

On the Migration of the Scientific Code Dyana from SMPs to Clusters of PCs and on to the “Grid”

Michela Taufer *Thomas Stricker*
G rard Roos *Peter G ntert*

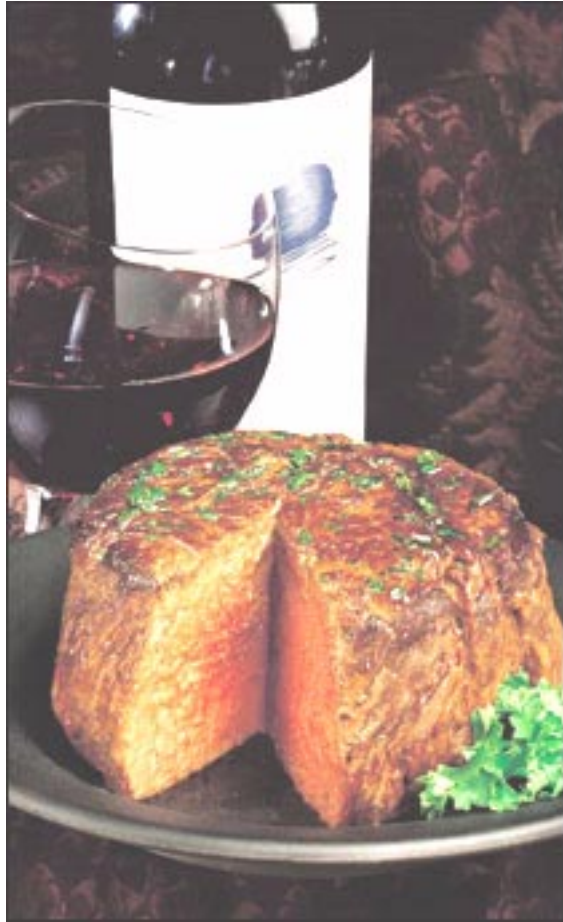
Institute for **Computer Systems** Institute for **Molecular Biology**
ETH Zentrum ETH H nggerberg
Swiss Institute of Technology
CH-8092 Z rich, Switzerland

Slides of this talk: www.cs.inf.ethz.ch/CoPs/talks/ccgrid02.pdf

 *Eidgen ssische
Technische Hochschule
Z rich*

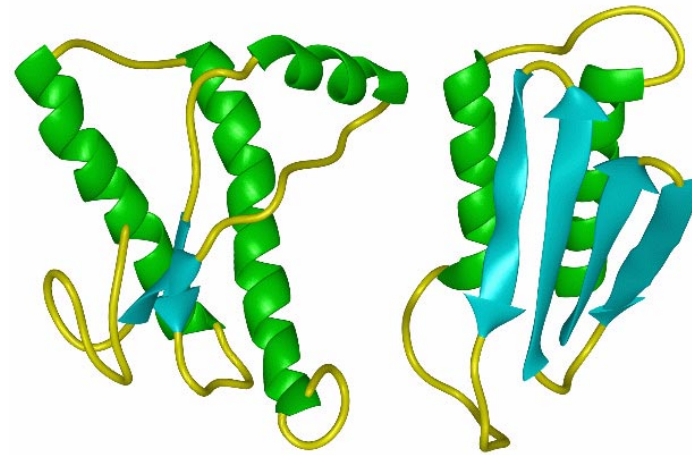
*CCGrid 2002
May 22, 2002*

Molecular Biology Problem



Is it still safe to eat such a steak?

How about infectious proteins like prions?

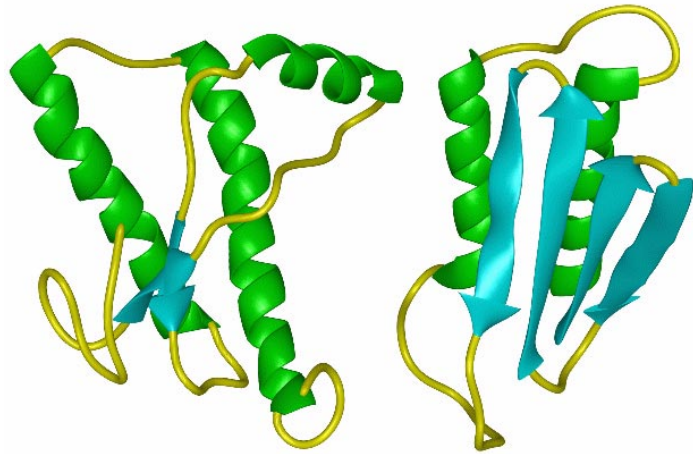


Two different foldings of the same prion...

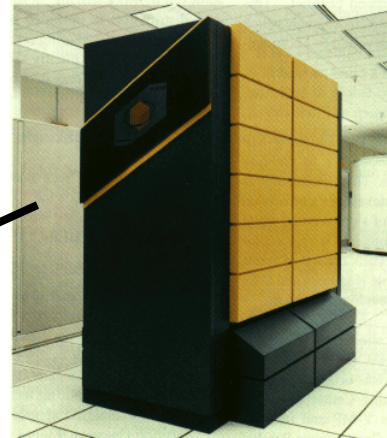
- Which one is infectious? Why?
- What is the precise structure of the infectious ones?

