The CPMD Code: from Clustered Regatta Frames to Blue Gene

A .Curioni
Computational Biochemistry and
Material Science Group
IBM Zurich Research Laboratory

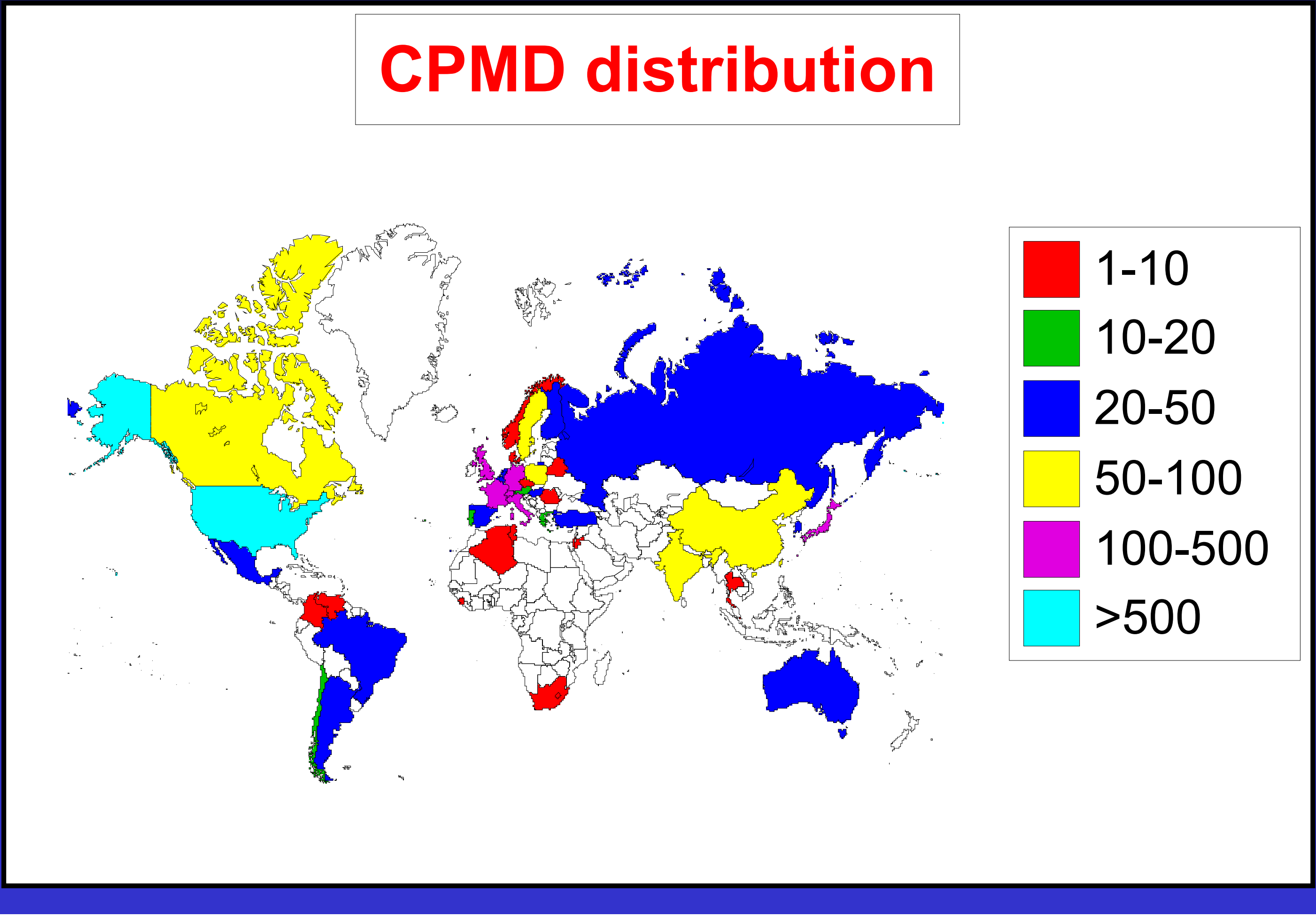
Outline of the talk

- Introduction
  - The CPMD Code: some history.
- Theory and Implementation
  - Strategy and Single Processor Optimization
  - Distributed Memory Parallelization (MPI)
  - Mixed Distributed/Shared Memory Parallelization (MPI/OpenMP)
  - Taskgrops parallelization
- Benchmark Results on p690, JS20 and BG/L
  - Results and Discussion
- Conclusion

CPMD history

- Born at IBM Zurich from the original Car-Parrinello Code in 1993;
- developed in many other sites during the years (more than 150,000 lines of code); it has many unique features, e.g. path-integral MD, QM/MM interfaces, TD-DFT and LR calculations;
- since 2001 distributed free for academic institutions (www.cpmd.org); more than 5000 licenses in more than 50 countries.

