Exacycle:
1 Billion Core-Hours of Computational Capacity for Researchers

David Konerding, Software Engineer
SARA HPC Cloud Computing Day 2011
http://research.google.com/university/exacycle_program.html

Google Research Blog

1 billion core-hours of computational capacity for researchers

Thursday, April 07, 2011 at 4/07/2011 03:00:00 PM

Posted by Dan Belov, Principal Engineer and David Konerding, Software Engineer

We're pleased to announce a new academic research grant program: Google Exacycle for Visiting Faculty. Through this program, we'll award up to 10 qualified researchers with at least 100 million core-hours each, for a total of 1 billion core-hours. The program is focused on large-scale, CPU-bound batch computations in research areas such as biomedicine, energy, finance, entertainment, and agriculture, amongst others. For example, projects developing large-scale genomic search and alignment, massively scaled Monte Carlo simulations, and sky survey image analysis could be an ideal fit.

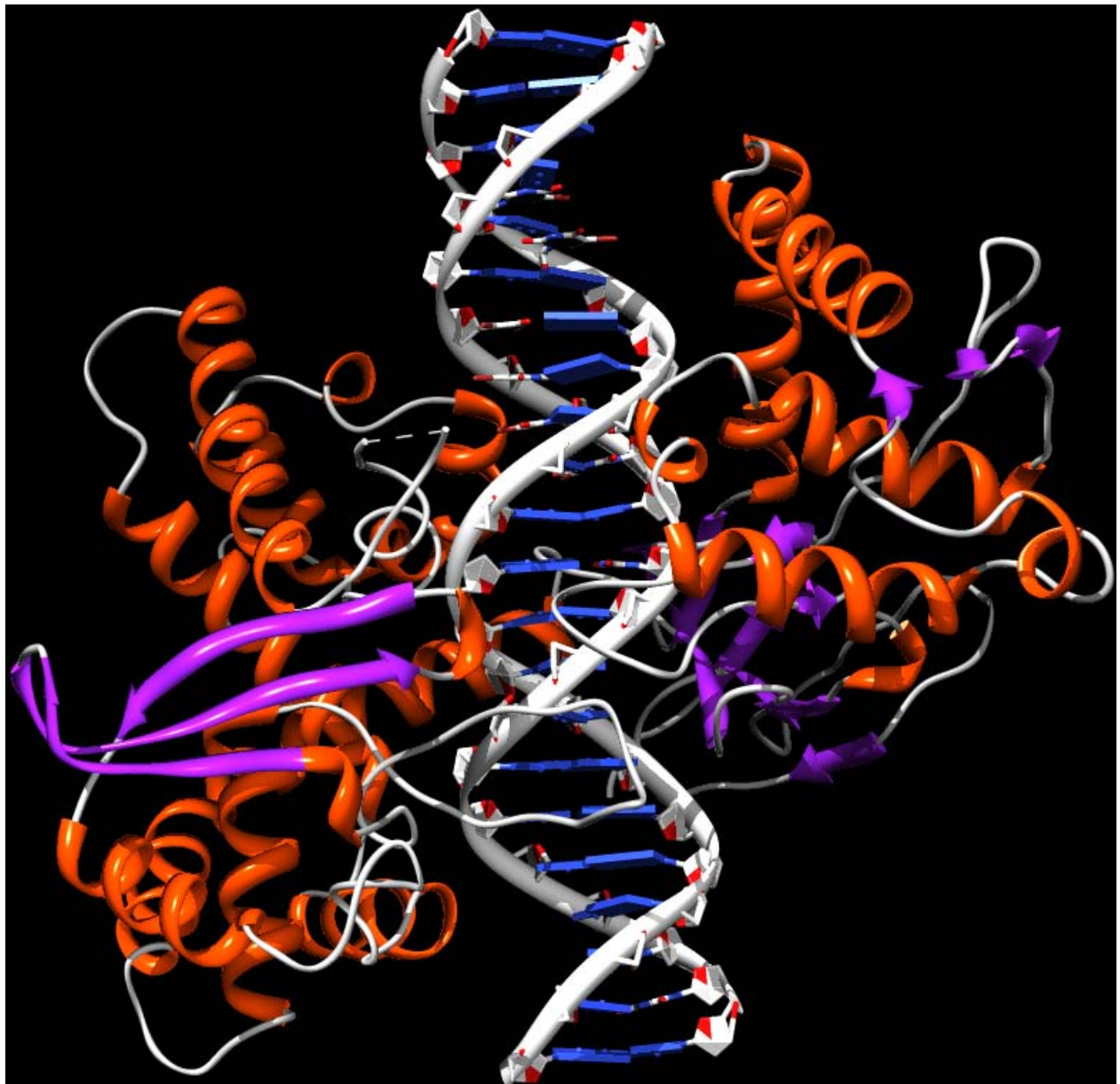
Exacycle for Visiting Faculty expands upon our current efforts through University Relations to stimulate advances in science and engineering research, and awardees will participate through the Visiting Faculty Program. We invite full-time faculty members from universities worldwide to apply. All grantees, including those outside of the U.S., will work on-site at specific Google offices in the U.S. or abroad. The exact Google office location will be determined at the time of project selection.