Computer Science Problem

Molecular Dynamics Code



Different Platforms



**Which ones
will work?**

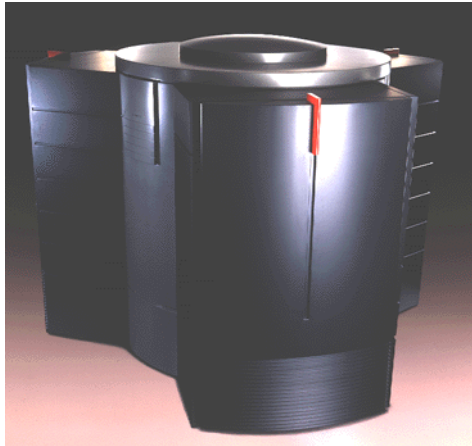
**Which ones
will be most
cost effective?**



Overview

- Motivation
- Platforms for high performance computing
 - From supercomputers to clusters of PCs
 - From clusters of PCs to widely distributed platforms on the “Grid”
- Migration of codes - our example Dyana
 - Reengineering codes during a migration
 - Performance model for the code Dyana
 - Calibration of the performance model
- Performance analysis and prediction for Dyana
- Conclusions and outlook

Platforms for High Performance Computing



Cray C90

DEC8400

DAS Cluster



fully coherent shared memory

distributed memory

SMP w/o slowdown

SMP with slowdown

Scalable MMP

Vector Memory System

Workstation Memory System

PC Memory System

Giga-Byte/s / Nano-Seconds
between Vector CPUs

Mega-Byte/s / Micro-Seconds
between Microprocessors

Vectorizing Compilers

Workstation Tools

MPI

Symmetry / Homogeneity



Platforms for High Performance Computing



DAS Cluster



Beowulfs



distributed.net



Scalable MMP

distributed memory

Mega-Byte/s / Micro-Seconds
between Microprocessors

MPI

PC Memory System

Mega-Byte/s / Milli-Seconds
between Microprocessors

Corba/RPC/RMI

Asymmetry / Heterogeneity

→ **Dyana** ?

Kind of Parallelism in Dyana

- Mainly **task parallelism**
- **Embarrassingly parallel** - but how much is really embarrassing?
- We can not **quantify amount of parallelism** based on the Dyana code alone. Need to look at the entire application.

Task parallelism specified in a small macro language, the Dyana Molecular Dynamic Simulation Language INCLAN:

```
1  do j 1 n parallel
2    cmd_1
3    cmd_2
4    ...
5    cmd_k
6  end do
7  cmd_seq
```

Properties of our Application of Dyana

Classical case of folding calculations is based on...

- simulated angle dynamics (angles of chemical bonds).
- energy minimizations of the bonds and far atom interactions.
- simulated annealing in search of state with low energy.

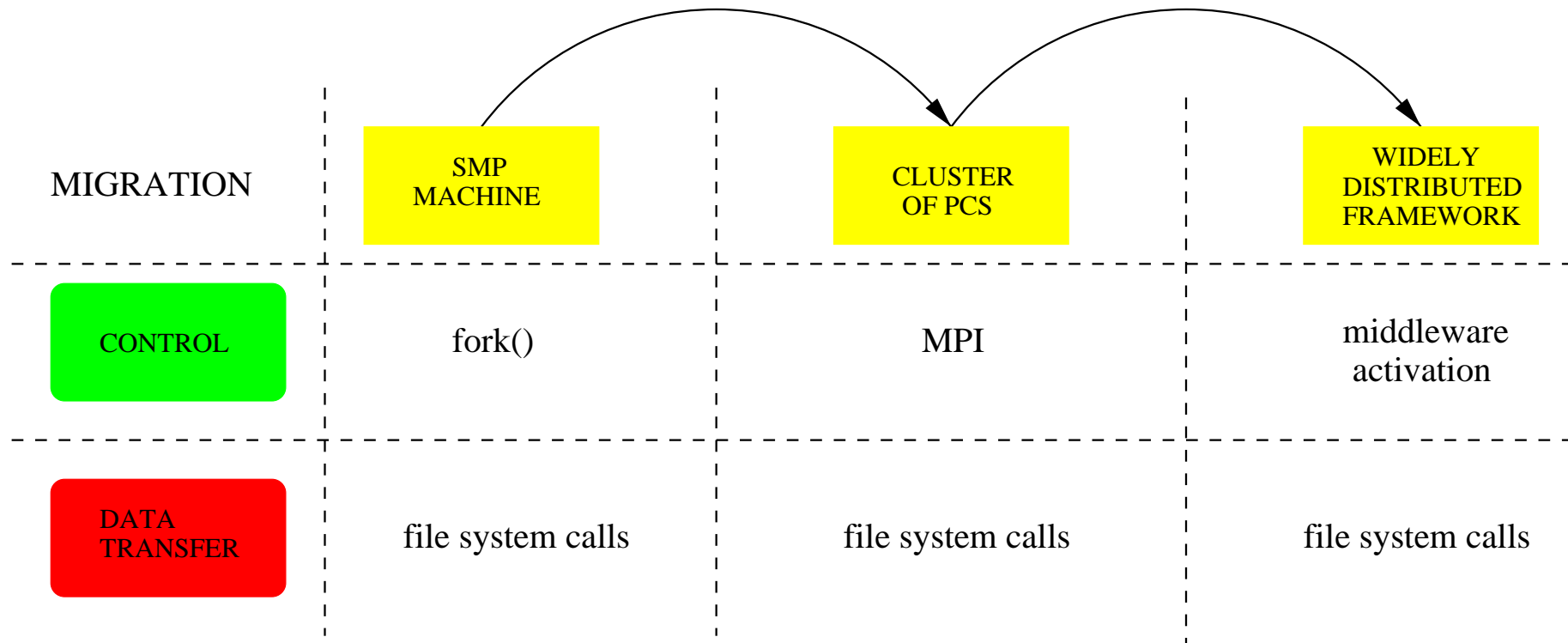
Quality of the simulation result is measured by ...