CPMD distribution:
An extended community.

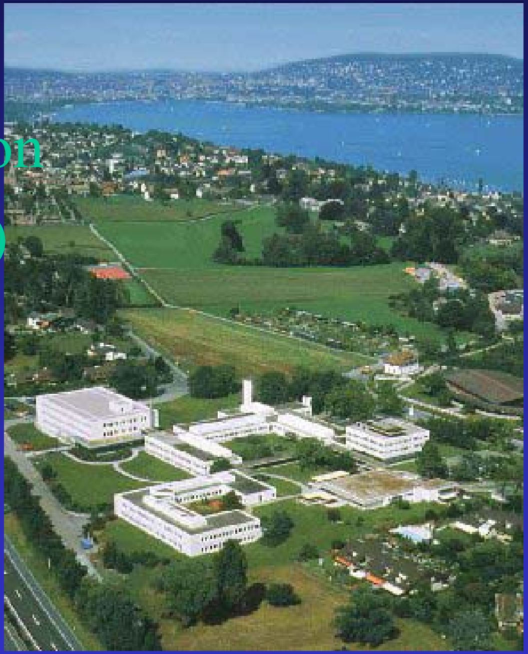


CPMD at IBM Zurich

Year	**System (limit)**	**Type of calculation**	**HW**	**Type of algorithm**
1992	one organic molecule of ~50 atoms	dynamics; electronic structure	RISC6000/580 **(125 MFlops)**	serial
1994	liquid 100 atoms. organics water	reaction dynamics - free energy	SP1-16 nodes **(2 GFlops)**	parallel/MPI
1996	biomolecules 200 atom models ***and in water***	reaction dynamics; electronic structure	SP2/66MHz 16 nodes **(4.2 GFlops)**	parallel/MPI
1998	complex interfaces 400 atoms. ***water/oxide*** ***organic/metal***	all of the above	SP2/166MHz 32 nodes **(20.5 GFlops)**	parallel/MPI
2000	supramolecular systems 1000 atoms. ***2D quantum dots arrays***	all of the above	SP3/200MHz 64/2 ways nodes **(102.4 GFlops)**	parallel/MPI+ OpenMP
2002	small proteins realistic interfaces 3000 atoms	all of the above	p690/1.3GHz 8/32 ways nodes **(1.3 TFlops)**	parallel/MPI+ OpenMP+ Globus

Outline of the talk

- Introduction
  - The CPMD Code: some history.
- Theory and Implementation
  - Strategy and Single Processor Optimization
  - Distributed Memory Parallelization (MPI)
  - Mixed Distributed/Shared Memory Parallelization (MPI/OpenMP)
  - Taskgrops parallelization
- Benchmark Results on p690, JS20 and BG/L
  - Results and Discussion
- Conclusion



Total Energy of a molecular system

(Khon-Sham formulation of DFT in the BO approximation)

$E_{tot}(R,r)=E_{el}(r;R)+E_{ion}(R)$

$E_{el}(r;R)=E_k+E_{ext}+E_h+E_{xc}$

$n_e(r)=\Sigma_\iota\ f_i\ |\Psi_\iota|^2$

$E_k=-1/2\ \Sigma_i <\Psi_\iota\ |\ \Delta\ |\Psi_\iota>$ (Kinetic Energy)