We are excited to accept proposals starting today. The application deadline is 11:59 p.m. PST May 31, 2011. Applicants are encouraged to send in their proposals early as awards will be granted starting in June.

More information and details on how to apply for a Google Exacycle for Visiting Faculty grant can be found on the Google Exacycle for Visiting Faculty website.

Background: Who am I and why did I build Exacycle?

My background is in Protein and Nucleic Acid Structure and Function (molecular simulations of DNA- and RNA- binding proteins)

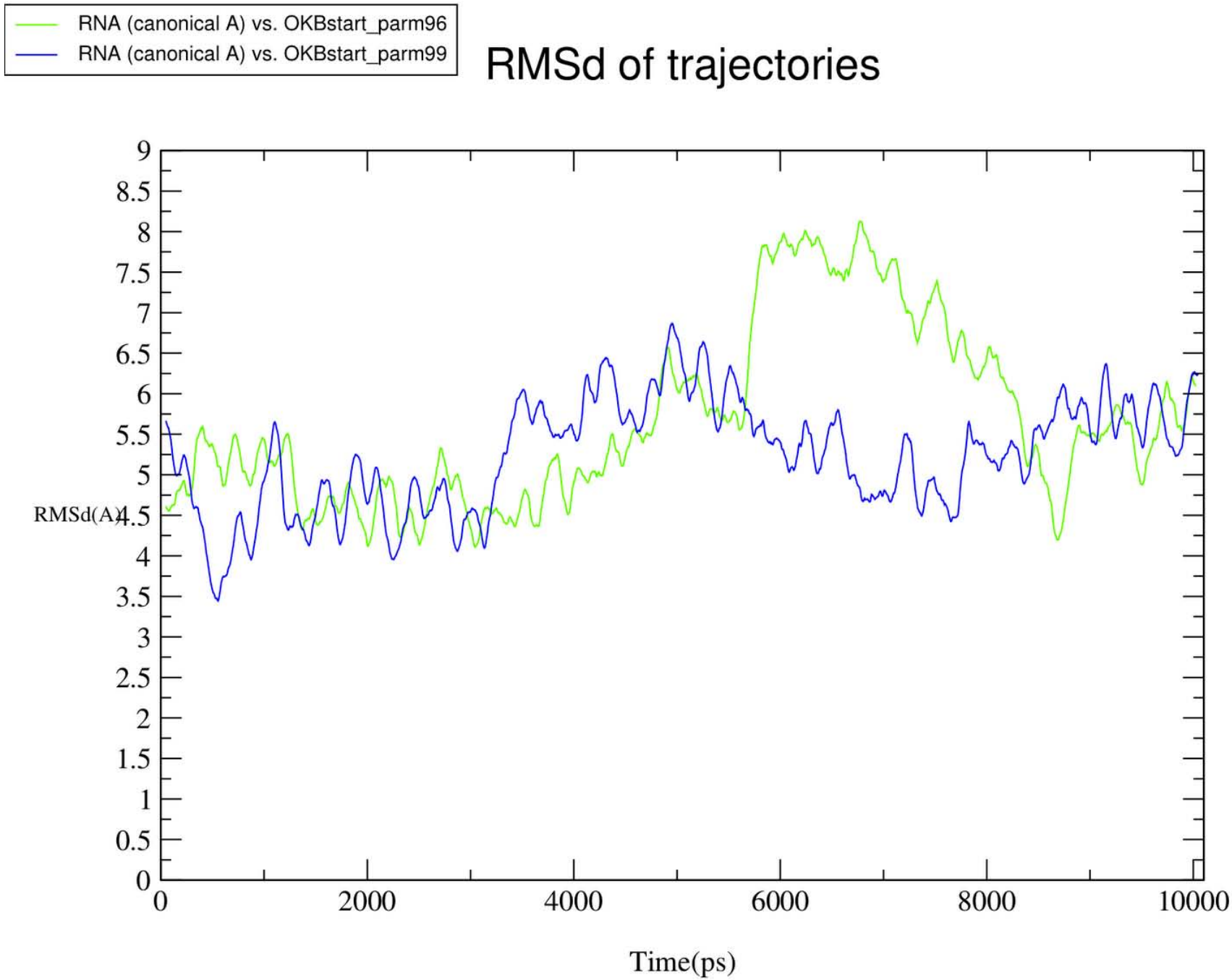


Background: Who am I and why did I build Exacycle?

