter such that short-range part is  $O(N)$ , and interpolates Fourier series to a mesh, thus allowing the use of  $O(N \log N)$  FFT

$$\hat{\rho}(\vec{k}) \cong \sum_n \exp(-i\vec{k} \cdot \vec{r}_n^h) \sum_j \phi_n(F_j) q_j$$

# Fast electrostatics algorithms

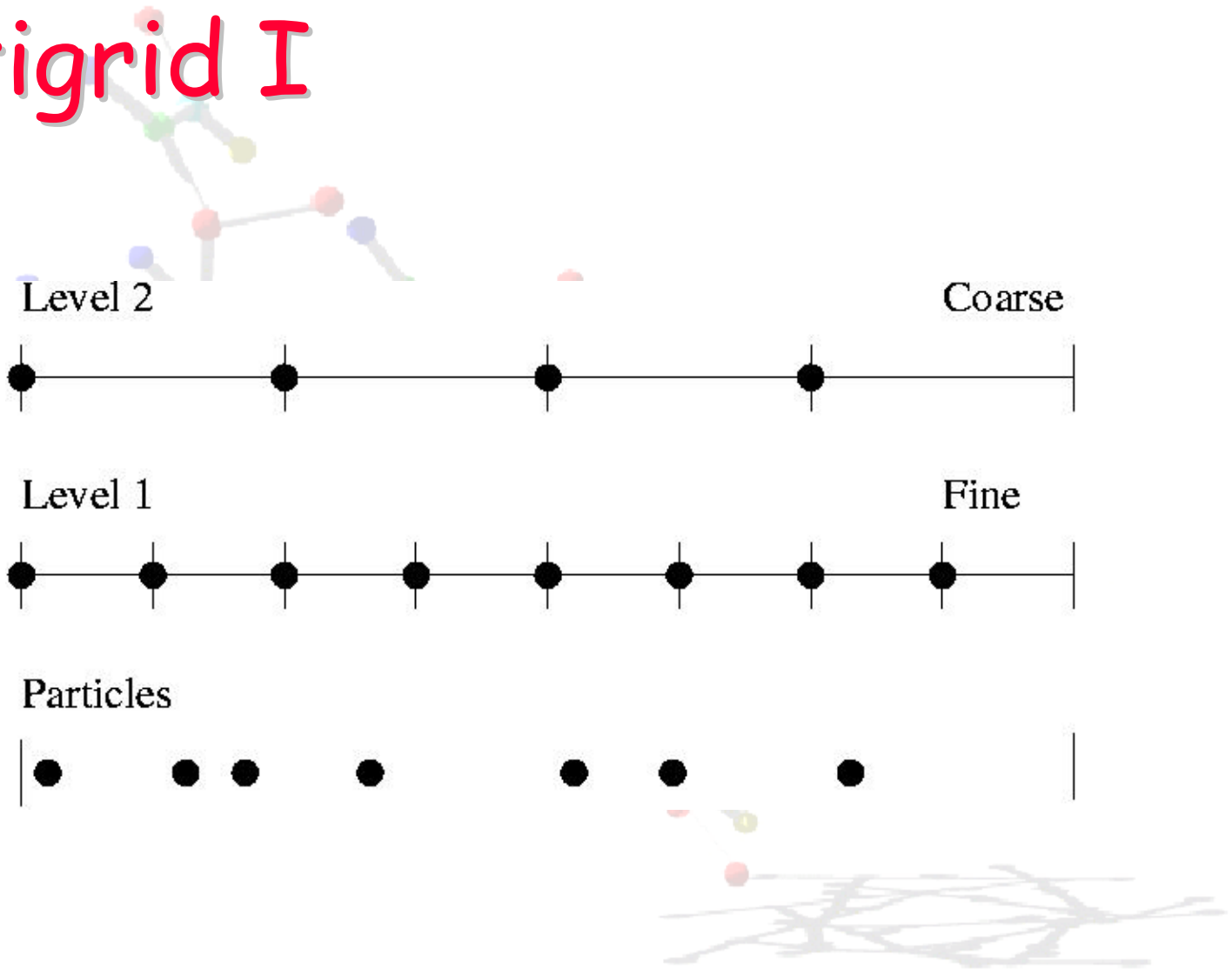
There are many methods. Which one to use for a given system and accuracy?

Ewald summation	$O(N^{3/2})$	Ewald, 1921
Fast Multipole Method	$O(N)$	Greengard, 1987
Particle Mesh Ewald	$O(N \log N)$	Darden, 1993
Multi-grid summation (dense mat-vec as a sum of sparse mat-vec)	$O(N)$	Brandt <i>et al.</i> , 1990 Skeel <i>et al.</i> , 2002 Izaguirre <i>et al.</i> , 2003

