Case study C: Parallel MD on linux clusters using DIprotein program

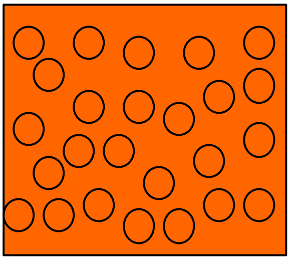
Stefano Cozzini (INFM udr Sissa)
(cozzini@sissa.it)

This case study

- ❑ First Part: MD
  - ❑ MD main concepts
  - ❑ MD for biological systems : proteins
  - ❑ Parallel MD: short review
- ❑ Second Part:
  - ❑ What is DIprotein
  - ❑ DIprotein parallel implementation
  - ❑ DIprotein parallel/serial performances
- ❑ Hands on session:
  - ❑ Install DIprotein
  - ❑ Measure Parallel performances

Molecular Dynamic technique

- ❑ A method to compute equilibrium and transport properties of classical many body systems using newton laws.



More details…

- ❑ Pick particles, masses and forces (or potential).
- ❑ Initialize positions and momentum (boundary conditions in space and time).
- ❑ Solve **F** = m **a** to determine **r**(t), **v**(t). (the integrator)
  - ❑ We need to make time discrete: $t_{k,}$ not continuous
- ❑ Compute properties along the trajectory
- ❑ Estimate errors.
- ❑ Try to use the simulation to answer physical questions.

Md algorithm

```
Program MD
call init
t=0
do while ( t<tmax)
   CALL FORCE(F,EN)
   CALL INTEGRATE
   t=t+dt
   CALL SAMPLE
end do
end
```

Formal aspects

- Equations of motion:
  - 2N differential equations:

$$m_i \frac{d\vec{v}_i}{dt} = \sum_j F_2(\vec{r}_i, \vec{r}_j) + \sum_j \sum_k F_3(\vec{r}_i, \vec{r}_j, \vec{r}_k) + \dots$$

$$\frac{d\vec{r}_i}{dt} = \vec{v}_i$$

- Energy conservation: $H = E_{cin} + V(r_1 \dots . r_n)$
- Integration algorithms: a lot !!
  - time-reversibile
- Many different thermo dynamical “ensemble”: NVT/NPT...

Characteristics of simulations

- ❑ Physical ones:
  - ❑ Potentials define the physics of the simulated systems
  - ❑ We need low accuracy because the potentials are not very realistic.
  - ❑ Small changes in accuracy lead to totally different trajectories. (the mixing or ergodic property).
  - ❑ Energy conservation is important; roughly equivalent to time-reversal invariance: allow 0.01kT fluctuation in the total energy.
- ❑ Computational ones
  - ❑ Time scales of simulations: 10^{-12} to several 10^{-9} second (nanoseconds)
  - ❑ CPU time is totally dominated by the calculation of forces.
  - ❑ Memory limits are not too important. [o(Natoms)]

Criteria for an Integrator

- ❑ simplicity (How long does it take to write and debug?)
- ❑ efficiency (How fast to advance a given system?)
- ❑ stability (what is the long-term energy conservation?)
- ❑ reliability (Can it handle a variety of temperatures, densities, potentials?)

The Verlet Algorithm

The nearly universal choice for an integrator is the Verlet (leapfrog) algorithm

$r(t+h) = r(t) + v(t)\,h + 1/2\,a(t)\,h^2 + b(t)\,h^3 + O(h^4)$ Taylor expand

$r(t-h) = r(t) - v(t)\,h + 1/2\,a(t)\,h^2 - b(t)\,h^3 + O(h^4)$ Reverse time

$r(t+h) = 2\,r(t) - r(t-h) + a(t)\,h^2 + O(h^4)$ Add

$v(t) = (r(t+h) - r(t-h))/(2h) + O(h^2)$ *estimate* velocities

Time reversal invariance is built in the energy does not drift either up or down.