Initially, I ran simulations (lasting a year, using 128 PEs, with 64-96x speedup) on Cray T3E. Well-balanced architecture.

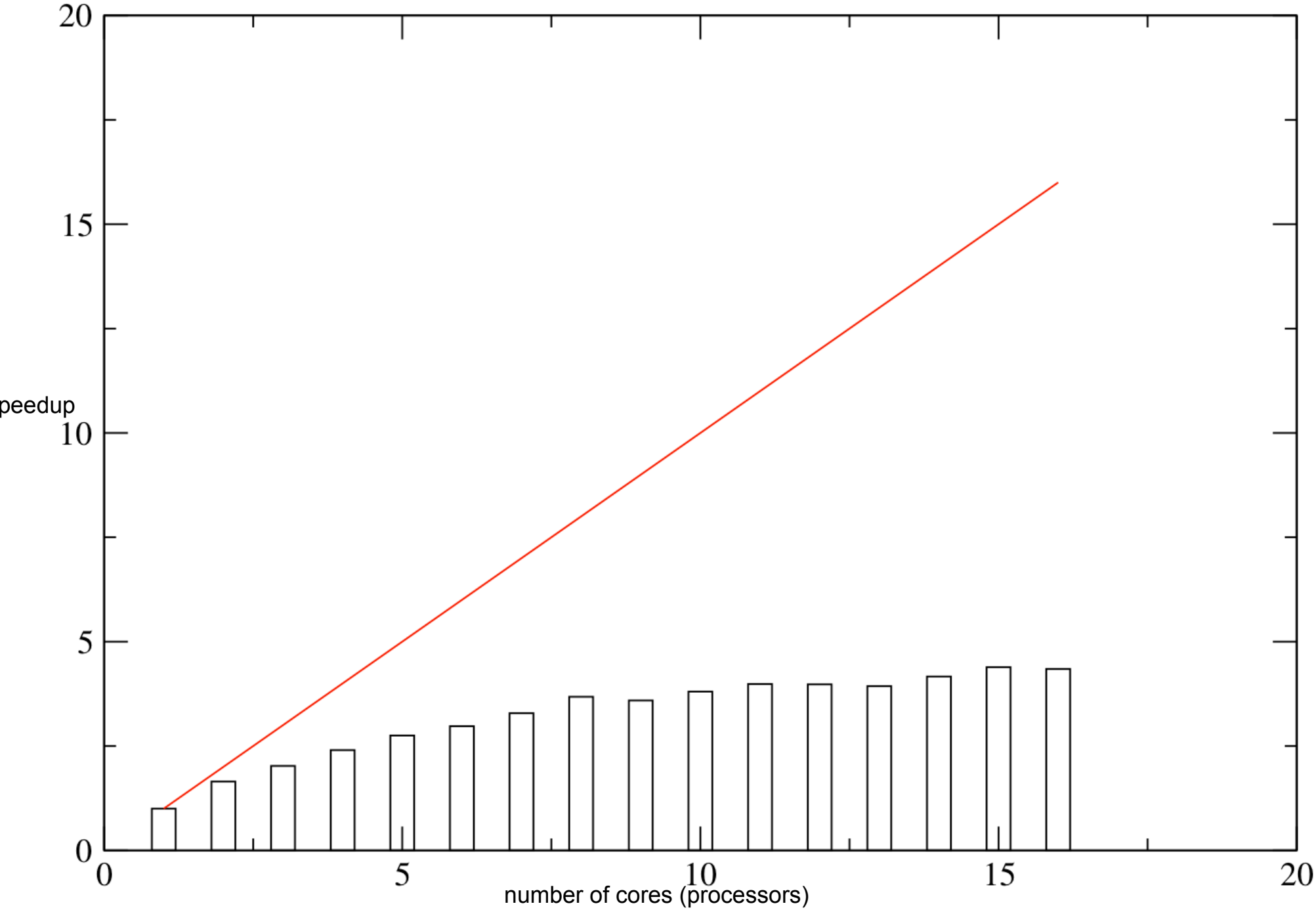
But as computers got faster and cheaper, I started building my own clusters (without the fast interconnect)

A year of Cray T3E simulations to test a force field change...



I started building my own clusters:

Cheap Linux Clusters running AMBER MPI (circa 2001) with 100BT didn't scale. AMBER MPI (circa 2005) with 1G didn't scale...



I switched to running many independent simulations and pooling results, which scales better (shared-nothing architecture). Well, at least shared-very-little.

By running many MD simulations in parallel (embarrassingly parallel), I could generate far more data (terabytes) for much less money.