- matching the results to NMR spectroscopy data.
- this method of reconstruction is indirect and hard to compute.
- it is a methodology of a genius biologist - but hard to automate!

Candidates of conformers generated by random variations:

- no need for dealing with faults.

Reengineering the Code for Different Communication Mechanisms



Proposed Methodology for Migration

Create accurate performance model based on use of machine-resources:

- **Parallelism** - the number of CPUs involved.
- **CPU speed** - the number of floating point instr. per second.
- **Memory system usage** - access time for different access types.
- **Communication amount** - the number of messages and bytes communicated per task lifecycle.

Learn performance characteristics from existing migrations:

- **Establish** and **calibrate** performance model with existing data.

Predict a future migration by accurate performance estimates:

- **Use** performance model **to check viability** of a migration to a future computational platform (*distributed.net and grids*).

A Performance Model for Dyana

Total Execution Time for a Dyana Run:

$$t_{dyana} = t_{init} + t_{siml} + t_{comm}$$

$$t_{dyana} = \left\lceil \frac{n}{p} \right\rceil \frac{FLOP_{conf}}{CPU_clock\ rate} (a\alpha + b\beta + c\gamma) + \max \left[\frac{size_{data}}{bandwidth_{slave}} ; \frac{p\ size_{data}}{bandwidth_{master}} \right]$$

n : Number of Conformers

p : Number of CPUs

Modelling Local Execution Time

Based on # of FLOp per conformer...

| Protein | Architecture | Compiler | FLOp _{conf} |
|-------------|--------------|--------------|----------------------|
| hPrP | Pentium | PGI Compiler | 6.5 GFLOp |
| hPrP | Alpha | DEC Compiler | 5.3 GFLOp |
| ER2 | Pentium | PGI Compiler | 1.6 GFLOp |
| ER2 | Alpha | DEC Compiler | 1.4 GFLOp |

Murphy's law: all constants are variable.

Impact of Memory System on Performance

Novel model highly simplifies characterization of memory system in different computing platforms:

- **Parameters**

- **Access Pattern, Stride** (spatial locality)
- **Working Set** (temporal locality)