How to set the time step

- ❑ Adjust to get energy conservation to 1% of kinetic energy.
- ❑ Leapfrog algorithm has a problem with round-off error.
- ❑ Use the equivalent velocity Verlet instead:

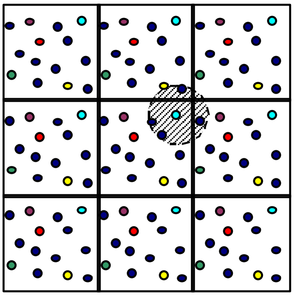
$$\mathbf{r}(t+h) = \mathbf{r}(t) + h\,[\,\mathbf{v}(t) + (h/2)\,\mathbf{a}(t)]$$
$$\mathbf{v}(t+h/2) = \mathbf{v}(t) + (h/2)\,\mathbf{a}(t)$$
$$\mathbf{v}(t+h) = \mathbf{v}(t+h/2) + (h/2)\,\mathbf{a}(t+h)$$

Spatial Boundary Conditions

Important because spatial scales are limited. What can we choose?

- No boundaries; e.g. droplet, protein in vacuum. If droplet has 1 million atoms and surface layer is 5 atoms thick⇒ 25% of atoms are on the surface.
- Periodic Boundaries: most popular choice because there are no surfaces (see next slide) but there can still be problems.
- Simulations on a sphere
- External potentials
- Mixed boundaries (e.g. infinite in z, periodic in x and y)

Periodic Boundary Condition:



In the figure, the central square is the real system and the surrounding squares are the replicated periodic images of the system.

Which potentials ?

- ❑ Atomic systems:
  - ❑ Simple Liquid/solids: short range
    - ❑ 2 body potentials: L.J. /Morse etc.
    - ❑ 3 body potentials : Tersoff etc..
    - ❑ Embedded atom model/ glue potential for metals
  - ❑ Charged systems: long range
    - ❑ Electrostatic potential : 1/r
- ❑ Molecular systems:
  - ❑ Intermolecular interactions (see above)
  - ❑ Intra-molecular interactions: see later

Short range: Lennard-Jones potential

$V(R) = \Sigma_{i<j} v(r_i - r_j)$ $v(r) = 4\varepsilon[(\sigma/r)^{12} - (\sigma/r)^6]$

ε= well depth

σ= wall of potential

Reduced units:

- ❑ Energy in ε
- ❑ Lengths in σ

Phase diagram is universal!

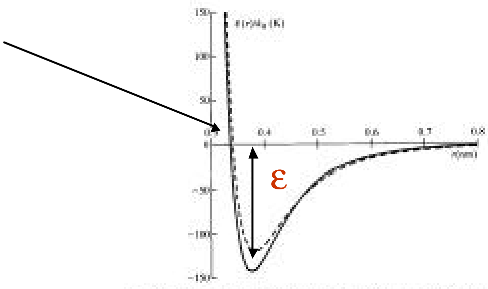


Fig. 1.3 Argon pair potentials. We illustrate the BBMS pair potential for argon (solid line) [Maitland and Smith 1971]. The BFW potential [Barker et al. 1971] is numerically very similar. Also shown is the Lennard-Jones 12-6 effective pair potential (dashed line) used in computer simulations of liquid argon.

Force computation (simple system)

- ❑ N body problem !
  - ❑ Each particle interact with the otherN-1

$\sim O(N^2)$

- ❑ 2 Kinds of interactions
  - ❑ Short range
  - ❑ Long range (i.e. electrostatic)

Tricks help to perform better

short range forces (1)

- ❑ Cutoff:

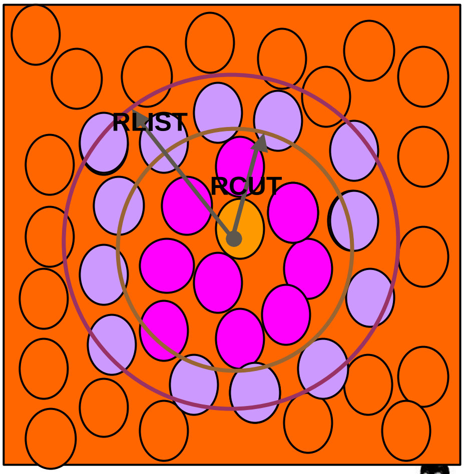
Do not count atoms beyond a certain cutoff radius

$T\sim cn_{rcu}N+ c_sN^2$

- ❑ Cutoff +Neighbor list

Build a list of neighbor atoms (larger than r_{cut}) and use it. Update the list every n_u timesteps.

$T\sim cn_vN + c_vN^2 /n_u$



Short range forces (2)

- ❑ Linked cell

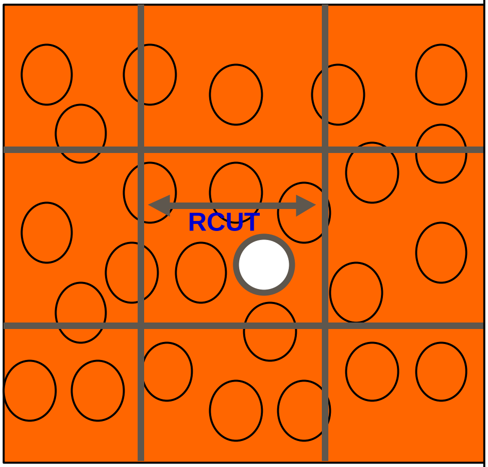
**Divide the system in cells.
Compute forces among atoms in the cell and in cell around**

$T=cn_lN+c_lN$

- ❑ Linked cell +verlet list

**Divide the system in cells.
Build a list form cells:
compute forces using the list.**

$T=cn_vN+c_lN/ n_u$



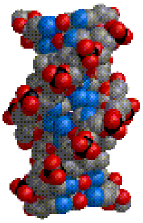
Long Range forces:

- ❑ Very time consuming task !
- ❑ Fundamental and not negligible in some system (I.e. biological systems)
- ❑ Some methods:
  - ❑ STANDARD
    - ❑ Reaction Field ~ N^2
    - ❑ EWALD ~ $N^{3/2}$
  - ❑ ADVANCED:
    - ❑ SPME ~ NlogN
    - ❑ FMA ~N (if N is enough large)

Md for proteins

- ❑ Proteins are complex object with a complex topology:
  - ❑ Many different types of atoms !
  - ❑ These atoms forms many different kind of structures..

COMPLEX Data Structures



- ❑ Potentials:
  - ❑ Inter-molecules (non-bonded) interactions
  - ❑ IntraMolecular (bonded) interactions

- ❑ Constraints..

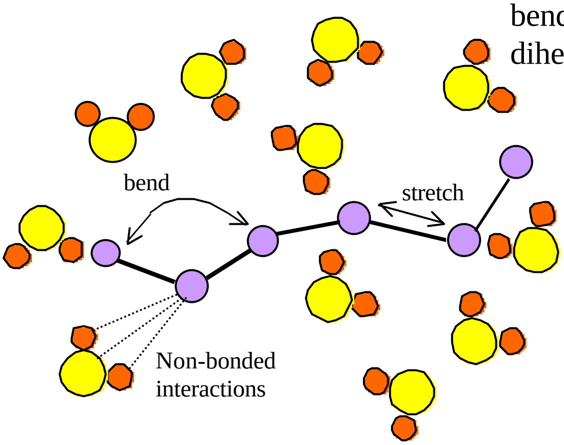
Protein potentials:

Bonded interactions

stretching: $V_{st} = \Sigma V(r_i - r_j)$

bending: $V_{bend} = \Sigma V(r_i, r_j, r_k)$

dihedral: $V_{dihed} = \Sigma V(r_i, r_j, r_k, r_l)$



Protein potentials

- Empirical potentials to describe interactions between molecules
- Many different “force fields” available.
- Typical potential is:
  - Two-body Lennard-Jones+ charge interaction
  - Bonding potential: $k_r(r_i-r_j)^2$
  - Bond angle potential $k_a(\theta- \theta_0)^2$
  - Dihedral angle: $v_n[1 - \cos(n\varphi)]$
  - All parameters taken from experiment.
  - Rules to decide when to use which parameter.