Unfortunately, this just traded off one IO bound problem for another: now I had terabytes of trajectory data and trying to analyze it would kill the NFS server

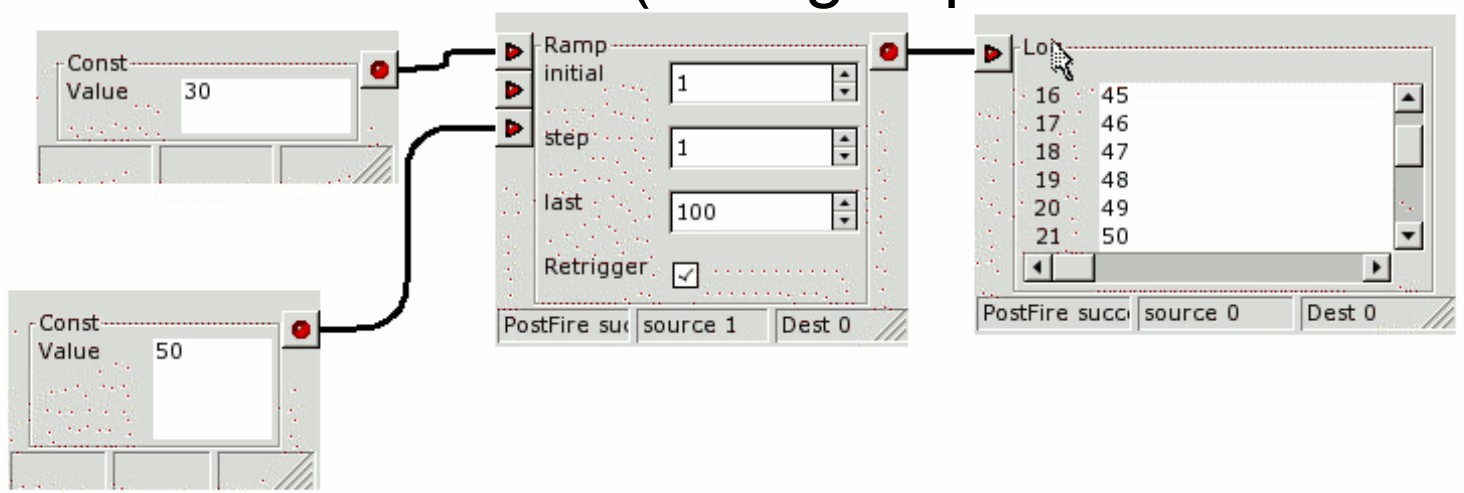
After observing this phenomenom across multiple discplines, I concluded: scientists can't be productive using their existing computing systems

- efficient, parallel data analysis of modest data sets is very hard with commodity hardware and software
- typical solutions involve very expensive hardware and software
- There are some inexpensive (or free) software systems that address this, but setup and tuning are hard.

=

Right Before Google (2006):

- I was a developer at LBL, working on pyGlobus and pyGridware
- Scientists needed higher-level GUI to wrap multiple applications, and application loops/ranges that execute on the Grid
- ViCE: ViCE: Visual Computing Environment for Scientific Workflows on the Grid (a long deprecated GTK GUI)



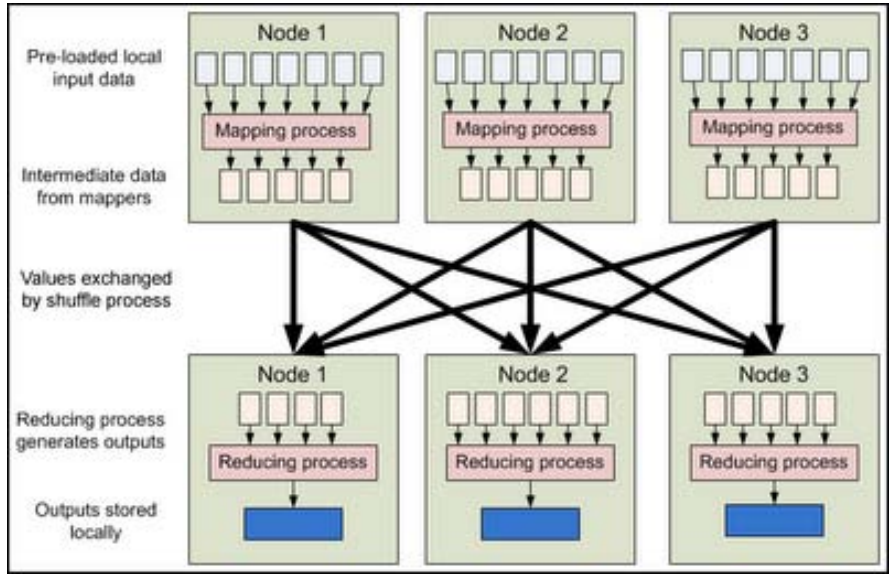
  - All computation on a server; BPEL-WS based (in theory)
  - Similar tools (Taverna etc) used today are far better, integrate well (??) with cloud & grid.