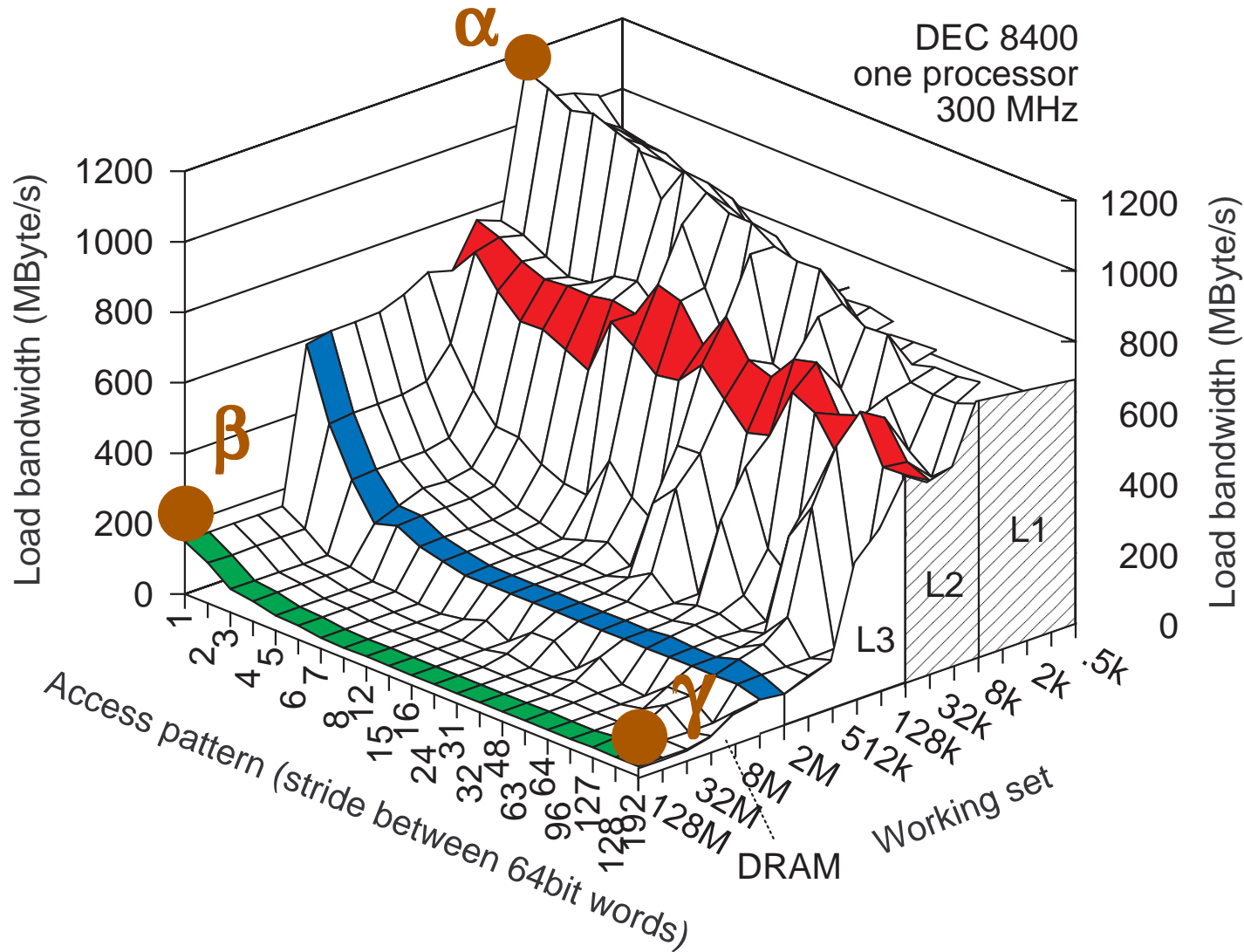
- **Values**

- **Transfer-Bandwidth** (for a large amount of data)

$$\text{Average-Access-Time} = 1 / (\text{Transfer-Bandwidth})$$

-> **Extended Copy Transfer Memory System Characterization**

Characterization of Local Memory



Modelling Local Execution Time (con't)

and FLOp per seconds adjusted with memory accesses:

| CPU Type | α | β | | γ | |
|---------------------|----------|---------|----|----------|----|
| | | CPUs | | CPUs | |
| | | 1 | 2 | 1 | 2 |