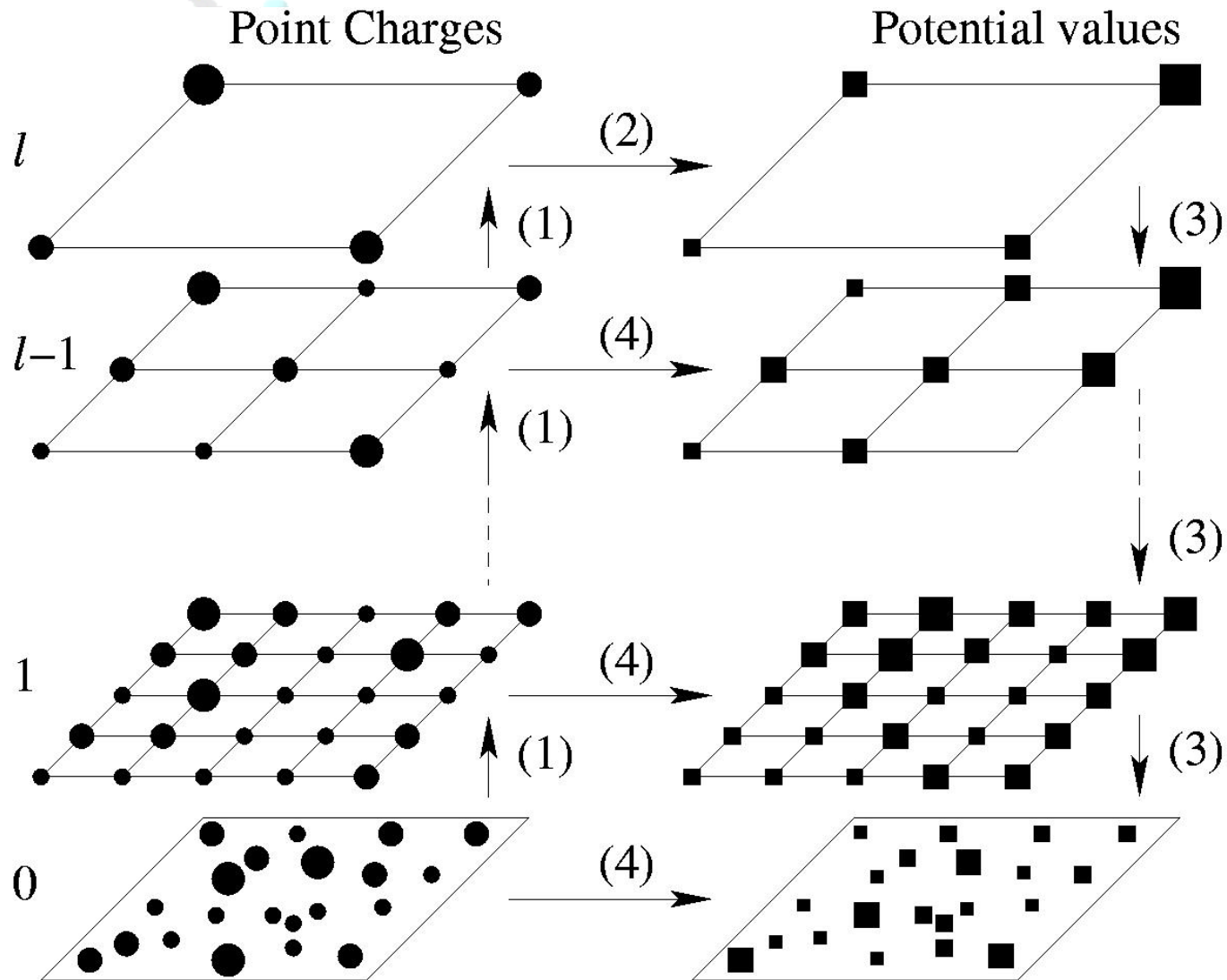
# Particle Mesh Ewald

- ◆ Following Ewald, separates the electrostatic interactions into two parts:
  - Direct-space short range evaluation
  - Fourier-space evaluation
- ◆ The Fourier term is approximated by using fast Fourier transforms on a grid
- ◆ Method parameters are grid size and cutoff of direct-space

# Multigrid I



# Multigrid II

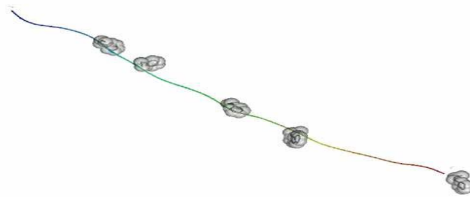


# Multigrid III

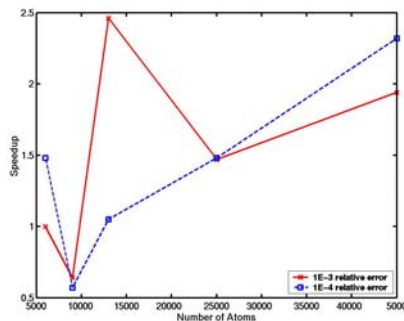
- ◆ Complex relationship among method parameters:
  - Cutoff and softening distances for potential evaluation at the particle and grid levels
  - Grid size and interpolation order
  - Number of levels
- ◆ Rules extracted from extensive evaluation encapsulated in MDSimAid
  - Fine tuned at run-time by running selected tests
  - Makes these methods easier to use

# Talk Roadmap

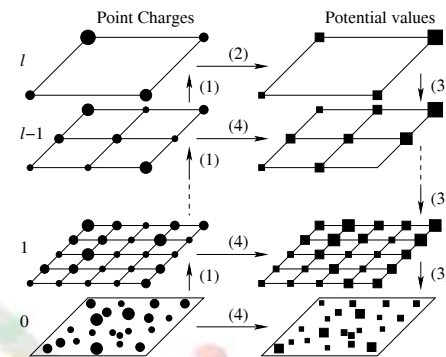
1. Motivation:  
automatic tuning of  
molecular simulations



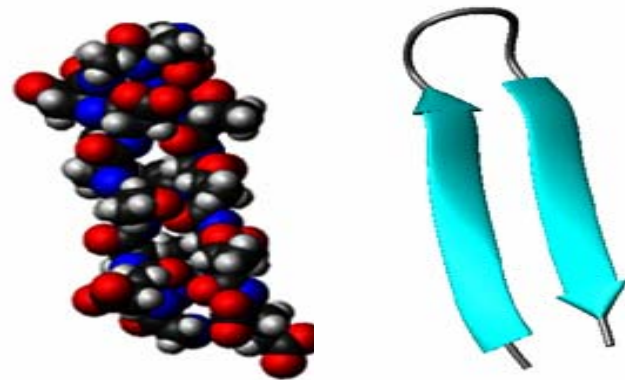
3. Evaluation of MDSimAid  
recommender system



2. Key to performance:  
fast electrostatics



4. Discussion and extensions



# Related Work I: Performance Models



- Darden et al., *J. Chem. Phys.* 1993
  - effect of varying parameters of Particle Mesh Ewald
- Petersen et al., *J. Chem. Phys.* 1995
  - accuracy and efficiency of Particle Mesh Ewald
- Krasny et al., *J. Chem. Phys.* 2000
  - used FMM to compute direct part of Ewald sum
- Skeel et al., *J. Comp. Chem.* 2002
  - study of parameters for multigrid (MG) method. Compared MG to Fast Multipole Method (FMM). MG faster than FMM for low accuracy

# Related Work II: Limitations

- ◆ Most published results
  - fail to suggest how to determine the specific values
  - provide general trends only
  - contain unknown constants in equations that model performance
  - Do not account for modern computer architectures, particularly memory subsystem
- ◆ For example, using parameters for MG suggested by MDSimAid gave an order of magnitude better accuracy and between 2 to 4 times faster execution than suggested by analytical results (Ko, 2002)

# Summary

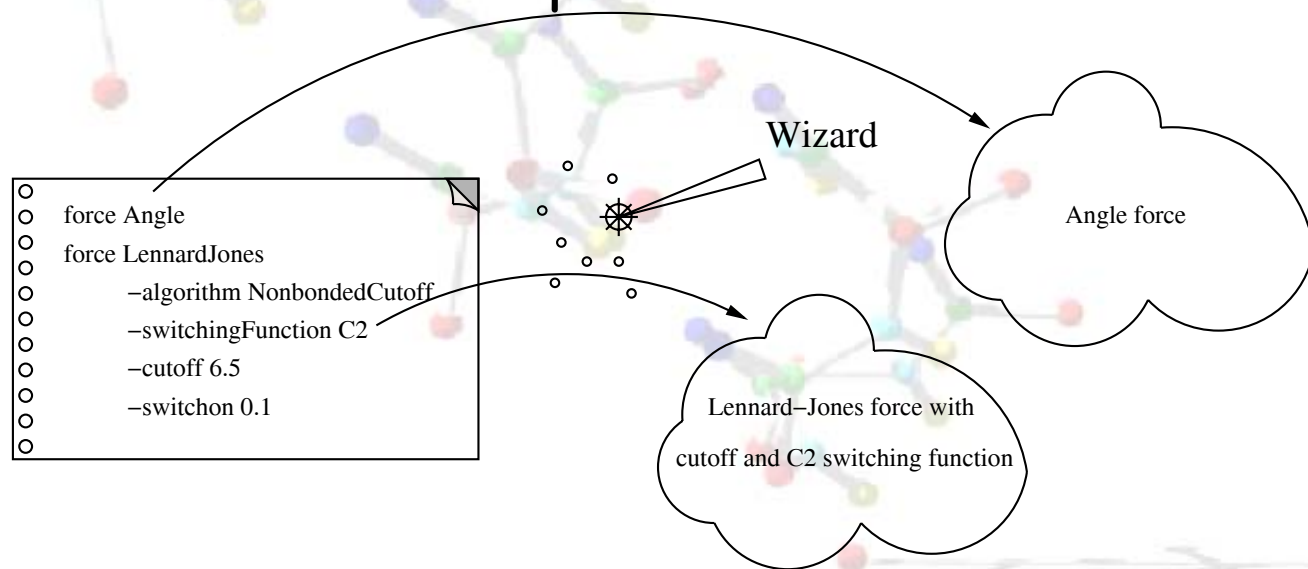
- ◆ General contributions of this study
  - Practical guidelines for choosing parameters for each fast electrostatics algorithms, and to choose among different algorithms
    - Implemented important algorithms with reasonable efficiency in ProtoMol
    - Tested algorithms for various system sizes and accuracy
    - Tested quality of these methods for MD of solvated proteins
  - Encapsulated results on a tool called MDSimAid
  - Hybrid rule-based and run-time optimization for choosing algorithm and parameter for MD