I came to Google to explore using MapReduce, GFS, and BigTable for scientific data generation and analysis

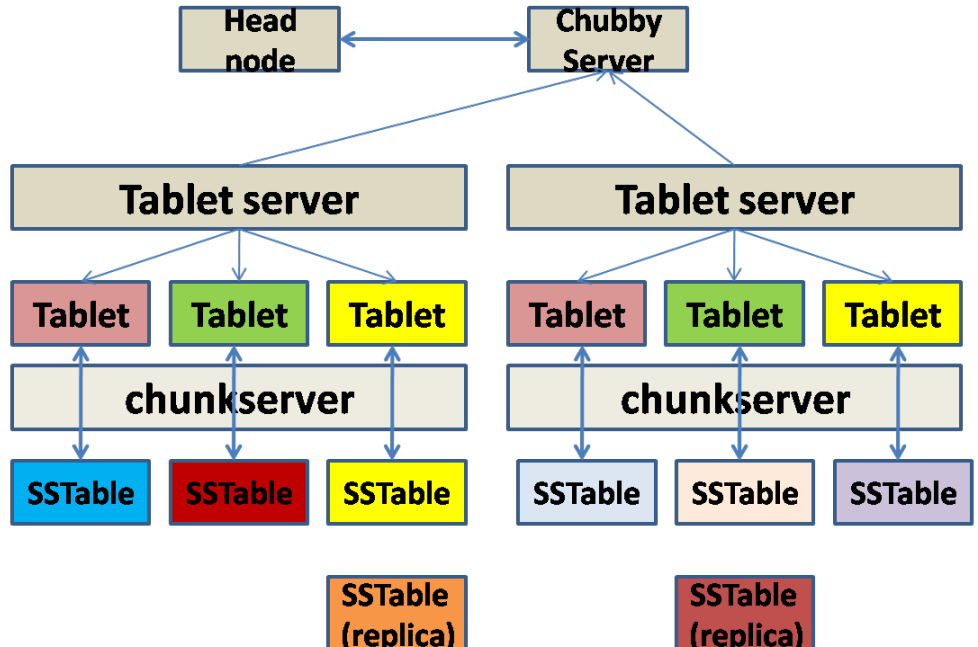
This has worked out well, but...

GFS

MapReduce



BigTable

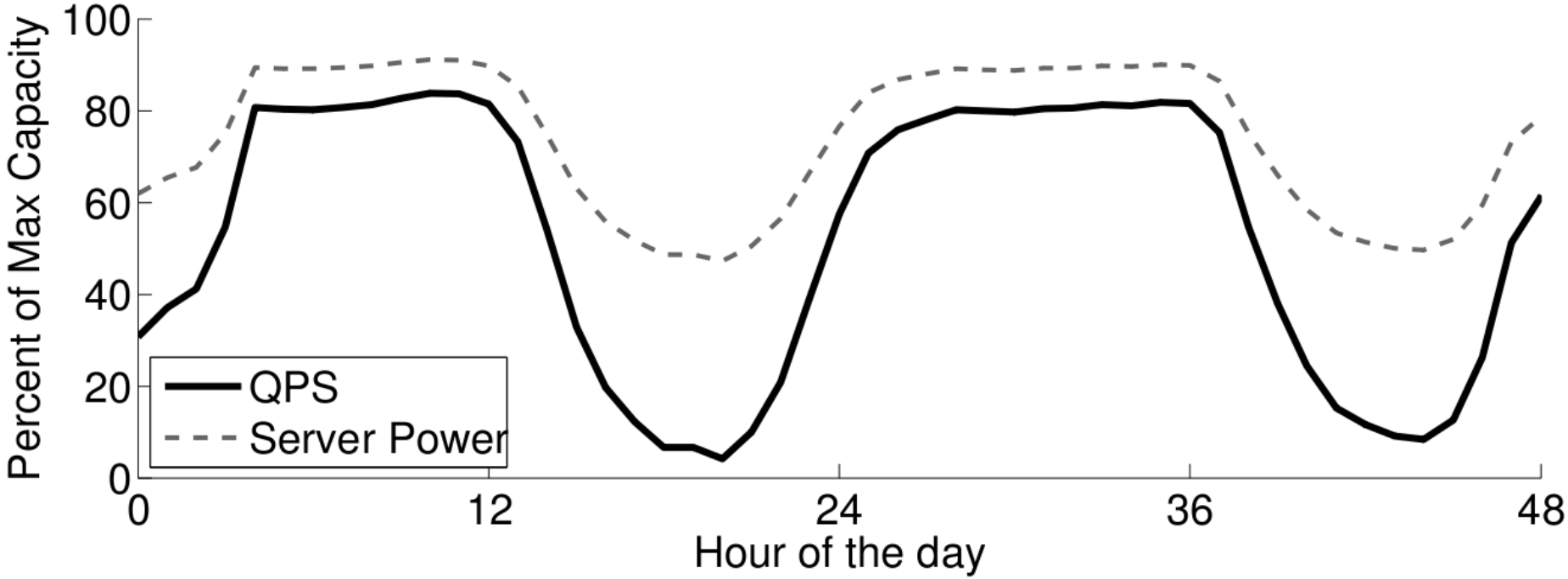


Collaborating with external scientists is tough

- Became a bottleneck since external scientists don't have access to Google systems
- Googlers should replace their daily job with a shell script every 6 months or so....