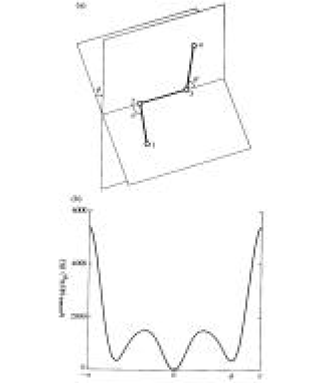


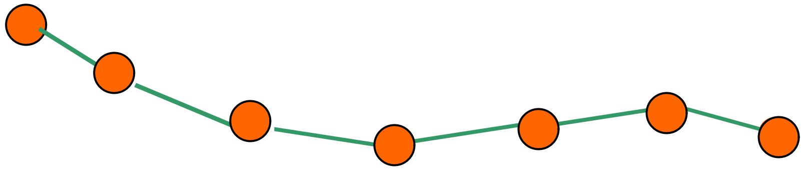
Fig. 1.8 (a) A model of butane [Ryckaert and Bellemans 1975]. (b) The torsional potential [Marechal and Ryckaert 1983].

Constraints

- In simulations of molecular systems there are various time scales:
  - Vibrational frequencies (very fast)
  - Rotations (fast)
  - Internal molecular reorientation (not so fast)
  - Diffusion (slow)
  - Melting (slow)
- The vibrational dynamics will set the time step but are decoupled and not interesting for the longer scales.
- Constraints simplify construction of potential.

Shake Method for constraints

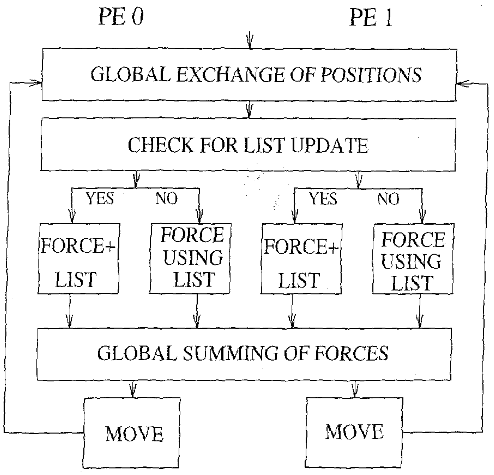
- Work directly with Cartesian atomic coordinates.
- Dynamics move forward without constraint
- Forces it back to satisfy the constraint.
  - Apply an iterative procedure (shake) to all the constraint atoms
  - In Proteins there are large chains of atoms to be constrained and this is a problem for parallel algorithms



Parallel MD: replicated data

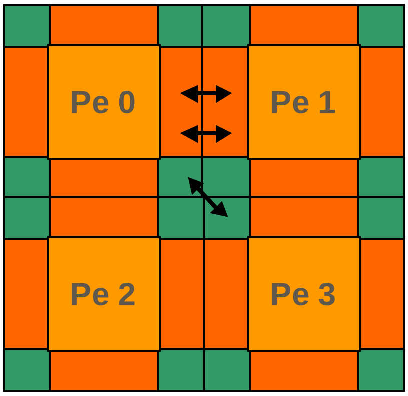
- The configuration data are replicated on each processor.
- Each processor computes on some part of this data.
- Communications:
  - Global communication: all processors involved
  - Easy to implement but heavy to compute
- RD is not scalable !!
  - Communication ~ o(Na)
  - Computation ~ o(Na/PEs)

2 Processor algorithm:



Parallel MD: domain decomposition

- Box divided among processors:
  - Each processors act on a subset
- Communications:
  - Between nearest neighbors
  - Complex patterns
- Scalability: OK !