# Experimental protocol

- ◆ These methods were tested and implemented in a common generic and OO framework, ProtoMol:
  1. Smooth Particle Mesh Ewald
  2. Multigrid summation
  3. Ewald summation
- ◆ Testing protocol:
  - Methods (1) and (2) above were compared against (3) to determine accuracy and relative speedup
  - Tested on water boxes and protein systems ranging from 1,000 to 100,000 atoms, and low and high accuracies
  - For selected protein systems, structural and transport properties were computed (e.g., Melittin, pdb id 2mlt, in water, 11845 atoms)

# Software adaptation I

- (1) Domain specific language to define MD simulations in ProtoMol (Matthey and Izaguirre, 2001-3)
- (2) "JIT" generation of prototypes: factories of template instantiations



# Software adaptation II

## (3) Timing and comparison facilities:

force compare time force Coulomb -algorithm PMEwald

-interpolation BSpline -cutoff 6.5 -gridsize 10 10 10

force compare time force Coulomb -algorithm PMEwald

-interpolation BSpline -cutoff 6.5 -gridsize 20 20 20

## (4) Selection at runtime - extensible module using Python

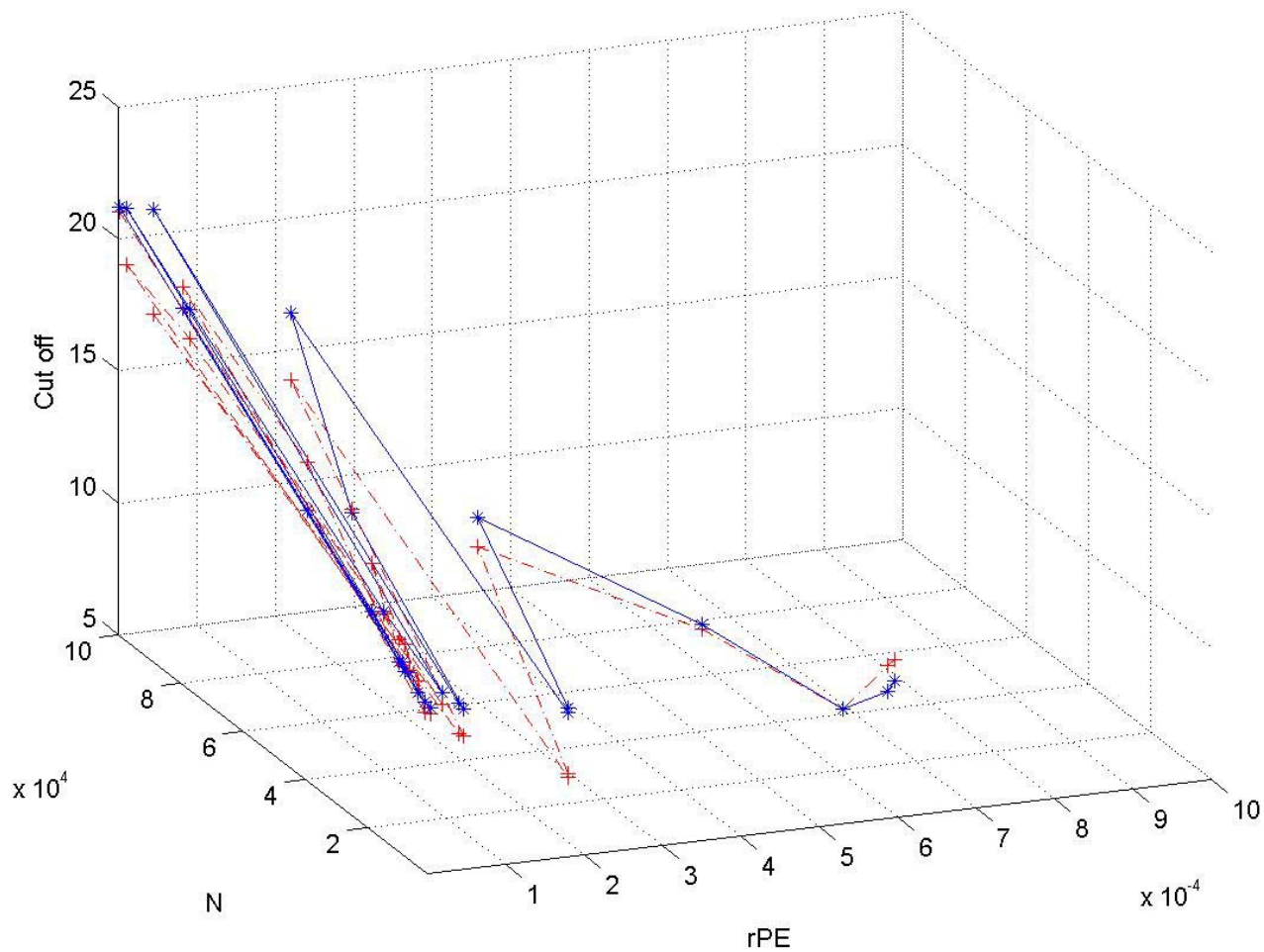
```
trial_param1[i]=cutoffvalue+1
```

```
test_PME(cutoffvalue+1,gridsize,...,  
         accuracy)
```

# Optimization strategies I

- (1) Rules generated from experimental data & analytical models (thousands of data points)
- (2A) At run-time,  $M$  tests are tried by exploring the parameter that changes accuracy or time most rapidly (*biased search*), or
- (2B) At run-time,  $M$  tests are tried by randomly exploring all *valid* method parameter combinations
- (3) Choose the fastest algorithm/parameter combination within accuracy constraints