| Pentium III 933 MHz | 1 | 13 | 18 | 33 | 75 |
| Pentium III 800 MHz | 1 | 17 | 21 | 26 | 50 |
| Pentium II 400 MHz | 1 | 11 | 14 | 29 | 40 |
| DEC Alpha 667 MHz | 1 | 7 | 7 | 34 | 34 |

$$\text{clocks_per_FLOp} = a\alpha + b\beta + c\gamma$$

Coefficients a, b, c are determined with least square fit!

Modelling Communication Time

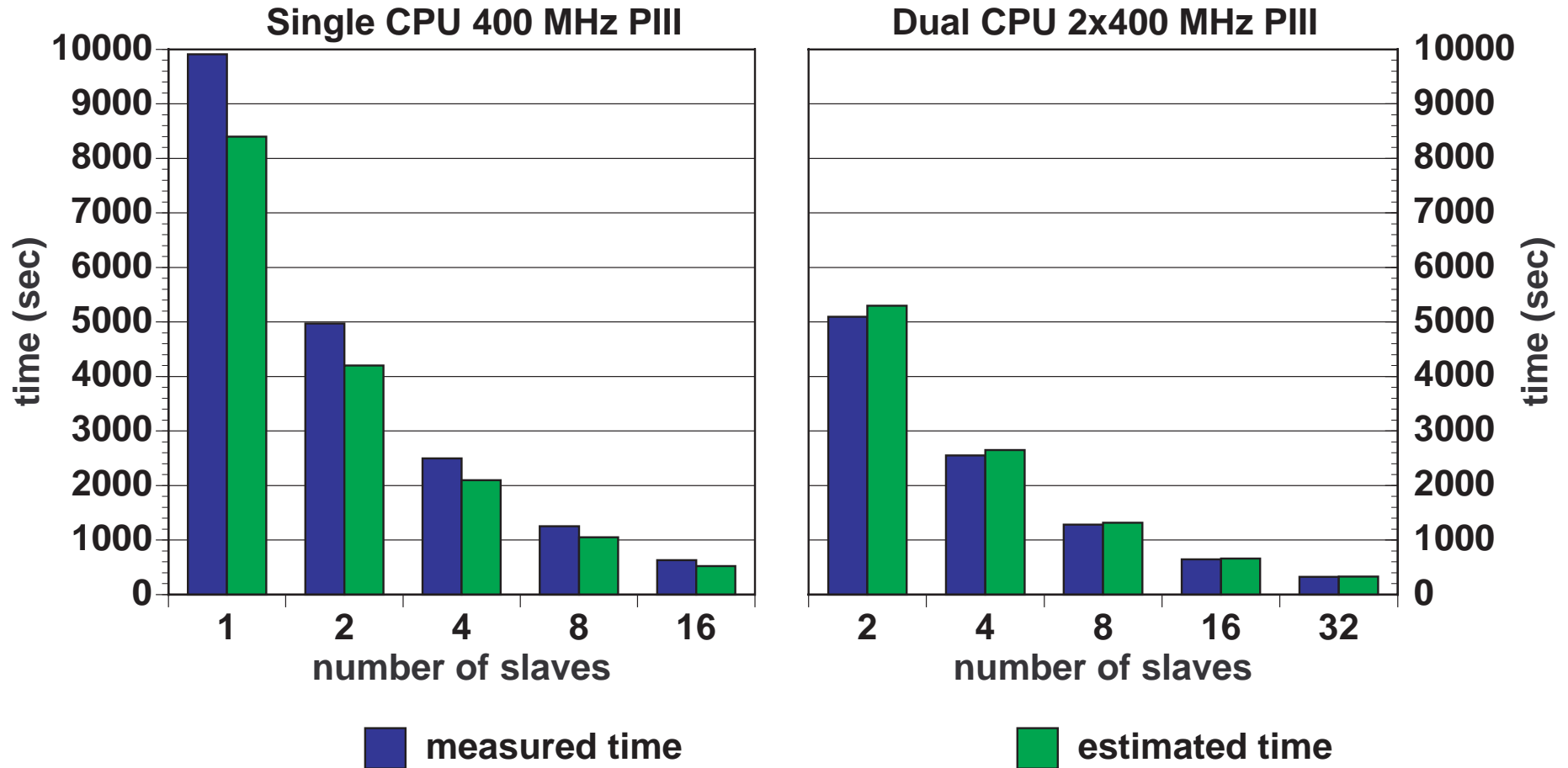
$$t_{comm} = \left[\frac{n}{p} \right] \max \left[\frac{size_{data}}{bandwidth_{slave}} ; \frac{p \cdot size_{data}}{bandwidth_{master}} \right]$$