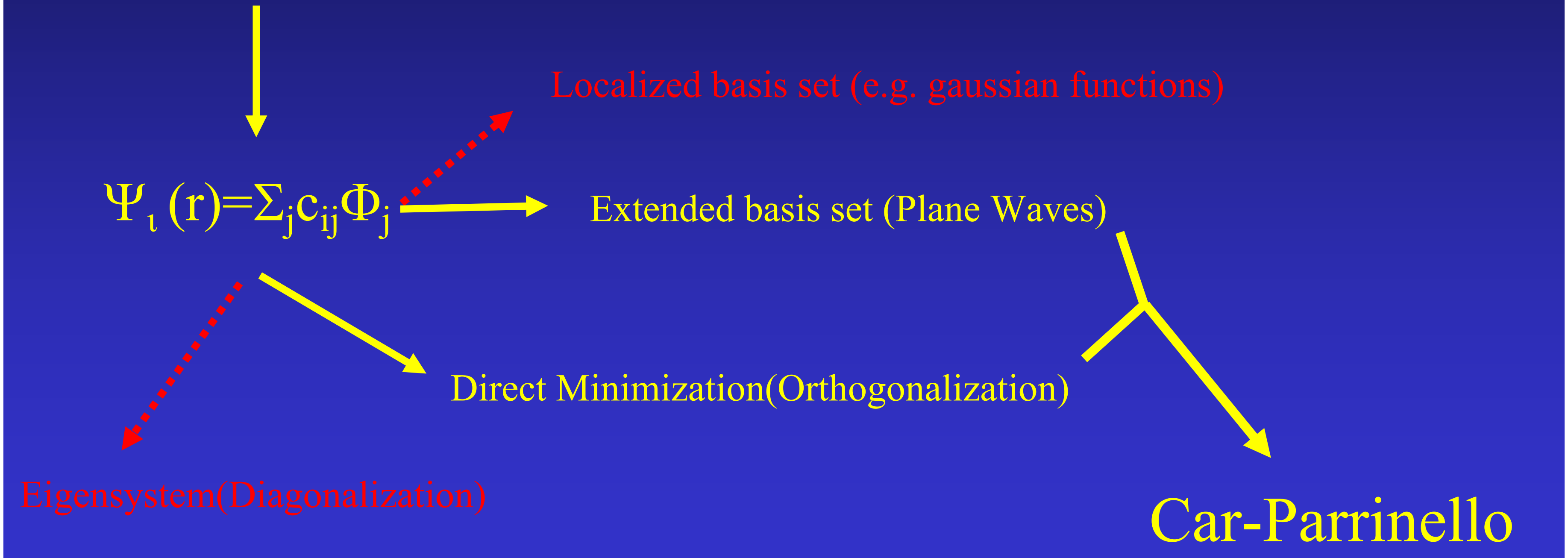
$E_{ext}=\int V_{ext}(r)n_e(r)dr$ (Nuclei/Electrons interaction Energy)

$E_h=1/2\int\int n_e(r_1)n_e(r_2)\ dr_1dr_2$ (Hartree Energy)

$E_{xc}=\int \varepsilon_{xc}(r)n_e(r)\ dr$ (Exchange-Correlation Energy (*ManyBody Term*))

Optimization of Molecular Structure

Optimization of E_{el} ------> Forces on Ions ------> Structure optimization or
Molecular Dynamics



$\Psi_{\iota}(r)=\Sigma_j c_{ij}\Phi_j$

Localized basis set (e.g. gaussian functions)

Extended basis set (Plane Waves)

Direct Minimization(Orthogonalization)

Eigensystem(Diagonalization)

Car-Parrinello

Total Energy of a molecular system with a plane wave basis set

$$\psi(\mathbf{r}) = \frac{1}{\sqrt{\Omega}} \sum_{\mathbf{G}} c_i(\mathbf{G}) e^{i\mathbf{G}\mathbf{r}}$$

$$E_{kin} = \frac{1}{2} \sum_i f_i \sum_{\mathbf{G}} \mathbf{G}^2 \left| c_i(\mathbf{G}) \right|^2$$

$$E_{loc} = \Omega \sum_{\mathbf{G}} n^*(\mathbf{G}) S(\mathbf{G}) V_{loc}(\mathbf{G})$$

$$E_H = 2\pi\Omega \sum_{\mathbf{G}} n^*{}_t(\mathbf{G}) \frac{1}{\mathbf{G}^2} n_t(\mathbf{G})$$

$$E_{XC} = \int d\mathbf{r} F_{XC}\left[n\right]$$

$$E_{nl} = \sum_I \sum_j f_j \omega \left| F_j^I \right|^2$$

Scaling I

The size of a system is determined by the number M of PWs needed for its accurate description, the number N of electrons, and the number I of ions.