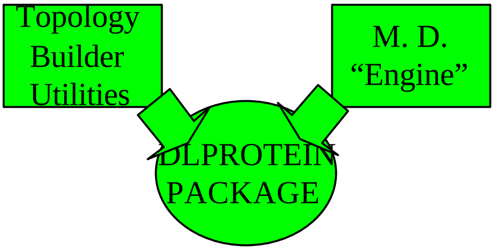


R. data vs D. Decomposition

	Replicated Data	Domain Decomposition
Implementation	Easy	Complex
Load balance	Easy	Could be difficult
Memory requirements	High	Low
Scalable	No	Yes
Treatment of Complex topology	Easy	Complex

Classical MD: DLPROTEIN-2:

- Package to create and simulate (MD) biomolecules: proteins
- starting point: DL_POLY from Daresbury Labs
- Developed by S. Melchionna + S.C.
- Replicated Data parallelism
- Object based approach using F90/95



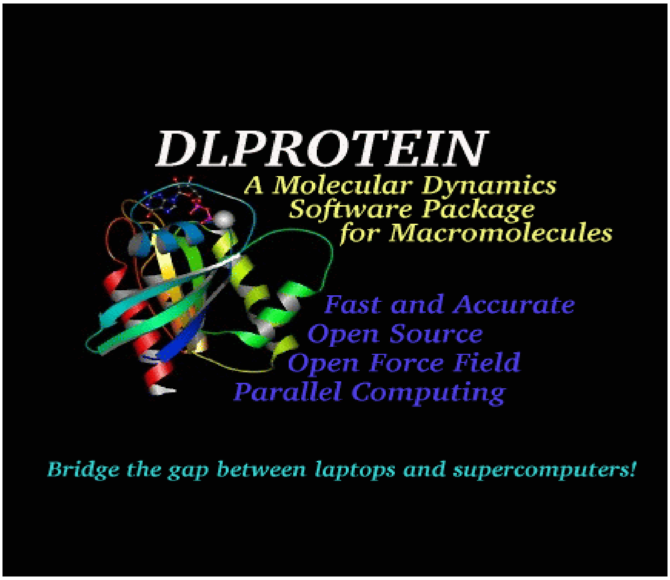
Topology Builder Utilities

M. D. “Engine”

DLPROTEIN PACKAGE

http://www.sissa.it/cm/DLPROTEIN

Ads.



DLPROTEIN

A Molecular Dynamics Software Package for Macromolecules

Fast and Accurate
Open Source
Open Force Field
Parallel Computing

Bridge the gap between laptops and supercomputers!

Dlprotein:

- ❑ Implements Replicated Data algorithm
- ❑ Parallel/Serial versions not too different
  - ❑ High level procedures are almost the same
  - ❑ A communication layer takes care to implement parallelism
- ❑ Parallel tasks are:
  - ❑ Forces Computations
    - ❑ Short range
    - ❑ Long range
  - ❑ Positions updates
  - ❑ Constraints Procedures

force computations in Dlprotein

- ❑ Short Range
- ❑ Neighbour list using all pairs scans if system small (case 2)
- ❑ Neighbour list using Linked Lists for large system (cas2 4)
- ❑ Long Range:
  - ❑ SPME: Smooth Particle Mesh Ewald
  - ❑ The only section of the code where to use external library (FFT)
- ❑ Periodic Boundary Conditions
  - ❑ Truncated octahedron
  - ❑ dodecahedron