# Example of parameter space exploration for PME

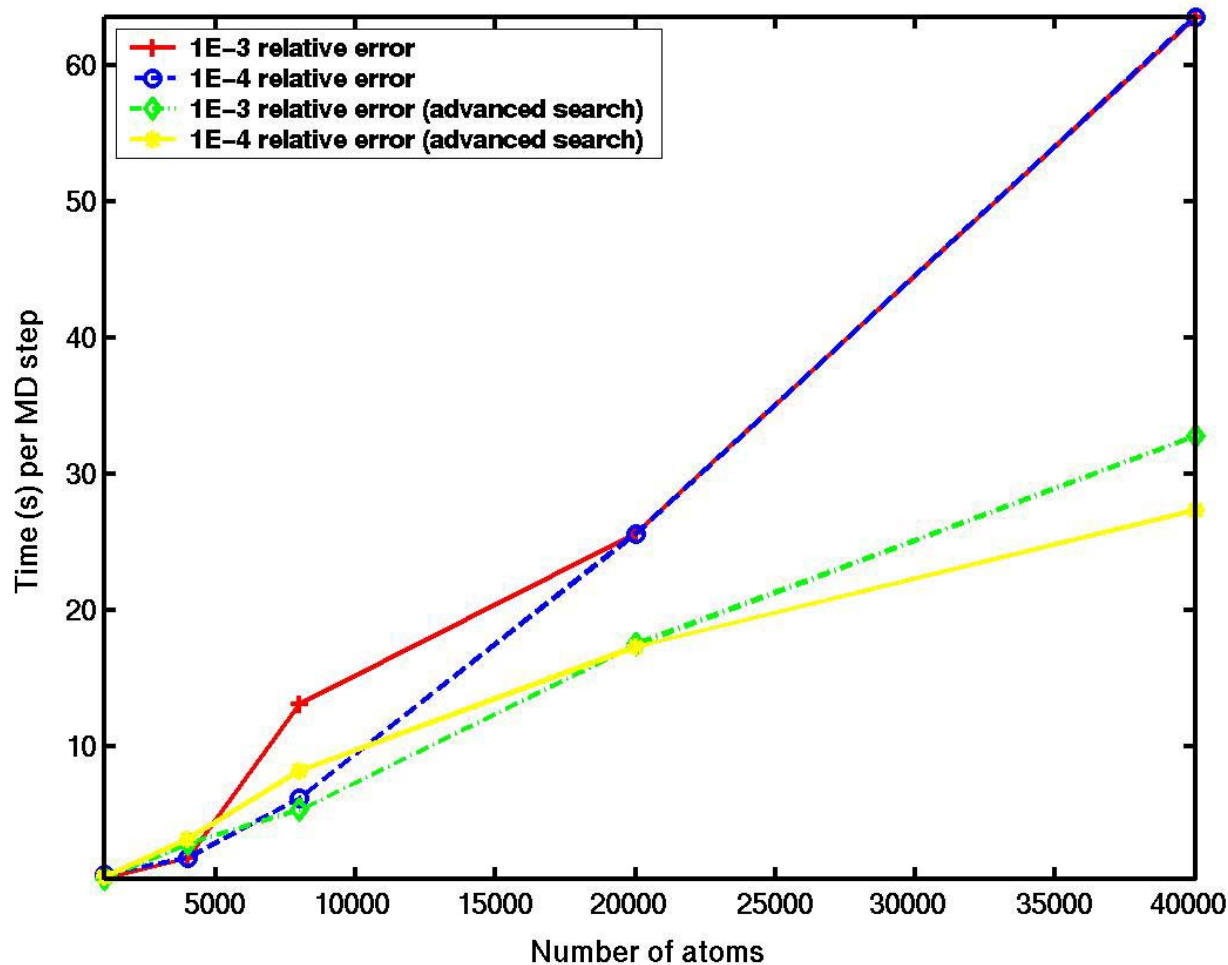


# Examples of rules in MDSimAid

- ◆ Rules can be generated automatically using inductive decision trees or regression rules (or any machine learning technique). Example:
  - Regression for PME:  $\text{cutoff} = 0.0001 * N - 4750.498 * rPE + 11.186$
  - Regression tree for MG-Ewald:
    - rPE  $\leq 1E-5$  : LM1 (51/76.5%)
    - rPE  $> 1E-5$  :
      - | rPE  $\leq 1E-4$  : LM2 (12/45.6%)
      - | rPE  $> 1E-4$  : LM3 (41/54.3%)
  - Models at the leaves:
    - LM1: level = 2.96
    - LM2: level = 1.83
    - LM3: level = 1.27
- ◆ Rules can be evaluated by correlation coefficients and other metrics:
  - Correlation coefficient for PME 0.9146
  - Correlation coefficient for MG-Ewald 0.8319

