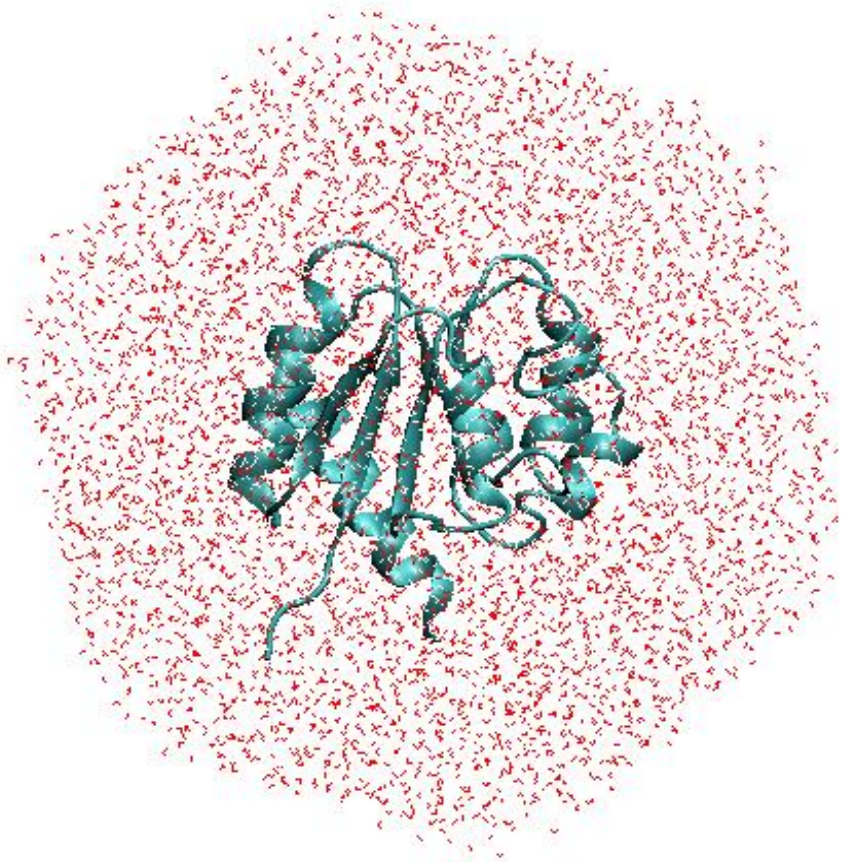
Protein molecular dynamics on OSG using CHARMM



Ana Damjanovic, Tim Miller, Bernard Brooks (NIH)

Petar Maksimovic (JHU)

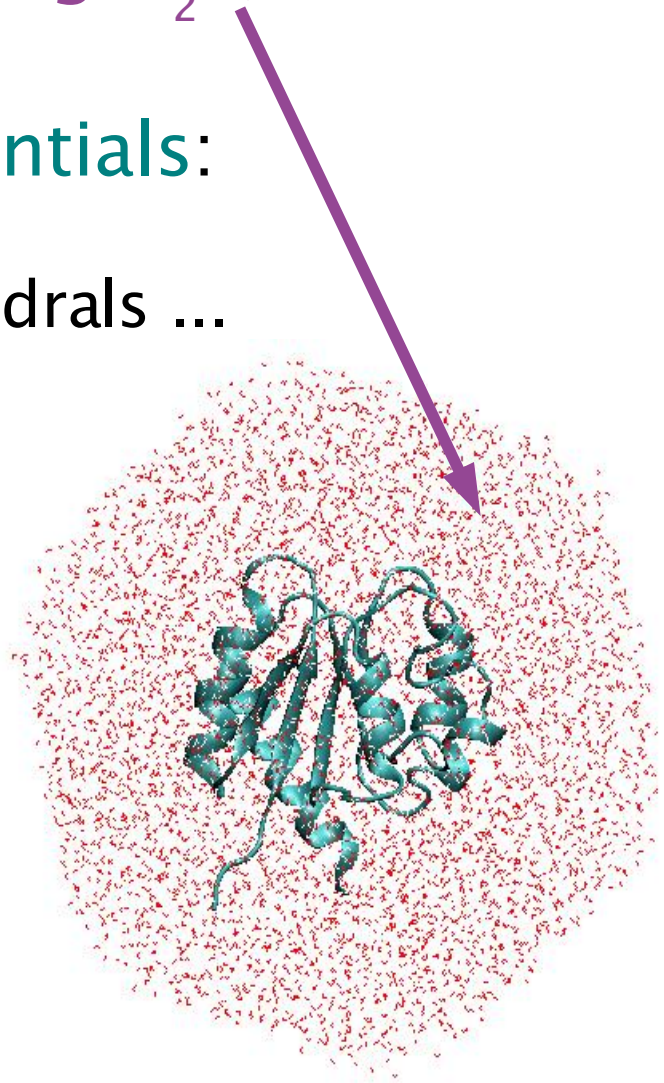
Wensheng Deng, Torre Wenaus (BNL), Frank Wuerthwein (UCSD)