No big magic!

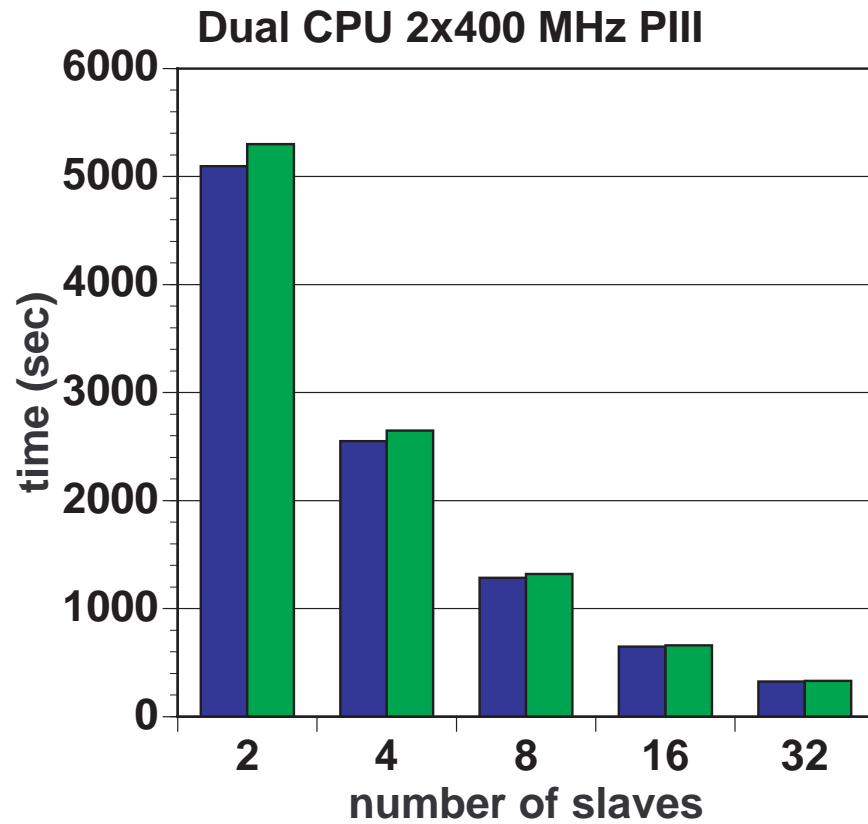
Messages are large

Either master or slave is the bottleneck

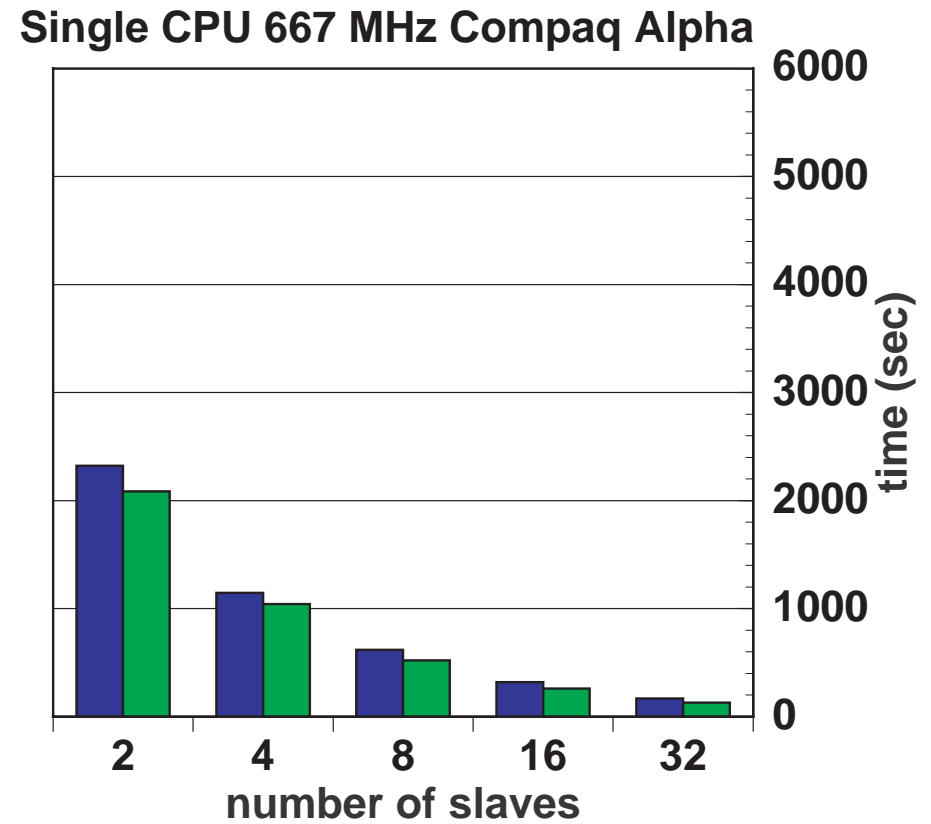
Quality of our Model



Quality of our Model

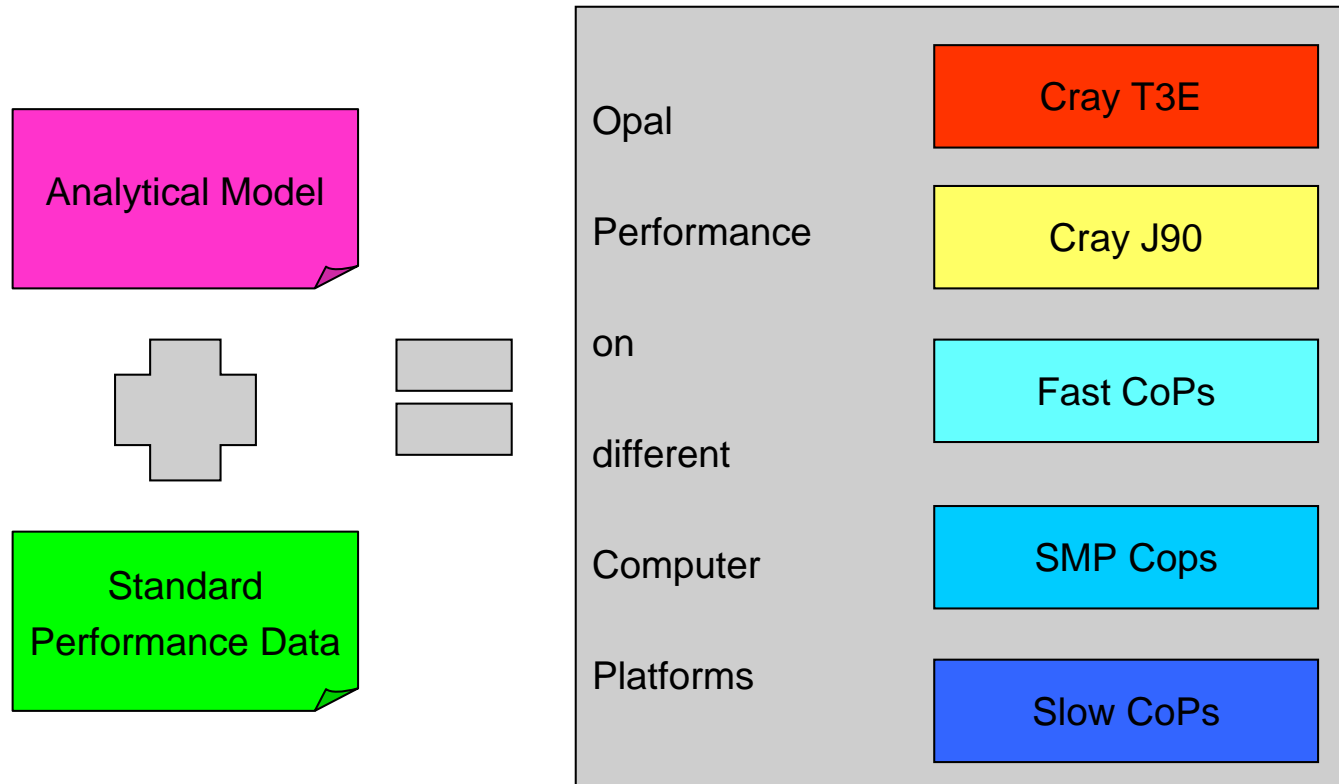


measured time



estimated time

Performance Prediction for Novel Platforms



- $t_{exec} = a_0 t_{flop} + (a_1 m_1 + \dots + a_k m_k) t_{clock}$

Amount of Parallelism usable for Grids

Modelling of sustainable parallelism for “distributed.net”

$$t_{comm} = \left\lceil \frac{n}{p} \right\rceil \max \left[\frac{size_{data}}{bandwidth_{slave}} ; \frac{p \cdot size_{data}}{bandwidth_{master}} \right]$$