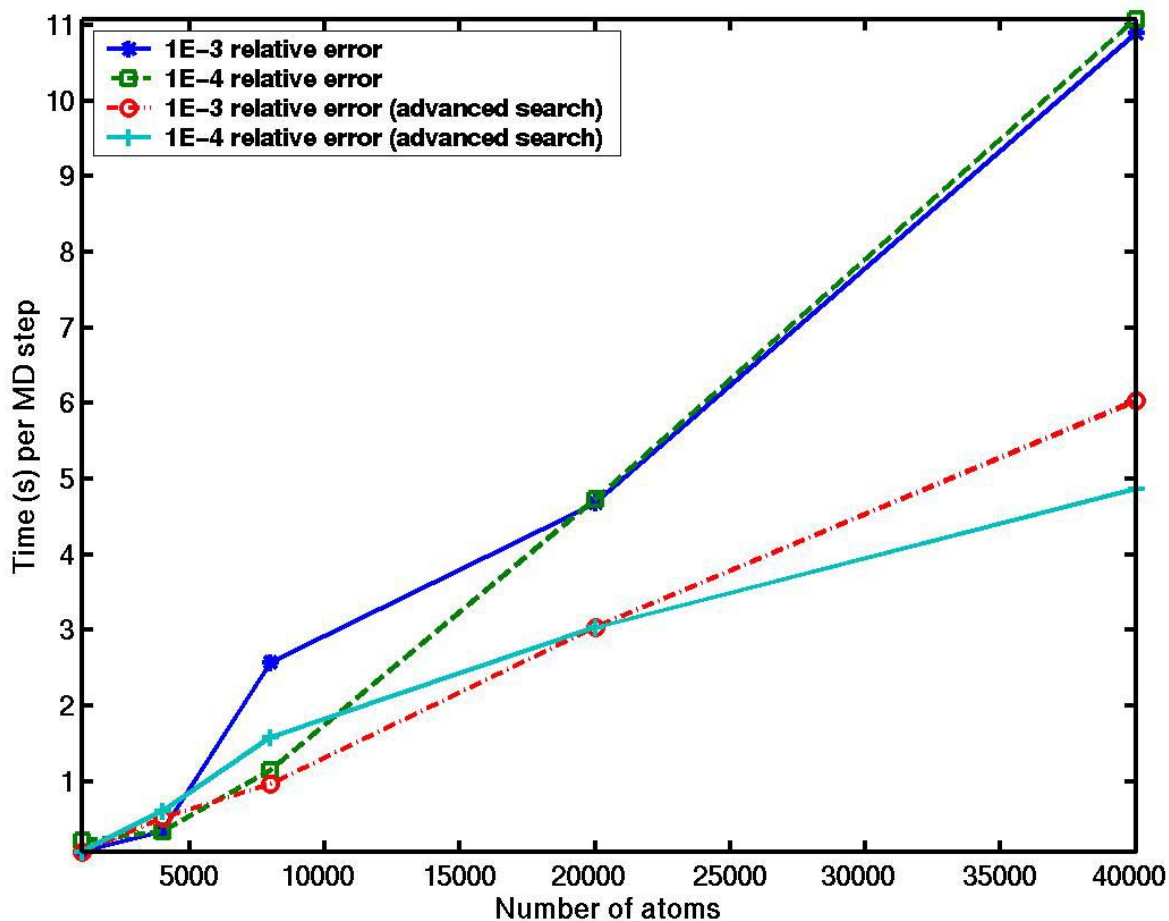
# Fastest algorithms found I

Platform: Solaris; Biased search with 3 trials and unbiased with 5 trials



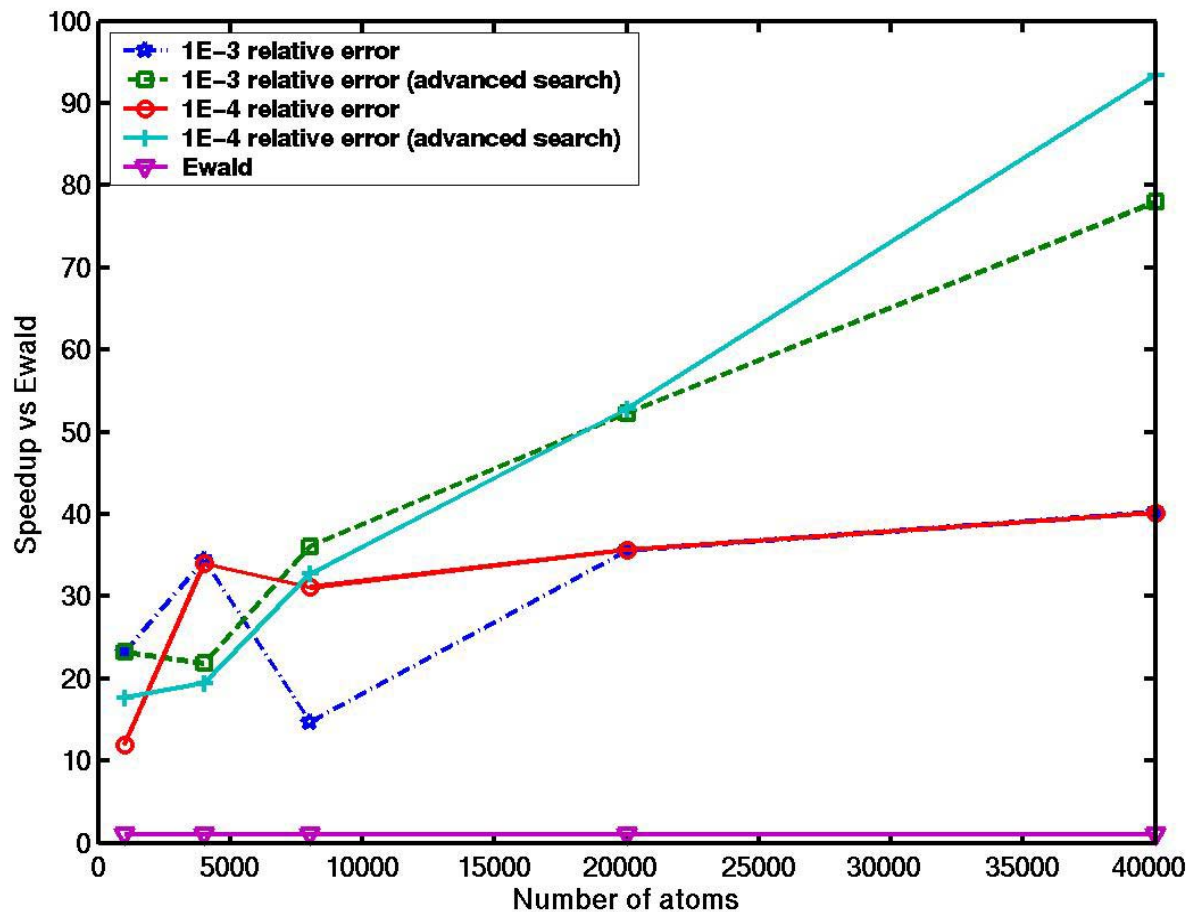
# Fastest algorithms found II

Platform: Linux; Biased search with 3 trials and unbiased with 5 trials



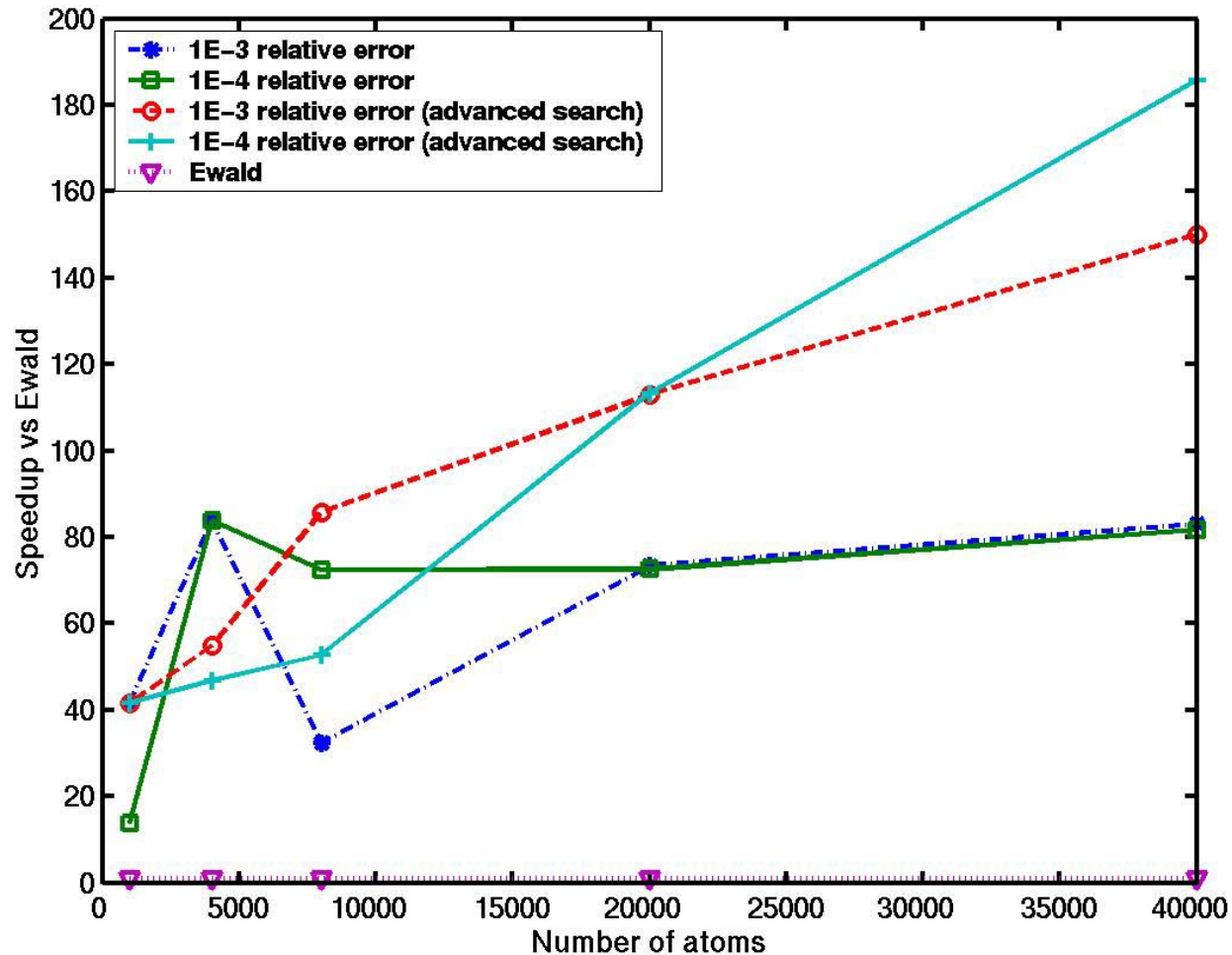
# Fastest algorithms found III

Platform: Solaris; Biased search with 3 trials and unbiased with 5 trials



# Fastest algorithms found IV

Platform: Linux; Biased search with 3 trials and unbiased with 5 trials