Why Molecular Dynamics (MD)

- MD bridges gap from theory to experiment
- Used in a large number of situations
- Holy Grail: protein folding
  (sequence <=> 3d structure <=> function)
- Another hot topic: conformational changes
  - when part of protein in position A, does one thing
  - when it moves to position B, does another

Molecular Dynamics Simulations

- Atoms described explicitly (including H_20)
- Interaction through empirical potentials:
  - electrostatic, van der Waals.
  - bond vibrations, angle bending, dihedrals ...
- Time evolution via integration of Newton's equation of motion.
- **timestep is 1-2 fs.**
  (can do motions up to ~ 100ns)



Why CHARMM

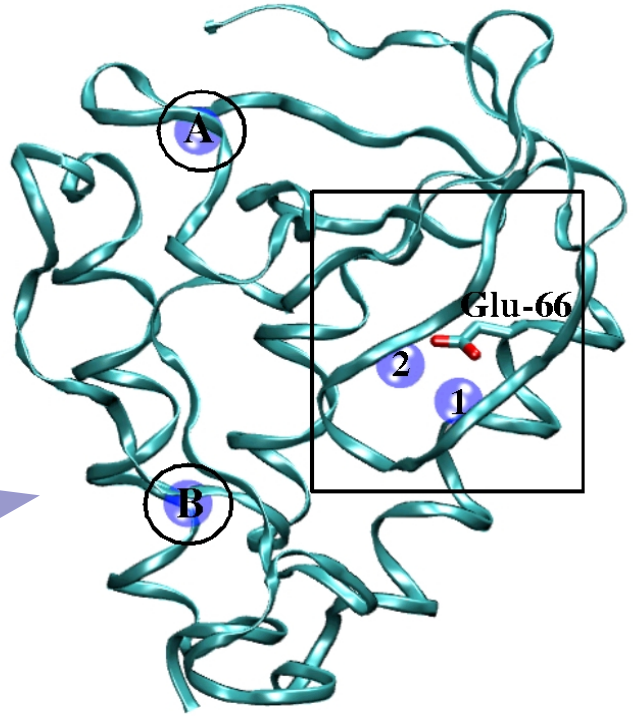
- Oldest and most widely used program for protein Molecular Dynamics
  - generic and flexible framework for MD, energy minimization, and analysis
  - http://www.charmm.org
  - a lot of expertise at NIH

\

- But: with straightforward changes, this approach can work with other MD suites (AMBER, NAMD, GROMACS...)

Why GRID

- In the past, one had to use a supercomputer or a cluster with fast connections
- Today, PCs are getting more powerful, each can simulate a whole small protein



- Some problems are statistical in nature (protein folding, conformational changes), so naturally *parallelize*