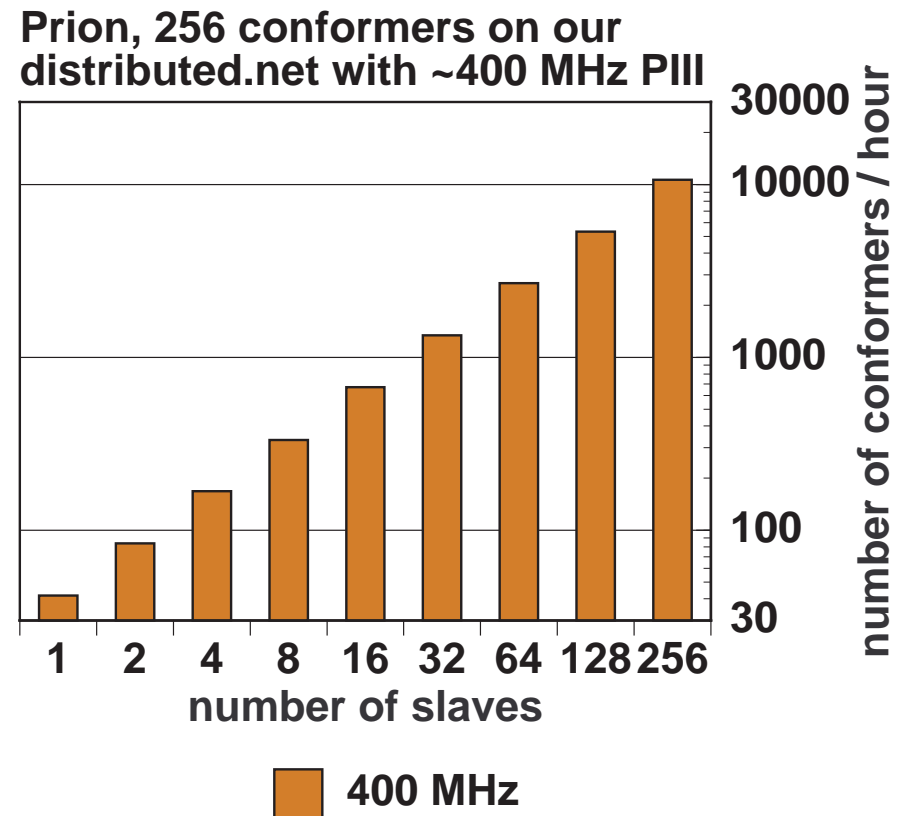
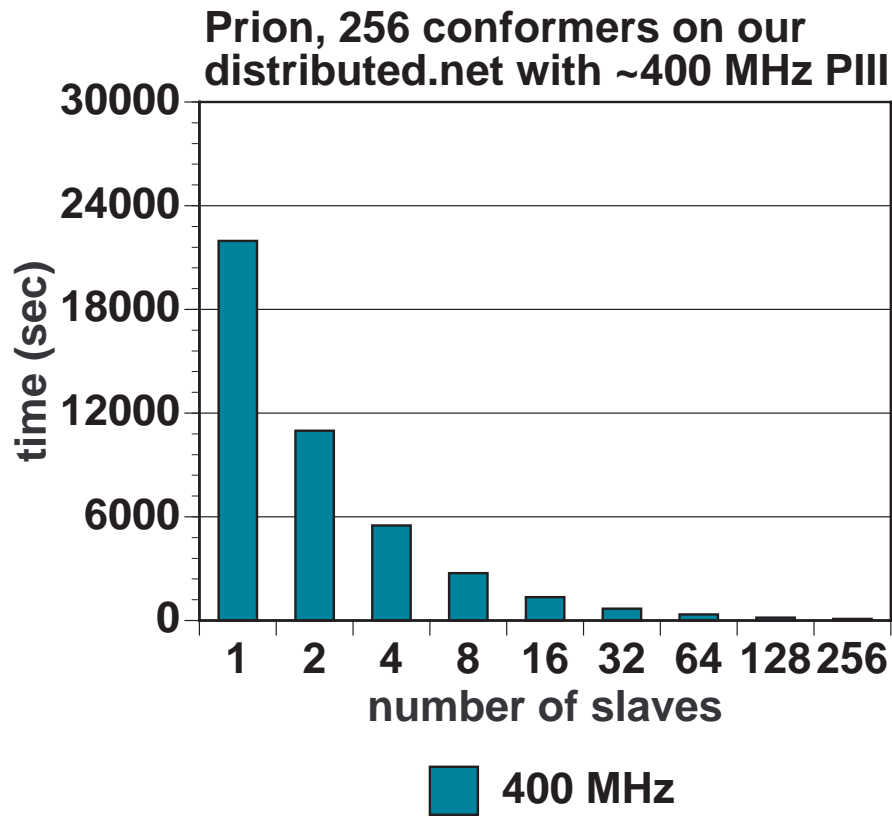
$$p > p_{max} = \frac{bandwidth_{master}}{bandwidth_{slave}} \approx 1500$$