So I built exacycle to enable scientists to use Google resources to run HPC cloud applications.

The dirurnal cycle defines resource availability



System capacity is typically peak + safety factor for overload

Constrained capacity at peak, significant off-peak capacity

Geo-targeting keeps most user traffic to Google frontends local to servers in the region

Troughs "move" globally through datacenters during the day

Exacycle: basic description

Essentially a massive "Map" phase of a larger MapReduce pipeline.

- Designed to run 100,000 to 1M concurrent work units (mappers)

- You must be willing to tradeoff flexibility for scale
  - break your job into millions of work units
  - each work unit cannot exceed 1 core x 1G RAM x 1hour
  - no interprocess or intermachine communication

- time-to-completion is days, weeks, or months (latency is high, but so is throughput)

- Applications must be compiled using Native Client, a security sandbox originally designed for secure browser appls

- Visiting Faculty have access to GFS and BigTable for storage, MapReduce for input prep and data analysis

Exacycle: command line, script, and API interactions

- you need to write a "driver" script to submit work
  - For simple jobs that contain no complex dependencies, can simply submit a command line job:

```
exacycle submit \
    --jobname=hello_world \
    --binary="%(binarydir)s/hello_world" \
    --num_total_work_units=1000000 \
    --arguments="--tasknum %(tasknum)d"

100000 work units submitted.
```

- your driver needs to periodically check that work units are completing successfully and submit more works
  - For simple jobs, command line monitoring:

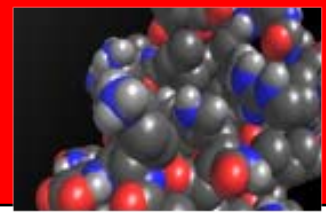
```
exacycle monitorjob \
   --jobname hello_world

Total work units: 1000000
Total finished:   549260
Total working:    457060
```

- REST API could be easily integrated with Grid-style master schedulers, Globus GRAM, Condor DAGman, etc.