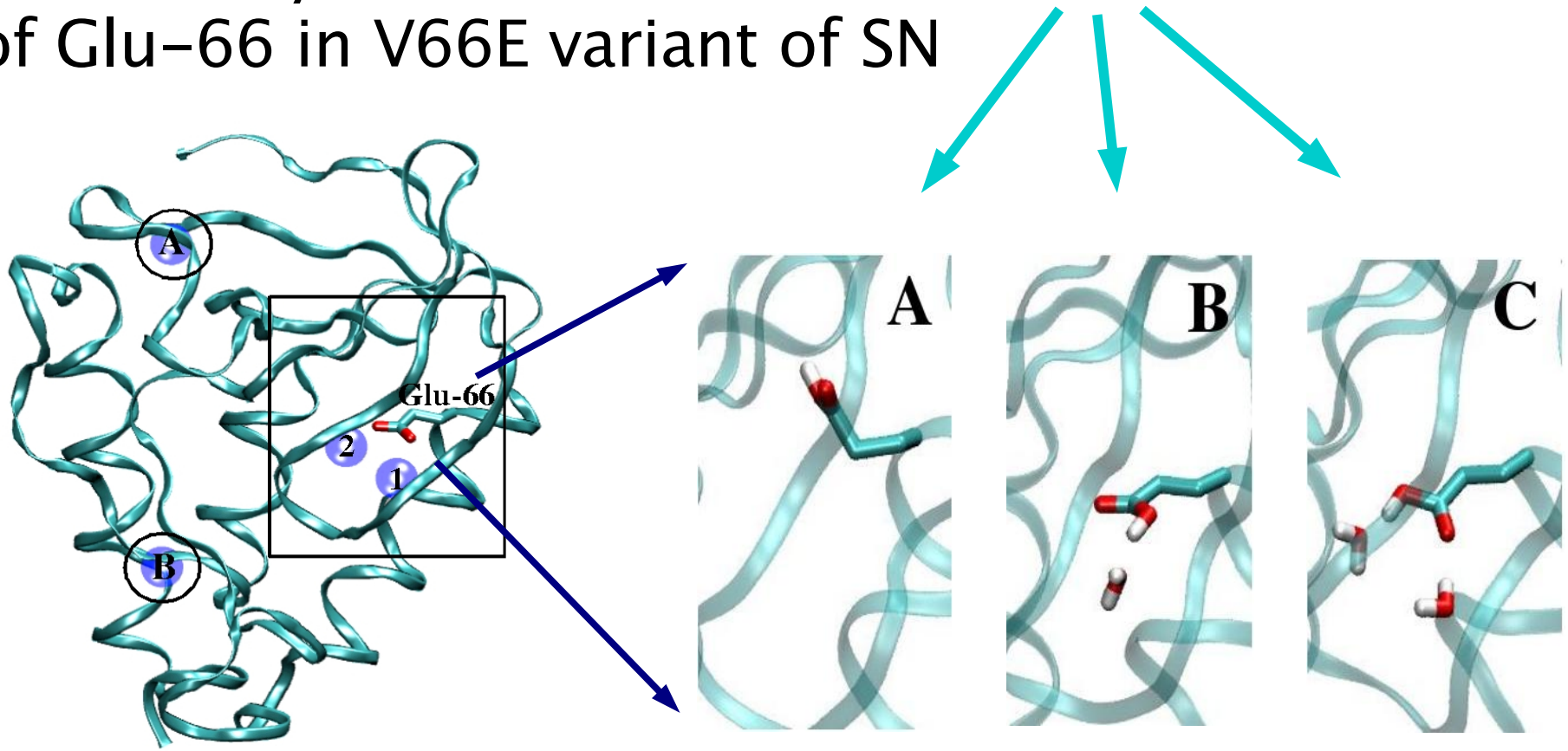
Why GRID (2)

Run in parallel, to

- get statistically meaningful results (experiments deal with ansambles so MD must too)
- increase chances of observing events on ~ 1μs timescale (protein folding, conformational changes)
- simulate similar proteins (comparative study)
- study effect of slightly different initial conditions

Example: water penetration in staphylococcal nuclease (SN)

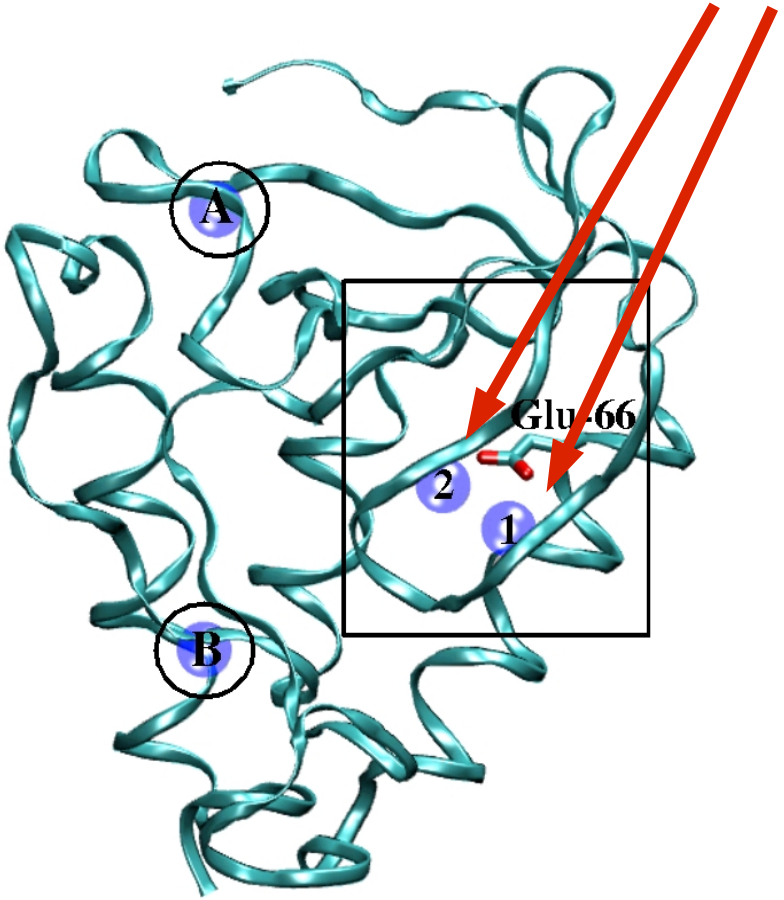
- Preliminary results: three conformations of Glu-66 in V66E variant of SN



- What is the population of each conformation?