Electronic minimization:

(CPU time)

-$NM\log M$ (e.g. calculation of the density, calculation of the forces)

-N^2M (e.g. orthogonalization)

(Memory)

-NM (electronic wavefunction in reciprocal space)

Scaling II

Structure minimization:

(CPU time)

-I^3 (NR)

(Memory)

-I^3 (Hessian)

For molecular systems up to 1000 atoms

(max size for a 100 GFlop computer):

M>>>N>I

Calculation dominated by 3D-FFT, memory by PWs.

Implementation Strategy

- Reduce the number of operations
  - use symmetry of the wavefunctions at Γ point: $c_i(\mathbf{G}) = c_i^*(-\mathbf{G})$
    - reduce the number of operation by a factor 2
    - two 3D-FFT can be made at the price of one
  - take advantages of the sparsity of our FFT (two diverse cutoffs)
    - reduce the number of operation by a factor 2 – std 3D-FFT routines not usable
- Use optimized BLAS routines (ESSL, ATLAS) whenever possible
  - Prefer routines where block algorithms are more efficient (e.g. DGEMM vs DGEMV)

Example of Single processor performance on Power4 (1.3GHz)

- System : 216 atom SiC supercell, 128x128x128 Mesh, Cutoff 60 Ry (477,534 plane waves)
  - Time per MD step = 313 s
  - Performance = 2 GFlops
  - Relative Perfomance= 38 %

- System : 512 atom SiC supercell, 168x168x168 Mesh, Cutoff 60 Ry (1,131,630 plane waves)
  - Time per MD step = 2761 s
  - Performance = 2.4 GFlops
  - Relative Perfomance= 46 %

DGEMM max Relative Performance = 66 %

Distributed Memory Parallelization

- Distribute PWs and real space mesh following these conditions:
  - **A processor hosts full planes of real-space grid points.**
  - Each processor has the same number of plane waves.
  - All plane waves with common y and z components are on the same processor.
  - The number of different (y, z) pairs of plane-wave components is the same on each processor.
  - The number of real-space planes is the same on each processor.

Distributed Memory Parallelization

Parallelize 3D-FFT : *(Mx,My,Mz) points , F(x,y,z)*
My,Mz 1D-FFT along x
data transposition F(x',y,z)
Mx,Mz 1D-FFT along y
data transposition F(x',y',z)
Mx,My 1D-FFT along z

Using the data distribution presented in the previous slide only a single ALL TO ALL communication is needed after the first 1D-FFT.

All the other data that have not direct dependence on plane waves indexes are replicated (e.g. overlap matrices)

Distributed Memory Parallelization Problems

- The number of grid points in x direction limits the maximum number of processors that can be used efficiently in the 3D FFT. (major cause of load unbalance).

- The the linear algebra involve in the calculation (e.g. orthogonalisation) is not parallel and limits the maximal speedup that can be achieved (and therefore the size of the system that can be calculated).

Distributed Memory Parallelization

Problems

- Replicated overlap matrices might become a memory bottleneck for large systems on many processors with small memory (e.g. Blue Gene/L)

- The time required for the all-to-all communications scale as Npe * latency, downgrading the performance in the case of communication adapters with relatively high latency.

(the latency it is always determined by the slowest link in clustered SMP servers)

Distributed Memory Parallelization

Solutions

- Mixed MPI/SMP parallelization
  - reduce the number of MPI tasks and therefore the impact of latency and load unbalance
  - parallelize the linear algebra involved in the orthogonalization
  - requires an SMP hardware

- Taskgroups parallelization
  - reduce drastically load unbalance and hide latency in the bandwidth
  - needs roughly double amount of communication

Mixed MPI/OpenMP parallelization

- OpenMP strategy:
  - Same data distribution
  - Add OpenMP directives on all large loops (e.g. NNR1, NGW)
  - Link multithreaded linear algebra libraries (e.g. esslsmp)
  - Add OpenMp directives to the zeroing routines and to the Gather and Scatter routines
  - Use 4-8 SMP task per MPI task (SMP parallelization within a MCM)

Taskgroups parallelization

- **Taskgroups strategy**:
  - Same data distribution
  - Given a number of taskgroups, arrange the processors in a 2 dimensional array; each processor is member of its column group and row group
  - initial data exchange in the column groups
  - each row group perform independently the FFTs
  - A final data exchange in the column groups restore the o original data distribution

- **If number of taskgroup = number of processors this approach correspond to a parallelization over the electronic states**