Bonded interaction in Dlprotein

- ❑ DLPROTEIN uses “interaction tables”
  - ❑ Each bonded interaction is identified :
    - ❑ $U_{type}(i_{type})$ $i_{type}=1,N_{type}$
    - ❑ type=bond,angle,dihedral
  - ❑ Array of pointers :Key_{type} (D_{type},i_{type}) store indexes of atoms needed to define the interaction
    - ❑ Dtype=2 for bond 3 angles 4 dihedral
  - ❑ This array used to compute forces and energy

```
          DO NUM=1, NANGLES
C ATOM NUMBERS
          I=KEYANG(1,NUM)
          J=KEYANG(2,NUM)
          K=KEYANG(3,NUM)
C Compute Forces using atoms I,J,K
          END DO
```

Parallel short range computation:

```
C External loop:
     DO I=IDNODE+1, N-1,NODES

c Internal one
             DO k=1,n_of_neighbour(I)
             j=list(k,i)
Ccompute forces
c
             fi=fi+fij
             fj=fj-fij
             END DO
     ENDDO
```

EX:4 PE

Node 0: 1 5 9 13

Node 1: 2 4 10 14..

Node 2: 3,7,11,15..

Node 3: 4,8,12,16….

⌘ GLOBAL SUM:

(all to allcommunication:reduce)

Size of N of atoms

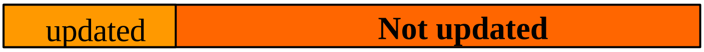
$$f_I = \sum_{k=0}^{k=nodes-1} f_{Ik}$$

Parallel long range computation:

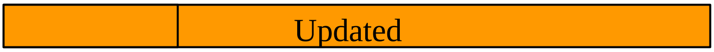
- ❑ Based on Parallel SPME
- ❑ SPME means GRID => Domain decomposition (great!) (scales well...)
- ❑ Drawback: in SPME we need FFT...
- ❑ Dlprotein uses the parallel 3D FFT algorithm available on the market:
  - ❑ Vendor Library (T3E /SP3)
  - ❑ Public domain: FFTW_MPI (linux cluster)
- ❑ Communications: within FFT (not coded by us !)

Parallel task for position update:

- Computation: Trivial : each PE updates N_a/N_{PE} positions



Communication: operation to exchange updated positions:



- All to all operation (Merge(mpi_all_gather)) size= NATOMS

Parallel bonded forces

- Each node requires (and owns) a copy of all the lists of bonded interactions
- Not a major computational task
- Parallelisation must be applied (or incorrect results)
- Parallelisation comes free (already doing a global sum for forces)

```
          DO NUM= IDNODE+1, NANGLES,NODES
C ATOM NUMBERS
          I=KEYANG(1,NUM)
          J=KEYANG(2,NUM)
          K=KEYANG(3,NUM)

C Compute forces using atoms I,JK
     ....
          END DO
```

Parallel task for constraints(shake):

- Computation: each PE shakes a Nmol/Npe if molecules can be distributed*

- Communication: operation to exchange updated positions:

  - 1.Map Molecules in atoms (indexing operations)
  - 2.All to all operation (Reduce) size= NATOMS

*(N of total constraints)/Npe <(N of constraints of the largest molecules)

Dlprotein: computational aspects

Tasks	Methods	Computational cost
Short Range Forces	Link-cell + neighbor lists	N_{neigh} x N_a
Long Range Forces	Smooth Particle-Mesh Ewald (SPME)	N_a x ln N_a
Bonded Forces comp.:	2-body,3-body,4-body potential	negligible
Updating Atoms	velocity verlet algorithms	negligible
Constraint procedures	Shake/Rattle	$N_{constraints}$

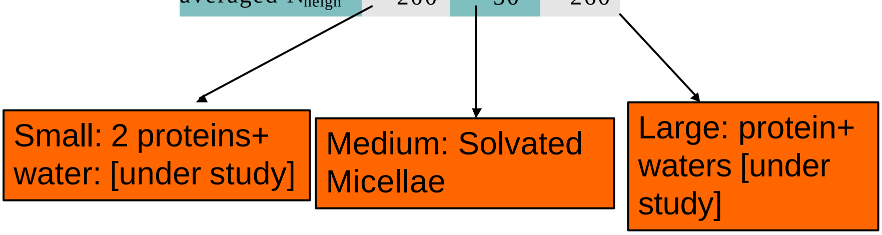
Scattered data access: not cache friendly !!