Use as a testbed for a faster MD simulation technology!

- Sensitive to initial conditions? – start with or without the two internal water molecules

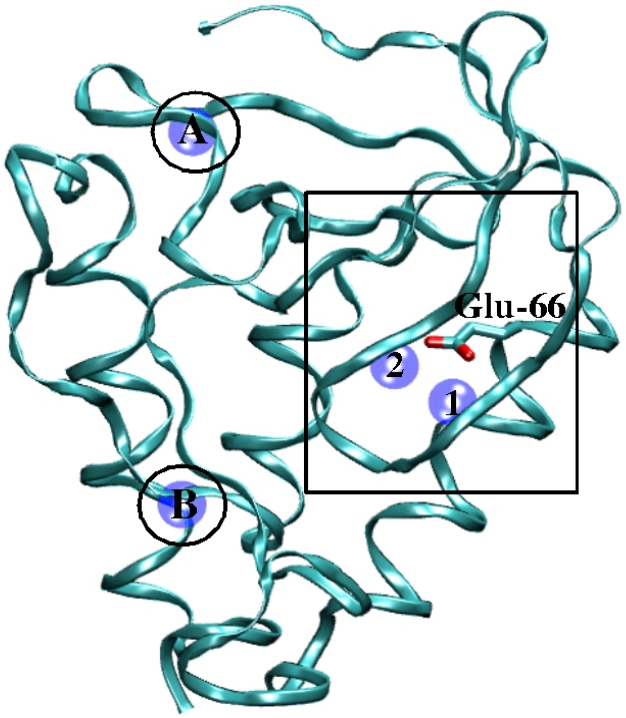


- Also testing a new method for conformational search: Self Guided Langevin Dynamics (SGLD)

- SGLD nudges particles along their average momentum -- things happen much more quckly