$$t_{comm_conf} = \frac{size_{data}}{bandwidth_{slave}} \approx 3 \text{ sec}$$

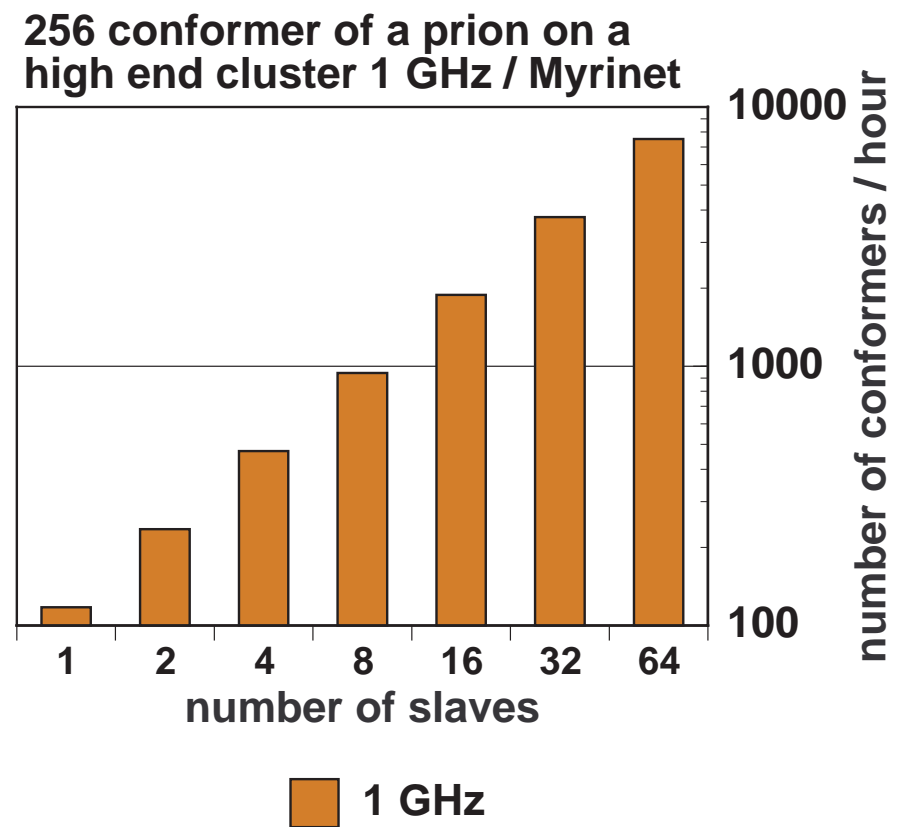
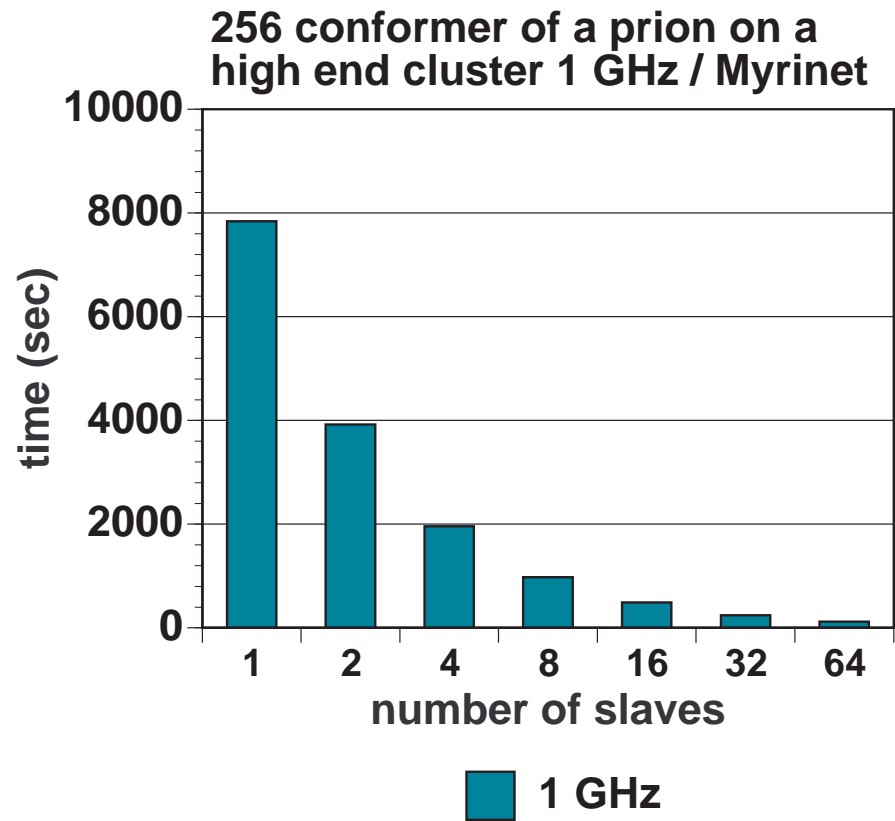
$$duty_cycle = \frac{t_{conf_siml}}{t_{comm_conf}} \approx \frac{85}{3} \approx 28$$

$$p_{max} \cdot duty_cycle = 1500 \cdot 28 \approx 42000$$

Scalability of Dyana for “distributed.net”



Scalability of Dyana for High End Cluster



Conclusion and Outlook

- We **successfully migrated** Dyana from **SMP-workstations** to **low-cost clusters** (Beowulf). Some reengineering required!
- Parallelism is specified **in the INCLAN** code and **not in Dyana**. Semi-automated research methodology of the biologist that calls for **short turn-arounds**.
- Based on performance analysis and modelling the prion folding app. with **Dyana scales** to over **42000 nodes** with one server. (**public interest** in research makes this ideal for **mad-cow-at-home**)
- Performance-**analysis, modelling&engineering** highly **successful**.
- Biologist lost interest prematurely - they asked funding agency for a cluster and got it! Proves **difficulty of interdisciplinary** work.
- A new much more promising project follows with **folding based on CHARMM** (task- and data-parallelism) - e.g. IPDPS 2002.