Distributed Memory Parallelization
Solutions

- Mixed MPI/SMP parallelization
  - reduce the number of MPI tasks and therefore the impact of latency and load unbalance
  - parallelize the linear algebra involved in the orthogonalization
  - requires SMP hardware

- Taskgroups parallelization
  - reduce drastically load unbalance and hide latency in the bandwidth
  - needs roughly double amount of communication

Outline of the talk

- Introduction
  - The CPMD Code: some history.
- Theory and Implementation
  - Strategy and Single Processor Optimization
  - Distributed Memory Parallelization (MPI)
  - Mixed Distributed/Shared Memory Parallelization (MPI/OpenMP)
  - Taskgrops parallelization
- Benchmark Results on p690, JS20 and BG/L
  - Results and Discussion
- Conclusion

Test Cases

- Test 1 : **216 atoms SiC supercell, 128x128x128 Mesh, Cutoff 60 Ry (477,534 plane waves)**

- Test 2 : 512 atoms SiC supercell, 168x168x168 Mesh, Cutoff 60 Ry (1,131,630 plane waves)

- **Test 3 : 1000 atoms SiC supercell, 256x256x256 Mesh, Cutoff 60 Ry (2,209,586 plane waves)**

Test 1: performance and Scaling on Single Regatta Frame (p690 1.3GHz)

- System : 216 atoms SiC supercell, 128x128x128 Mesh,
  Cutoff 60 Ry (477,534 plane waves)

N Proc	Time/Step (s)	Performance (GFlops)	Parallel Efficiency	Relative Performance
1	313	2.0	-	38%
2	161	3.9	98%	37%
4	83	7.6	95%	36%
8	48	13.1	82%	32%
16	28	23.1	72%	28%
32	18	35.9	57%	22%

Test 2: performance and Scaling on Single Regatta Frame (p690 1.3GHz)

- System : 512 atoms SiC supercell, 168x168x168 Mesh,
  Cutoff 60 Ry (1,131,630 plane waves)

N Proc	Time/Step (s)	Performance (GFlops)	Parallel Efficiency	Relative Performance
1	2761	2.4	-	46%
2	1422	4.6	97%	45%
4	728	8.9	95%	44%
8	395	16.5	88%	41%
16	212	30.8	82%	38%
32	126	51.6	72%	34%

Example: MPI vs MPI/SMP
Single Regatta Frame

System	MPI/SMP tasks	Performance (GFlops)
216 atoms	32/1	36
216 atoms	8/4	36
512 atoms	32/1	52
512 atoms	8/4	50

Test 1 on switched Regatta frames (p690 1.3GHz) (Colony switch)

- Test 1 : 216 atoms SiC supercell, 128x128x128 Mesh,
  Cutoff 60 Ry (477,534 plane waves)

N Proc	MPI/SMP tasks	Time per step (s)	Performance (GFlops)	Parallel Efficiency
1024	**128/8**	**4.24**	**160**	**8%**
1024	***256/4***	***8.0***	***86***	***4%***

- Limit Case : 50% of the time spent in all to all with 128 MPI tasks;
- Latency Bound
- Federation Switch speed up the calculation by a factor ~1.5
- Best mixing with colony 1 MPI / 8 SMP ;
  with federation 1 MPI / 4 SMP

Test 2 on switched Regatta frames (p690 1.3GHz) (Colony switch)

- Test 2 : 512 atoms SiC supercell, 168x168x168 Mesh,
  Cutoff 60 Ry (1,131,630 plane waves)

N Proc	MPI/SMP tasks	Time per step (s)	Performance (GFlops)	Parallel Efficiency
672	64/8	20.1	380	24%
1280	160/8	10.6	703	23%
1280	*320/4*	*22.1*	*339*	*21%*

•Limit Case : 50% of the time spent in all to all with 128 MPI tasks;

•Latency Bound

•Federation Switch speed up the calculation by a factor ~1.5

•Best mixing with colony 1 MPI / 8 SMP ;
with federation 1 MPI / 4 SMP

Test 3 on switched Regatta frames

- Test 3 : 1000 atoms SiC supercell, 256x256x256 Mesh, Cutoff 60 Ry (2,209,586 plane waves)

N Proc	MPI/SMP tasks	Time per step (s)	Performance (GFlops)	Parallel Efficiency
512	64/8	99.5	563	46%
1024	128/8	56.3	1017	43%
1024	256/4	71.9	780	31%
1232	154/8	52.1	1087	37%