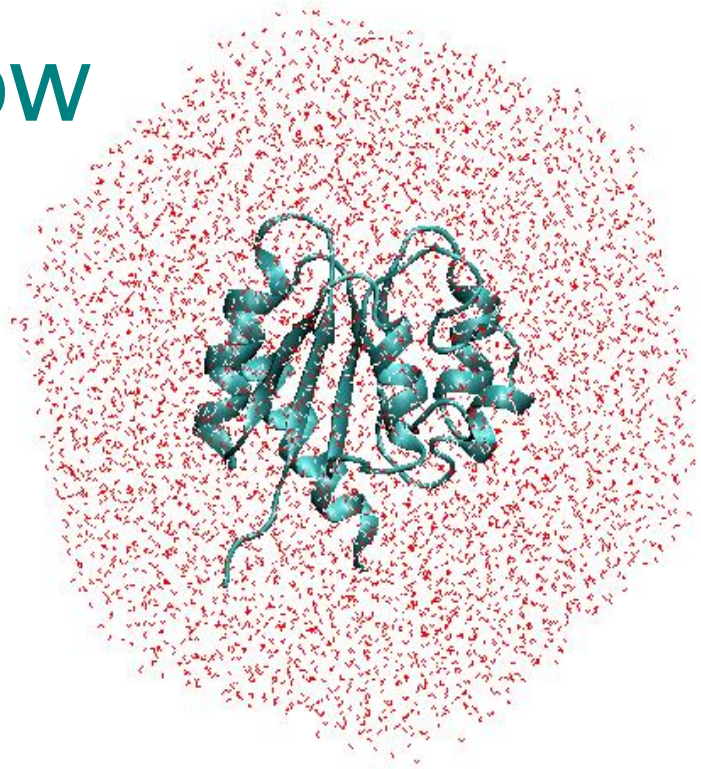
- But, **is SGLD == MD ???**

What we have simulated



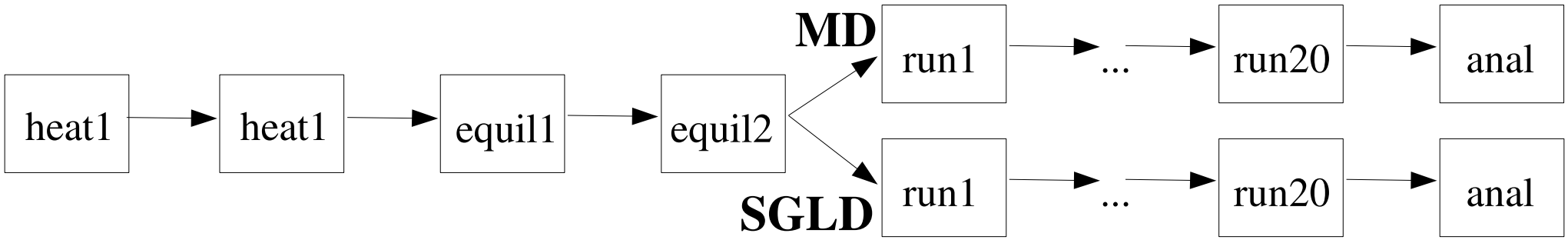
- SN with water molecules 1 and 2
  - regular MD (40 simulations)
  - SGLD (40 simulations)
- SN without water molecules 1 and 2
  - regular MD (40 simulations)
  - SGLD (40 simulations)

CHARMM workflow



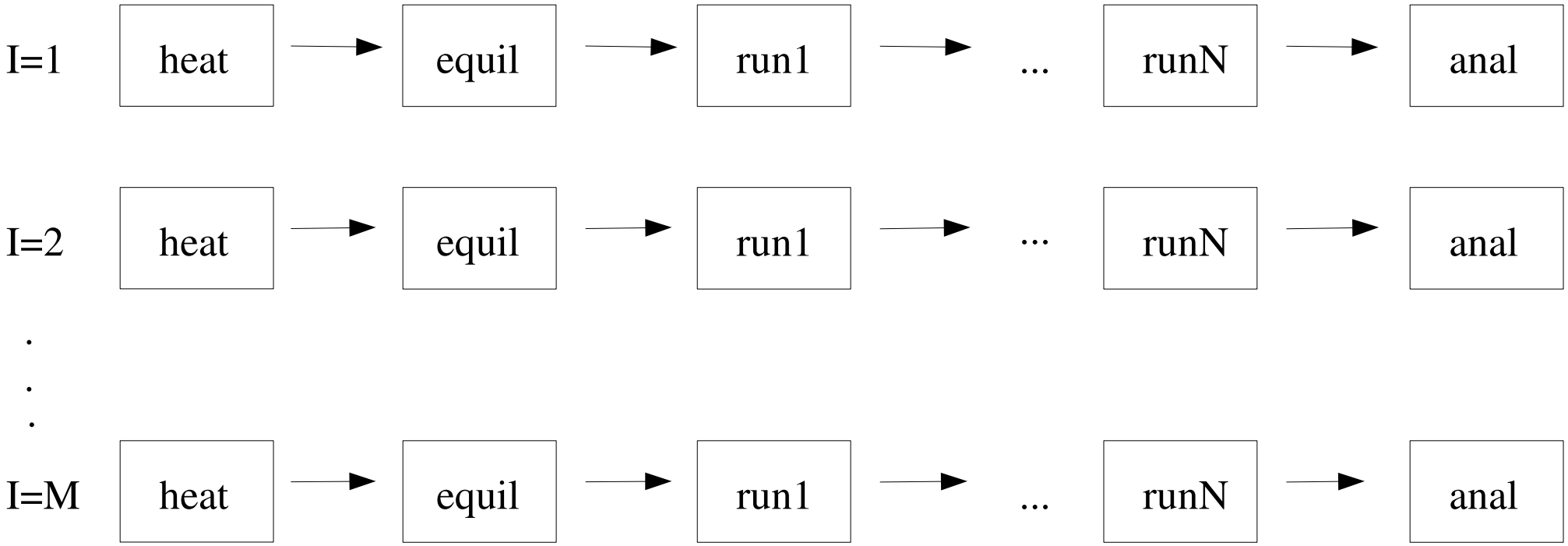
- 2k protein atoms
  + 16k water atoms
  = 18k atoms

- Grid queues up to 72 hrs: divide into a sequence of jobs (each 50ps long = 50k timesteps)



`Thread and Wave' model

- Each independent starting point is a thread
- Each step in the analysis is a wave
- A MD can have several threads with many waves



CHARMM job management

- We use PanDA to submit jobs
- Run our own AutoPilot, though
- Initially, each wave in each thread would submit the next wave in that thread.
  Problem: proved to be unreliable
- Solution: manage all jobs by a daemon (rcdaemon) from a single machine where all the output and meta-files are stored (helen.pha.jhu.edu)