The Dlprotein bottleneck: communication

- For each step:
  - Reduce (all to all) to sum forces...
  - Merge (all to all) to update positions
  - Reduce(all to all) to update positions after shake procedure
  - Sum over processors some (all to all) dynamic quantities (energy.. Virials ...)
- To run efficiently:
  - Good bandwidth
  - Efficient all to all implementations
  - Latency is a problem but not too much

DLPROTEIN data sets:

Parameter	Test 1	Test 2	Test 3
N_{atoms}	19176	32829	58701
N of waters	5494	8513	17824
averaged N_{neigh}	≈ 200	≈ 50	≈ 260

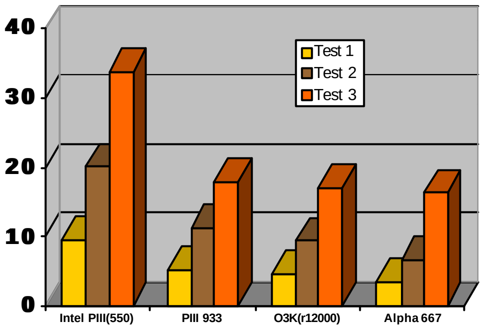


Small: 2 proteins+ water: [under study]

Medium: Solvated Micellae

Large: protein+ waters [under study]

DIprotein: serial performance (june 2001)

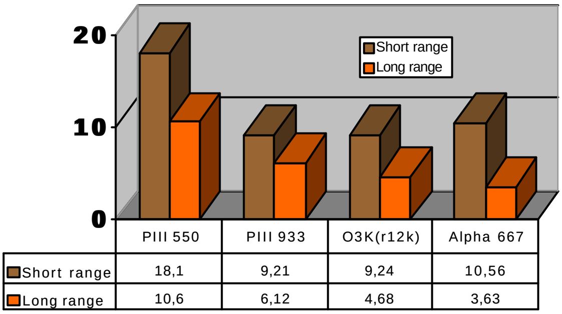


Note:
1. Increase from 550 to 933 is impressive..

2. INTEL 933 not far from RISC (especially if the system is large)

3. Alpha 667 is the best one...

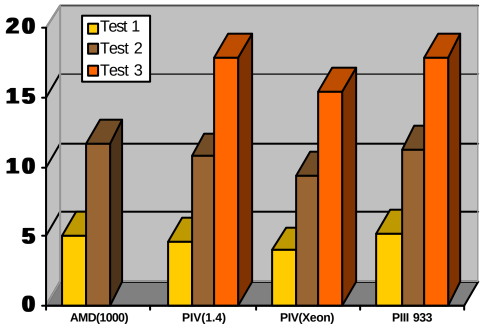
DLPROTEIN: Test 3 dissection:



	PIII 550	PIII 933	O3K(r12k)	Alpha 667
🟫 Short range	18,1	9,21	9,24	10,56
🟧 Long range	10,6	6,12	4,68	3,63

Observations:
1. INTEL 933 ~ RISC for short range (cache unfriendly)
(expensive RIMM efficient !!)
2. RISC much better for long range (cache friendly and FFT routine)

Dlprotein: serial performance october 2001

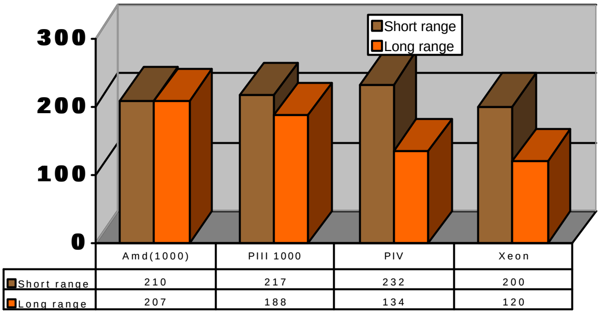


Note:
1. Increase from PIII to PIV is not significant…

2. PIV Xeon better

3.AMD same as PIII (PIV)