Folding@home
distributed computing

Folding@home
distributed computing

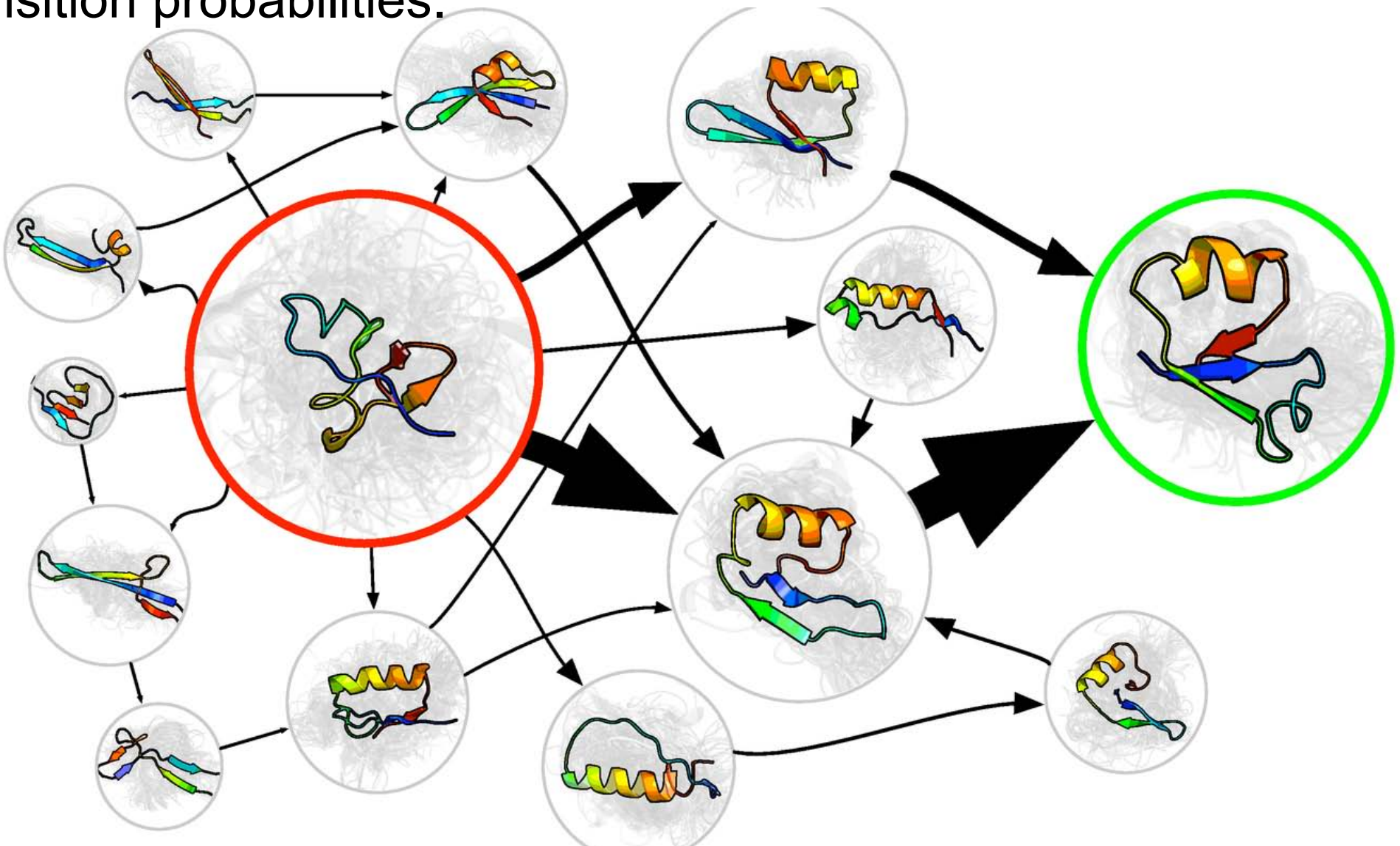




You can help scientists studying these diseases by simply running a piece of software.

Folding@home is a distributed computing project -- people from throughout the world download and run software to band together to make one of the largest supercomputers in the world. Every computer takes the project closer to our goals. Folding@home uses novel computational methods coupled to distributed computing, to simulate problems millions of times more challenging than previously achieved.

Protein (mis)folding is linked to disease, such as Alzheimer's, ALS, Huntington's, Parkinson's disease, and many Cancers.
Moreover, when proteins do not fold correctly (i.e. "misfold"), there can be serious consequences, including many well known diseases, such as Alzheimer's, Mad Cow (BSE), CJD, ALS, Huntington's, Parkinson's disease, and many Cancers and cancer-related syndromes.

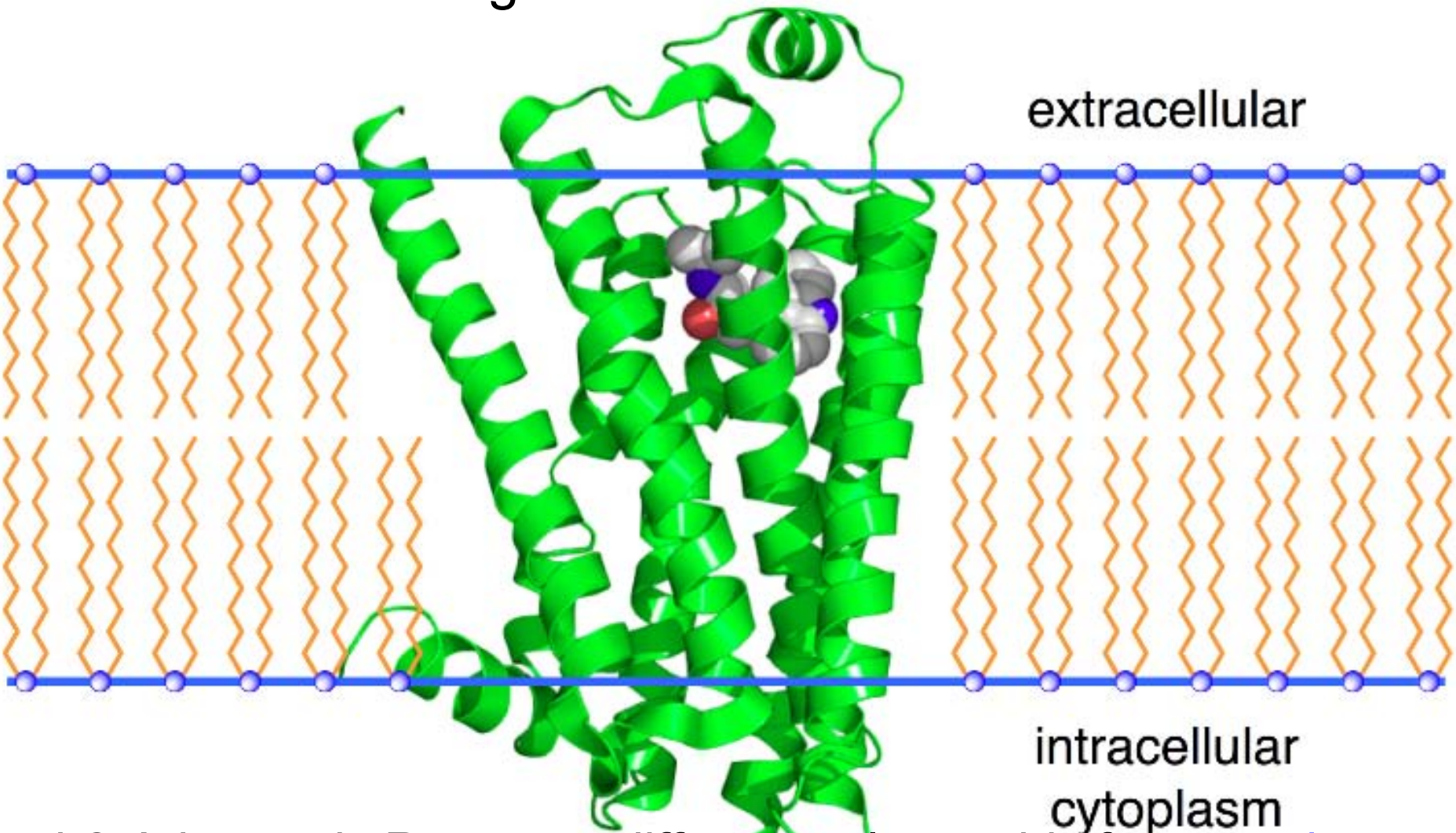
Protein folding kinetics can be modelled as transitions between substates. This graph shows an estimate of the graph and its transition probabilities.