CHARMM job specification

```
# section 1: job specification
# here we specify the different scripts we want to use
JOB heat USE heat.inp
JOB heat2 USE heat2.inp
JOB eq USE eq.inp
JOB md USE run.inp
JOB analp USE anal_phipsi.inp
JOB analw USE anal_water.inp

# section 2: threads
NTHREADS 30
THREADPARAMS 'I=[threadid]'

# section 3: Default input requirement and output production
REQDEFAULT [PREVWAVE .res]
PRODEFAULT [.res,.trj,.pdb,.log]

# section 3: order
# The ONLY keywords are
# BRANCH ... REJOIN
# TEST ... ENDTEST
# LABEL, RESULT, ELSE
BEGIN
heat REQUIRES [NONE NONE]
 heat2
 eq*2
 BRANCH A APPENDPARAMS G=0
   md*20
   analp REQUIRES [ALLDONE .res,.trj] PRODUCES [.dih1,.dih2]
   analw REQUIRES [ALLDONE .res,.trj] PRODUCES [.wat]
 BRANCH B APPENDPARAMS G=1
   md*20
   analp REQUIRES [ALLDONE .res,.trj] PRODUCES [.dih1,.dih2]
   analw REQUIRES [ALLDONE .res,.trj] PRODUCES [.wat]
END
```

- User supplies a job description file
- CHARMM scripts used
- Define threads
- Input/Output
- Define branches
- Order waves

Problems and Solutions

- Problem: globus-url-copy sometimes unavailable at grid sites (different setup)
- Solution: distribute with CHARMM exe!
- Problem: globus-url-copy rarely corrupts thajectories and the restart file (what the next wave needs). Happened 25 times. A big deal because this invalidates all subsequent waves.
- Solution (working on it): compare MD5 sums before and after copying

More Problems and Solutions

- Problem: the global job submission daemon is a single point of failure! The machine got rebooted, the daemon did not start, and we lost about a week or running since the new waves were not being submitted.
- Solution: (working on it) automatic startup on boot (duh), better monitoring.
- But, hey, it's only 1 week out of 6 months!
- A testament on how smooth this was: we got so lazy that we didn't even check!

Unfounded Worries and Fears

- Worry: the configuration of various grid sites was different and evolving in time
- Resolution: Torre's AutoPilot did a spectacular job of submitting only to the right clusters. Had only a handful of failures due to this.
- Worry: a typical MD simulation wants to run continuously for a long time. So possible wait times worried us a lot. (Add directly to total time.)
- Resolution: actual average wait time was only about 5 mins per wave!

CPU time used

- Status: clean-up, waiting for ~ 3 threads to finish
- Each setup and MD wave took ~ 12hrs
- 4 setup + 2*20 MD waves (1->2 branching)
- Ran two 30-thread simulations of 2*20 waves (each used 15842.40 hours of CPU)
- Ran two 10-thread simulations of 2*20 waves (each used 5280.8 hours of CPU)
- Total hours ~ 42246.4 of CPU

Data I/O (to/from Grid)

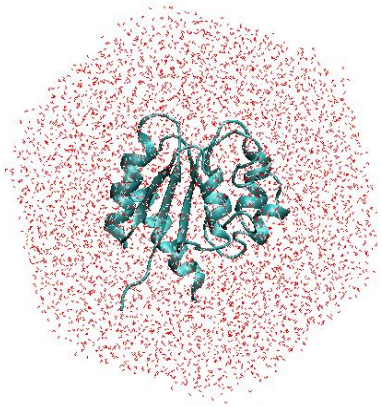
- CHARMM jobs need to save their state at the trajectory repository
- Each job fetched ~ 5 MB from repository (4.3 restart file + 0.7 MB executable and other junk)
- Each job put back 115 MB (trajectories + restart file)
- Total output of all jobs ~ 30 GB (peanuts compared to HEP!)

Conclusions

- Very positive experience so far:
  - failure modes understood
  - wait times short
- With automatic resubmission of jobs by the daemon, very hands off and trouble-free
- Useful immediately for small groups without substantial computing resources
- For protein folding/conformational searches, need to be able to `branch' from one initial configuration (working on it)

BACKUP SLIDES

Structure -> Dynamics -> Function



Timescales of protein motion:

femto-pico: bond vibrations, angle bending

pico-nano: loop motions, surface sidechains, water penetration

nano-micro: folding in small peptides, helix-coil transitions

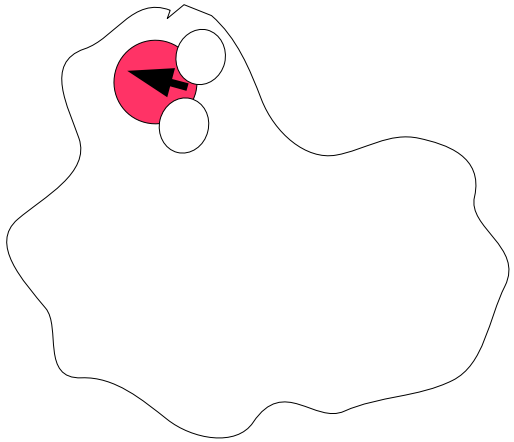
micro-seconds: conformational rearrangements, protein folding, catalysis