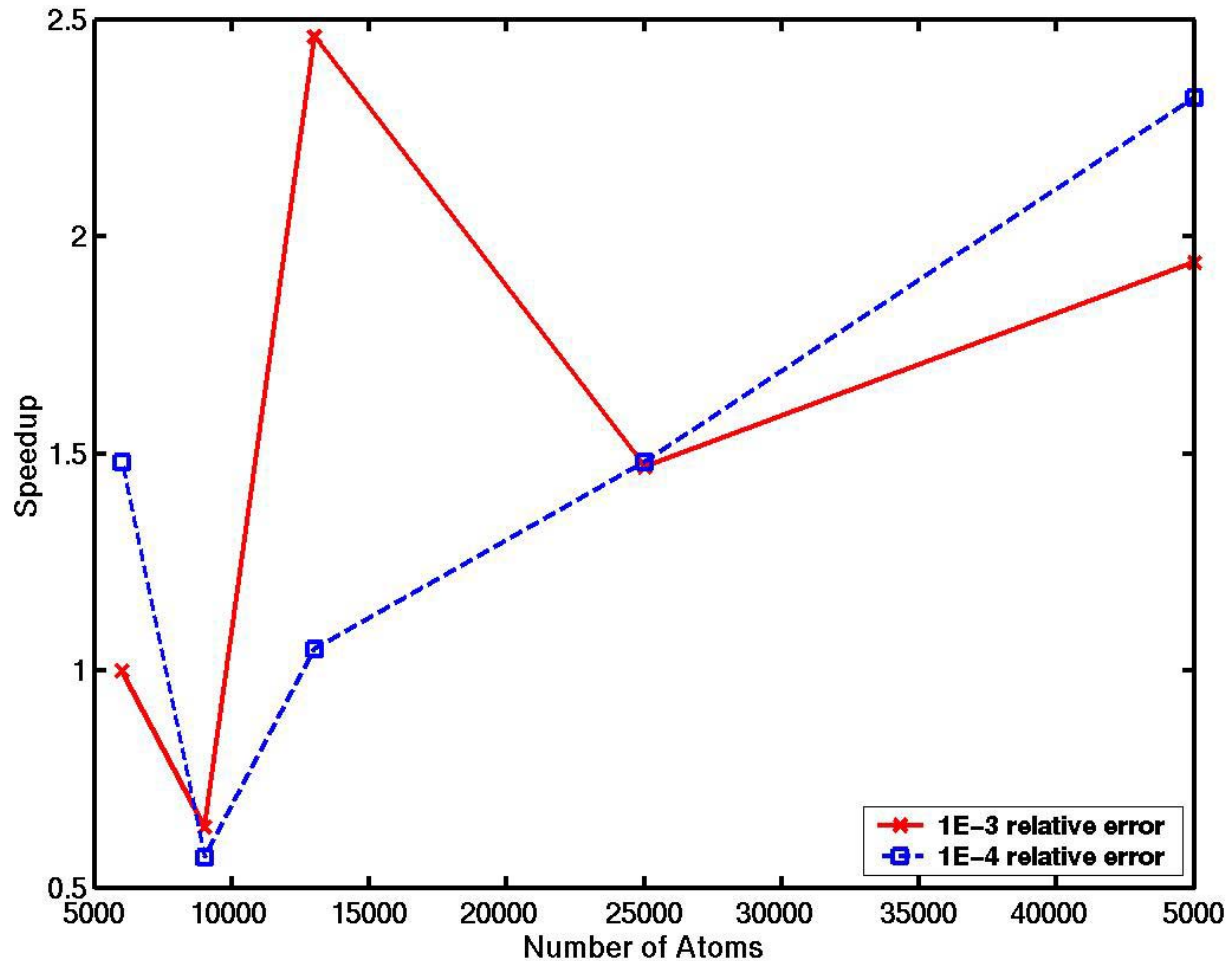


# Optimization strategies II

- ◆ Hybrid optimization strategies work well for algorithmic tuning:
  - Possible to obtain rules for general trends
  - Runtime optimization provides fine-tuning and may signal invalid rules
- ◆ Rules provide initial guess; unbiased local search in parameter space provides improvements
  - In our tests, local search found three times faster fast electrostatic algorithms, at a cost of 3 times more computational search