- Mixed approach instrumental to obtain these results
- Federation switch ~25 % improvement

Test 1: BlueGene/L (prototype 500MHz)/Mesh

- Test 1 : 216 atoms SiC supercell, 128x128x128 Mesh,
  Cutoff 60 Ry (477,534 plane waves)

N Proc	**TaskGroups**	**Time per step (s)**	**Parallel Efficiency**
8	**1**	**90.**	**100%**
16	***1***	**45.5**	**100%**
32	***2***	**23.1**	**97%**
64	***2***	***12.1***	***97%***
128	***4***	***6.5***	***90%***
256	***4***	***3.5***	***81%***
512	***8***	***2.1***	***68%***
1024	***16***	***1.2***	***60%***

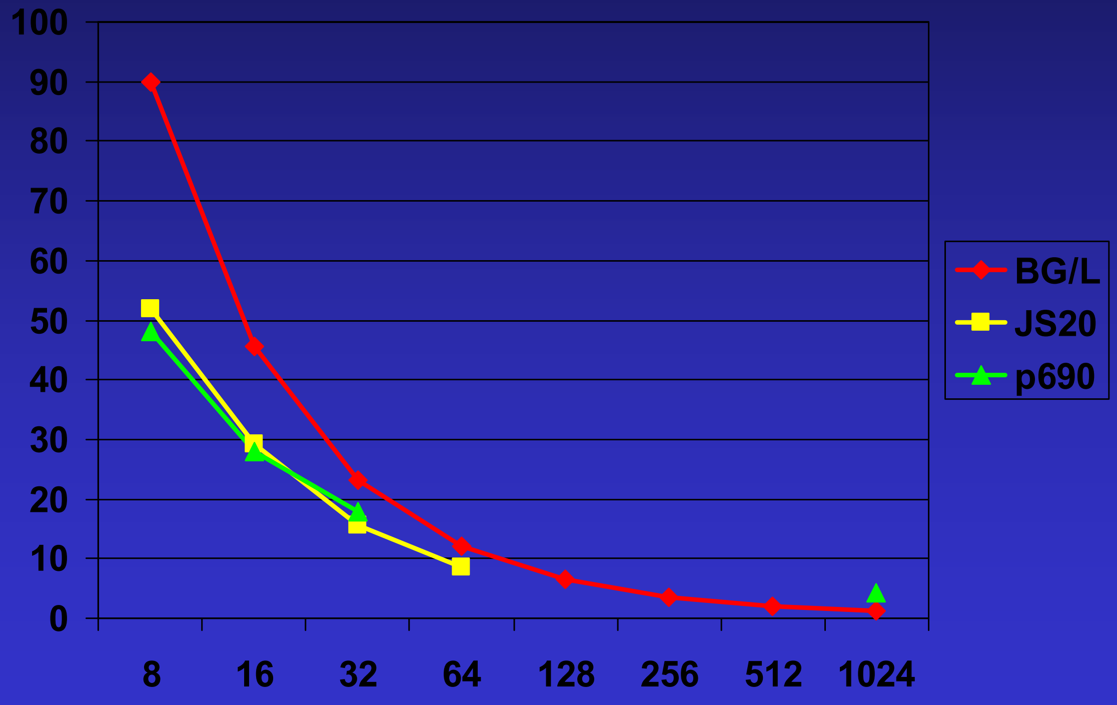
Test 1: Performance and Scaling on JS20 blades PPC970 1.6GHz

- System : 216 atoms SiC supercell, 128x128x128 Mesh,
  Cutoff 60 Ry (477,534 plane waves)

N Proc	Time/Step (s)	Performance (GFlops)	Parallel Efficiency	Relative Performance
4	102.5	6.1	100%	24%
8	52.0	12.1	98%	24%
16	29.1	19.3	80%	19%
32	15.6	27.6	76%	14%
64	8.6	50.1	51%	12%

- Performance using 2 MPI task per PPC970 node; degradation ~30%
  - Degradation ~ 1 % on p690 , ~ 70% on Xeon 2.8 GHz

Test 1: Scaling - comparison p690-JS20-BG/L



Acknowledgements

- **W. Andreoni**
- **J.J. Porta, G. Banhot and B. Walkup**
- **J. Hutter**

- **The HPCx consortium**
- **T. Kennedy**
- **A. Trew**