Physical complexity:
various shapes, sizes,
bound non-protein molecules

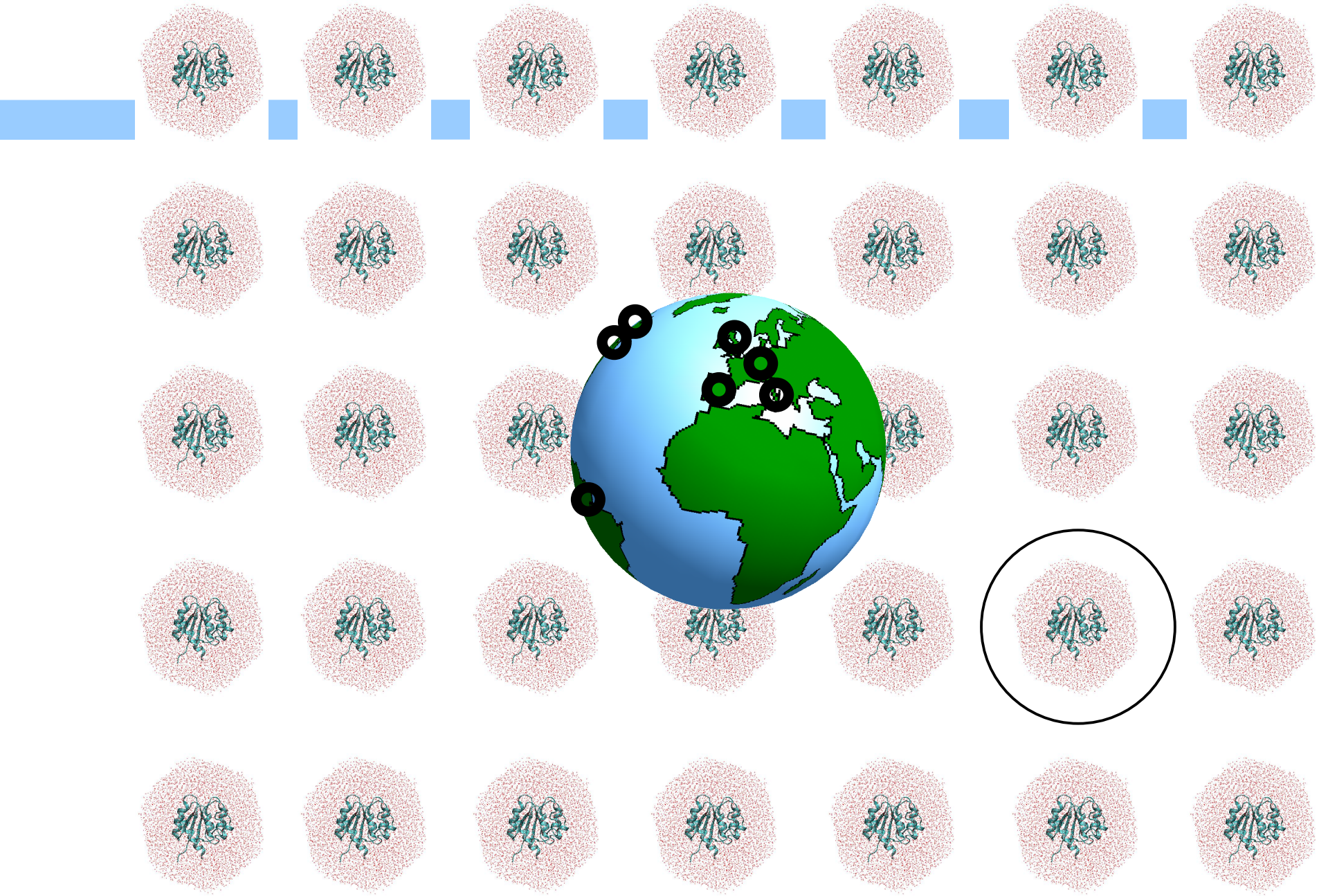
Environment: water,
membrane, pH, ions,
gases, small molecules,
macromolecules

Achieve meaningful statistics

Water penetration into proteins



* Proteins live in water, and water lives in proteins.
* Water often has important functional roles: proton transfer, enzymatic catalysis
* Numbers and positions of water molecules in x-ray structures of proteins are often ill resolved
* Approach – use MD simulations to observe penetration of water into proteins



With a software that can babysit the jobs while we sleep ...

software managing your jobs

Input

Output

Implementation of CHARMM on the OSG

What do we need to have (requirements)?