Molecular Simulation of ab Initio Protein Folding for a Millisecond Folder NTL9(1-39)
Vincent A. Voelz, Gregory R. Bowman, Kyle Beauchamp, and Vijay S. Pande
Departments of Chemistry and Structural Biology, and Biophysics Program, Stanford University, Stanford, California 94305

Folding@Home@Google

G-protein Coupled Receptors are cell membrane-bound proteins useds for signal transduction



b2-Adrenergic Receptor: different polymorphic forms, point mutations, and/or downregulation of this gene are associated with nocturnal asthma, obesity and type 2 diabetes.

Folding at Home is running on Exacycle as the initial scientific user

Kai Kohlhoff, Visiting Faculty from Stanford (now a fulltime Research Scientist at Google, is simulating a number of medically important protein structures from the G-Protein Coupled Receptor family.

G-protein coupled receptors (GPCRs) are proteins that are important for many biochemical processes in the human body. They act as signal transducer in the membrane of cells.

GPCRs are involved in many human diseases and about half the volume of the prescription drug market consists of drugs targeting GPCRs. Size of the GPCR market: well over $25 Billion Revenue/year.

To avoid side effects, improve the effectiveness of existing drugs and develop entirely new drugs, knowing the mode of action of GPCRs is essential.

3 other projects have signed on as Visiting Faculty (2 more coming)

Current simulations are based on medically important protein receptors

Our simulations include:

- beta-2 adrenergic receptor:
  - respiration, flight-or-fight response, glucose supply
  - major target for asthma medications

- beta-2 adrenergic receptor:
  - cardiac output
  - major target heart disease medications

- Adenosine A2A receptor:
  - vasodilation of cornoary arteries
  - major target for abnormal heart patterns
  - research target for Parkinson's
  - caffeine binds to this receptor

GPCR Simulation methodology

- Molecular dynamics simulations using Gromacs (www.gromacs.org)
- 100,000+ concurrent simulations with no communication (114Kcore seconds/second = 1 Billion core-hours/year)
- Use random number seed and different starting structures to explore phase space
- Cluster generated structures and build Markov State Models representing probability of transition between substates
- Sample the substates and re-seed the simulations to avoid getting stuck in local minima
- Can simultaneously run any number of different proteins and small molecules as distinct experiments

Preliminary Results

- Roughly 2 milliseconds of simulation time in six months
- 4th largest Folding@Home provider (trailing only aggregate groups which have existed for years)
- Simulations execute 378TFlop/sec (similar to 17th fastest supercomputer in the world) (embarrassingly parallel code)
- over 260Tbytes of raw data generated- we are starting large-scale analysis by applying FlumeJava[1], Dremel[2] and Pregel[3] to existing algorithms including MSMBuilder[4]

1. http://research.google.com/pubs/DistributedSystemsandParallelComputing.html
2. http://www.google.com/research/pubs/pub36632.html
3. http://googleresearch.blogspot.com/2009/06/large-scale-graph-computing-at-google.html
4. https://simtk.org/home/msmbuilder