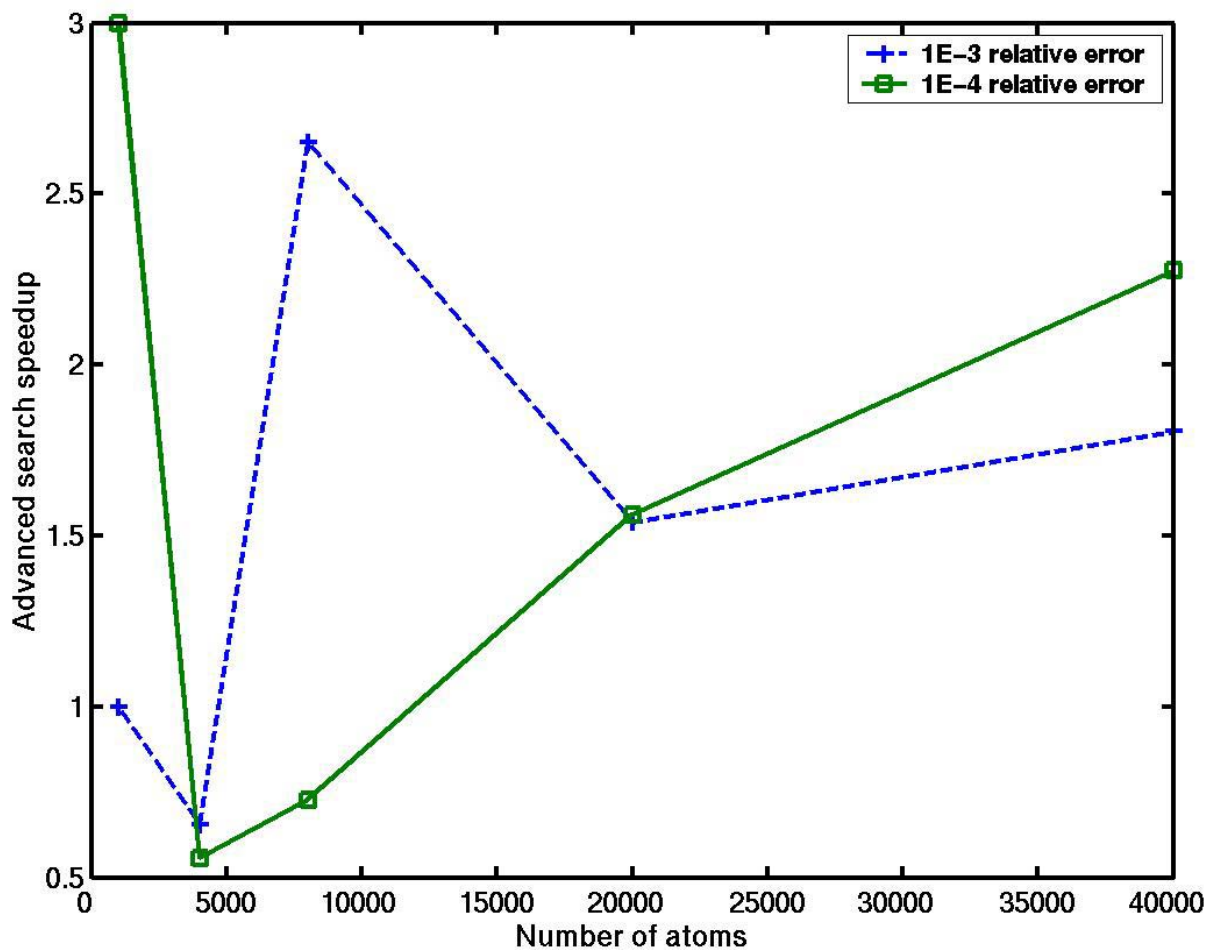
# Hybrid optimization speedup

Platform: Solaris

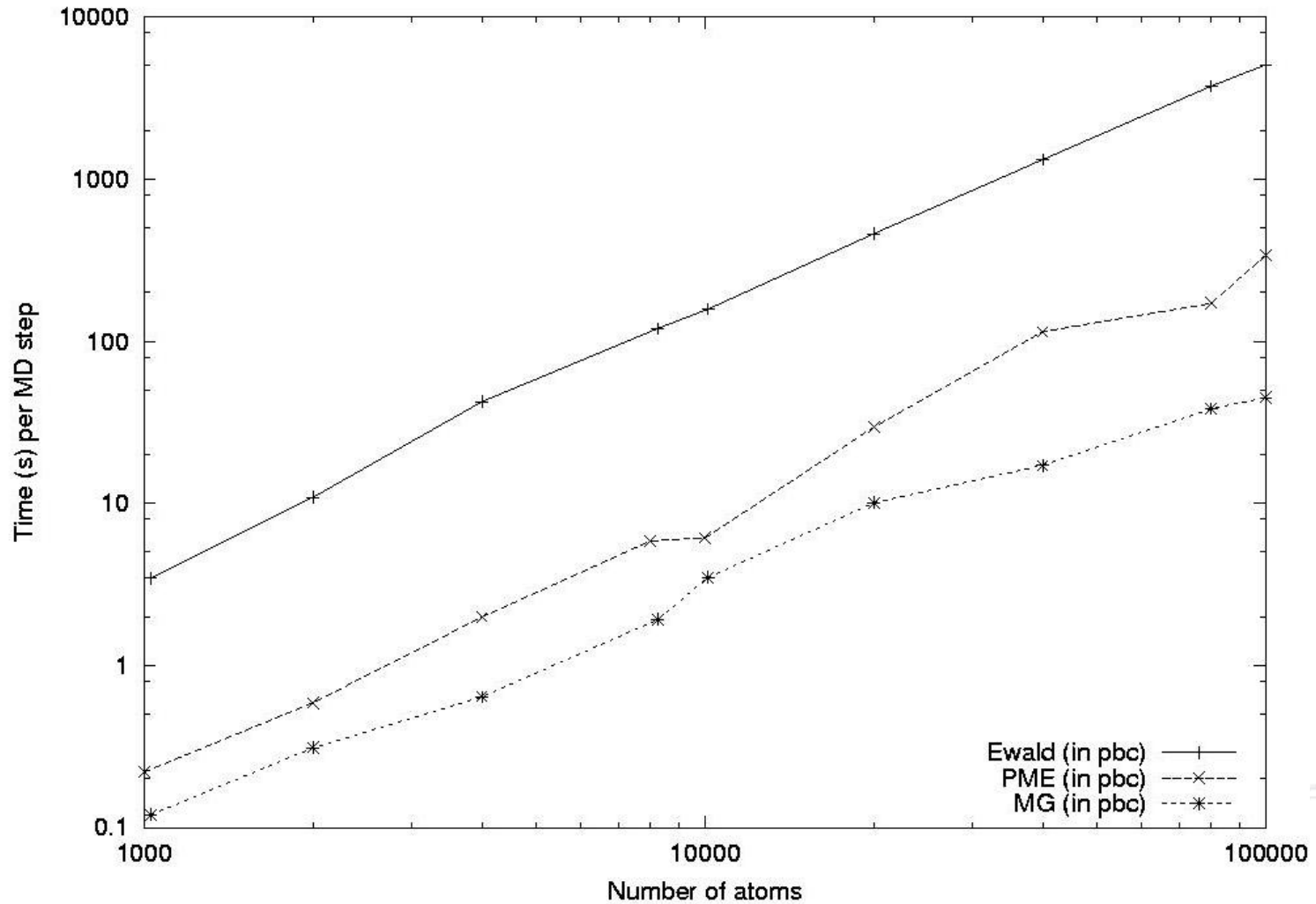


# Hybrid optimization speedup

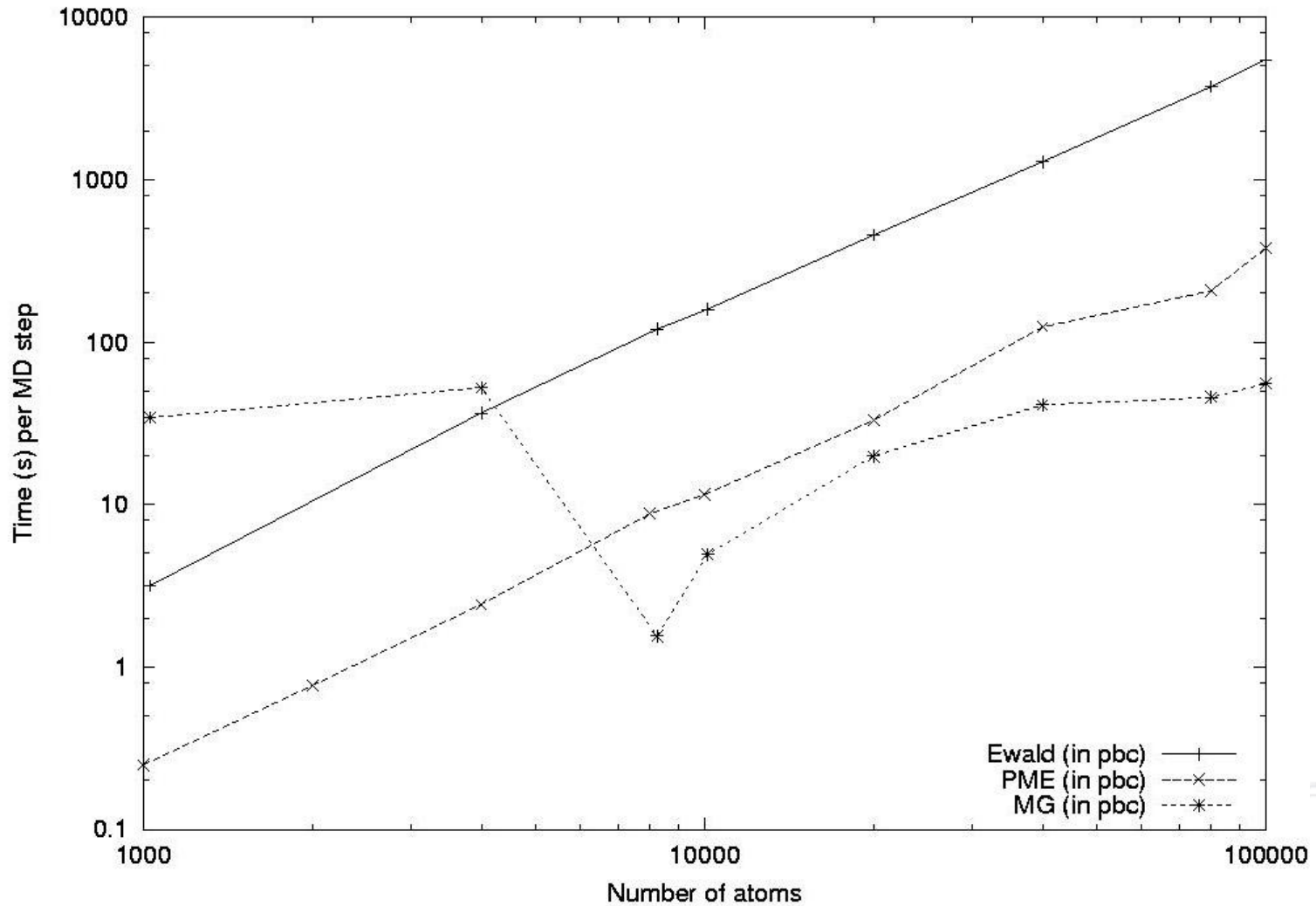
Platform: Linux



# Results ( $10^{-4}$ rPE)

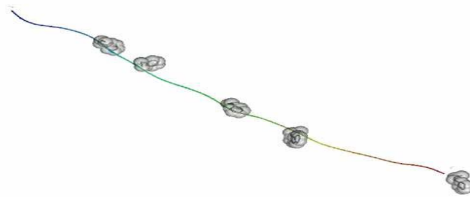


# Results ( $10^{-5}$ rPE)

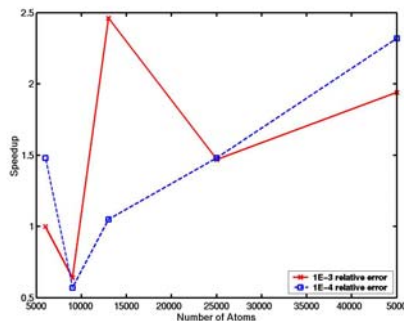


# Talk Roadmap

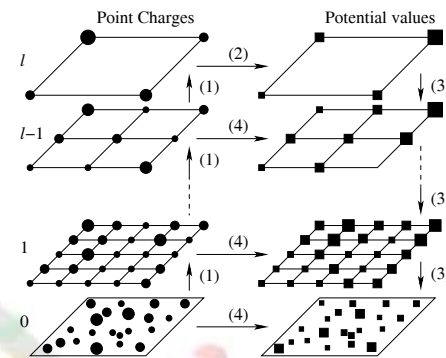
1. Motivation:  
automatic tuning of  
molecular simulations



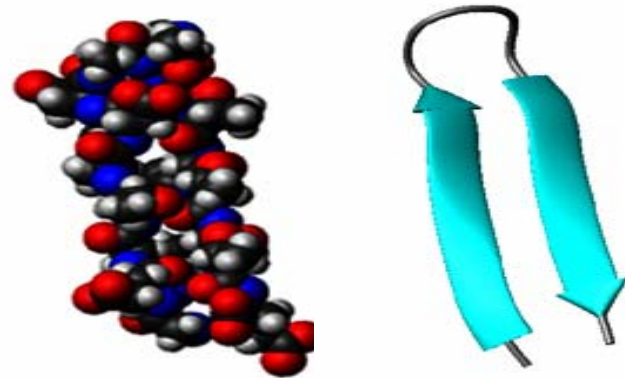
3. Evaluation of MDSimAid  
recommender system



2. Key to performance:  
fast electrostatics



4. Discussion and extensions



# Related Work III

- ◆ Algorithm or software selection problem (Rice, 1976)
- ◆ Optimization at higher level than PHIPAC, ATLAS, etc.
- ◆ Optimization approach similar to SPIRAL (Moura *et al.*, 2000)
- ◆ Some rules are generated by inductive methods, similar to PYTHIA II (Houstis *et al.*, 2000)
- ◆ Some rules come from performance models, similar to SALSA (Dongarra & Eijkhout, 2002) and BeBop (Vuduc, Yelick, Demmel, 2001-3)

# Discussion

## ◆ Extensions

### ■ Fit in framework for SANS

- Timing facilities (Autopilot at UIUC, etc.)
- Algorithmic description metadata on XML
- History database, agents for update of rules

### ■ Extend to more parts of MD simulation protocol:

- Multiple time stepping integrators
- Parallelism, clusters, grids
- Lower level parameters (matrix-vector block size, etc.)? Or use ATLAS!

## ◆ For further reference:

- <http://www.nd.edu/~lcls/mdsimaid>
- <http://www.nd.edu/~lcls/protomol>
- <http://www.nd.edu/~izaguirr>

# Acknowledgements



- ◆ NSF Biocomplexity grant IBN-0083653
- ◆ NSF ACI CAREER Award ACI-0135195
- ◆ NSF REU grant
- ◆ Prof. Ricardo Vilalta and students, Univ. of Houston, for work on inductive learning