- A way to set up various run parameters.
- Ability to submit and track many jobs.
- Easy access to input and output files from the grid.

What application specific challenges must we deal with?

- The framework must allow for maximum flexibility since CHARMM can do many things.
- Efficient handling of many input and output files.
- Figuring out queue lengths and resource limitations and tailoring jobs to them.
- Restarting failed jobs.

Solution: Use PanDA and a custom set of management scripts

The Scheduler Interface

- We use the PanDA front end.
- We also use TestPilot and run our own pilot scheduler for maximum control.
- Users can track jobs via a Web interface.

Job definition from the researcher's point of view

The following steps are required to set up and submit a job:

1. Obtain CHARMM and the PandaForCharmm software.
2. Create the various input scripts needed for the job.
3. Pack these and other necessary files into a tar.gz to be extracted on the execution host.
4. Modify parameters in charmmJob.sh, example:

Example thread and wave parameters:

```
export tarball=ana2.tar.gz
export exe=c33a2-lrg.one
export jobname=ana3
export threadparams="I=[jobid]"
export inpscripts="heat=heat.inp,equ=eq.inp,md=run.inp"
export threaddef="heat,equ*2,md*2"
```

5. Run charmmJob.sh
6. Watch your jobs run!

The Web Interface (constructed by Torre Wenaus)

Configuration

Update

Panda monitor

Quick guide, twiki

Jobs - search
Recent running, activated, waiting, assigned, defined, finished, failed jobs
Select analysis, production, test jobs

Quick search
Job
Dataset
Task
File

Summaries
Blocks: days
Errors: days
Nodes: days
Daily usage

Tasks - search
Generic Task Req
EvGen Task Req
CTBsim Task Req
Task list
Task browser

Datasets - search
Dataset browser
New datasets
Panda subscriptions
All subscriptions

Sites - see all
BNL BU IU OU SLAC UC UMICH UTA LCG NG

Applications
CHARMM

Dashboards: Production DDM Sites & Grids Analysis Physics data Task definition Usage & Quotas TestPilot Plots ArdaDash

CHARMM job overview

Recent CHARMM job submitters: Ana Damjanovic Benjamin Timothy Allen Miller

Recent CHARMM pilots: all submitted (8) scheduled (0) running (4) finished failed aborted

Queues used by CHARMM

Recent CHARMM jobs (last 3 days): ana1 ana2 ana3

All jobs:

	jobs	active	run	finish	fail
Totals:	*50*	*0*	*42*	*8*	*0*

Jobname ana1:

Wave	jobs	active	run	finish	fail	PandaIDs
Totals:	*23*	*0*	*22*	*1*	*0*	
4	1	0	0	1	0	1049437
3	4	0	4	0	0	1008684 1009260 1009327 1010430
2	3	0	3	0	0	998138 999868 1002365
1	12	0	12	0	0	989027 992341 992342 993107 993108 995119 995421 996927 997229 998134 998137 998139
0	3	0	3	0	0	1039253 1039254 1039255

Jobname ana2:

Wave	jobs	active	run	finish	fail	PandaIDs
Totals:	*10*	*0*	*10*	*0*	*0*	
0	10	0	10	0	0	1055685 1055686 1055687 1055688 1055689 1055690 1055691 1055692 1055693 1055694

Jobname ana3:

Wave	jobs	active	run	finish	fail	PandaIDs
Totals:	*17*	*0*	*10*	*7*	*0*	
2	3	0	3	0	0	1067943 1068544 1069746
1	4	0	1	3	0	1069745 1061206 1061207 1061208
0	10	0	6	4	0	1056295 1056296 1056297 1056298 1056299 1056303 1056300 1056301 1056302 1056304

Where we are and where we want to go

Currently:

- Basic set up of the thread and wave model is completed and we've tested our own scripts extensively.
- We have started production runs with fifty threads of twelve waves.
- 100K step jobs are taking about 1 day to finish. This means we can simulate 1 ns per thread in 180 to 300 hours of wall time!

Future Directions

- Ability to introduce "branches" in the script sequence, to allow, for example, extra analysis of "interesting" structures.
- Better tracking of in-progress jobs along with failure detection and possible correction.
- A graphical front-end for job definition and submission.
- Gaining a better understanding of various sites and queues so we can better match jobs to resources.