DLPROTEIN: Test 1 dissection:

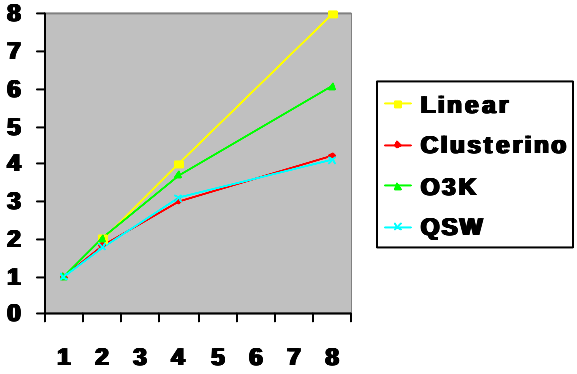


	Amd(1000)	PIII 1000	PIV	Xeon
🟫 Short range	210	217	232	200
🟧 Long range	207	188	134	120

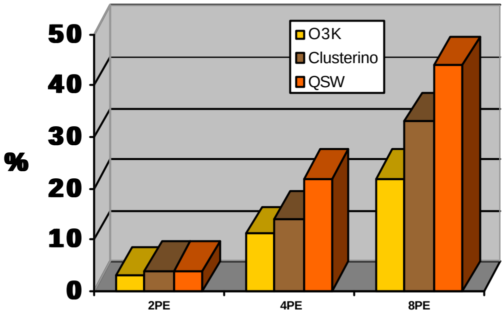
Observations:
1. PIII (and AMD) better in short range than PIV !!! (why ?)
1. PIV and Xeon much better for long range.. (expected…)

DIprotein 2.1 scalability (test 2)

Scalability is limited by RD parallel algorithm



Communication times vs global time for DIprotein (test 2) :



communication/computation ratio is getting too large for Linux clusters !!

DIprotein communications:

- Reduce operations:
  - collective operation
  - datasize ~ Na
- Merge operations:
  - collective operations
  - data size ~Na

	Reduce		
PE	O3K	QSW	550/(933)
2	4.4	8.6	12.8(10.2)
4	9.6	22.7	28.2(25.1)
8	11.9	22.5	44.7

	merge		
PE	O3K	QSW	550(933)
2	2.9	2 .0	6.9(4.7)
4	4.1	4.0	13.1(11.7)
8	5.7	4.2	27.0

How to improve performance ?

- Parallel:
  - shake procedure: use the same mapping system for molecules and atoms: save a communication operation
- Serial:
  - Fit data structures on the hardware machine:exploit cache meachanism

An example:

Better algorithm to deal with water molecules
(cache access improved): Result on water sample

	INTEL 933			ALPHA 667		
	V 2.0	V 2.1	Speed-up	V 2.0	V 2.1	Speed-up
Total	2.99	2.00	+49%	2.41	2.32	+4%
Short range	1.81	1.14	+59%	1.36	1.08	+25%
Constraints	0.54	0.24	+225%	0.41	0.60	-46%

Result on test 2:

Section	INTEL 933	ALPHA 667
Total	**+23%**	**+17%**
Short Range	**+27%**	**+29%**
Constraints	**+273%**	**+23%**

Summing up:

- ❑ MD codes on Linux Clusters:
  - ❑ Linux cluster is a very good resource for this kind of codes
  - ❑ Intrinsic limit of the code limit scalability (even on other parallel machines) but performance/price is unbeatable
  - ❑ Some tricks allow to do better (W.I.P.)
  - ❑ New (and cheap) processors should increase performances