Scientific Computing on Parallel Machines

Cálculo científico en ordenadores paralelos

Markus Uhlmann

CIEMAT

www.ciemat.es/sweb/comfos/personal/uhlmann

UCM – Abril 2008

Schedule

Part I	Introduction to parallel programming	lecture 1
Part II	Introduction to MPI	
	General introduction	
	& MPI program structure	lecture 2
	Point-to-point communication	lecture 3,4
	Collective communication	lecture 5
	2D Poisson example	lecture 6,7
	Non-contiguous data & Mixed datatypes	lecture 7
	Virtual topologies & Communication subsets	lecture 8
	Use of linear algebra libraries	lecture 9
	Extensions	lecture 10

Part II Introduction to MPI

Turbulent flow of incompressible fluids

Navier-Stokes equations

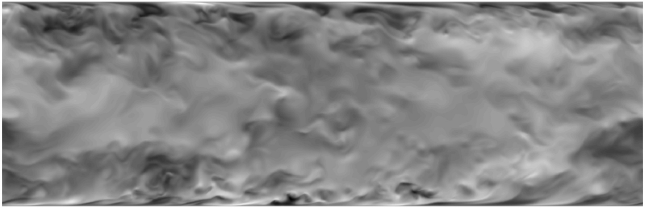
$$
\begin{aligned}
\partial_t \mathbf{u} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} + \nabla p &= \nu \nabla^2 \mathbf{u} \\
\nabla \cdot \mathbf{u} &= 0
\end{aligned}
$$

- non-linearity gives rise to chaotic flow state (turbulence)
- broad range of spatial and temporal scales
- high resolution requirements for realistic Reynolds numbers
- direct numerical simulation only possible for idealized configurations

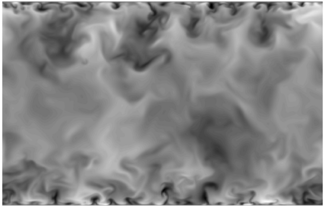
Example: turbulent field in plane channel flow

Planes from instantaneous field

$\longrightarrow$ flow direction



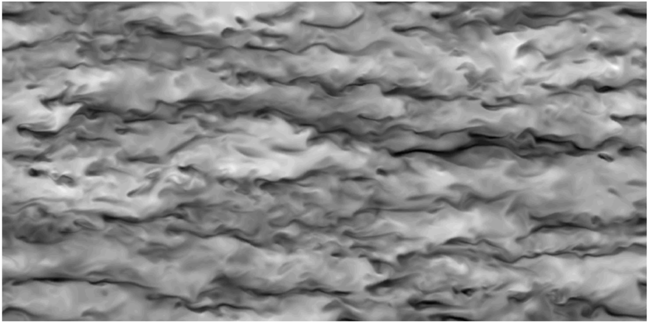
$\otimes$ flow direction



Example: turbulent field in plane channel flow

A plane near the wall

$\longrightarrow$ flow direction



A plane at the centerline

$\longrightarrow$ flow direction

Numerical solution techniques for Navier-Stokes

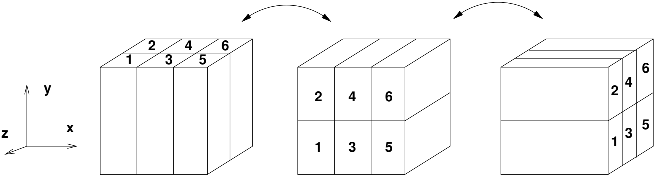
Pseudo-spectral methods

- spatial discretization by Fourier / orth. polynomial expansions
- high resolving efficiency
- efficient algorithms through FFT
- but: global data dependencies
- operations can often be decoupled along (some) coordinate directions

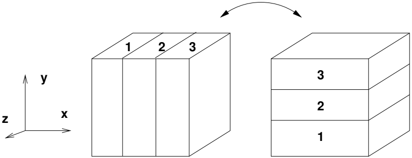
$\longrightarrow$ instead of parallel FFT: global transpose methods

Domain decomposition in spectral methods

Pencil model

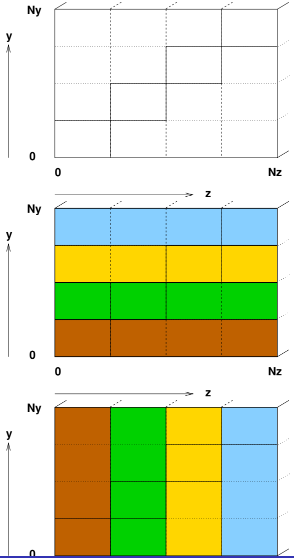


Slice model



- requires global data transpose at each time step

Data transpose technique in spectral methods



Slice data model

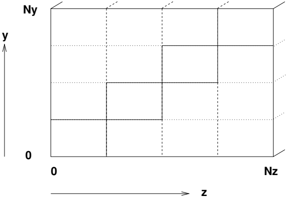
- from y-cut
- to z-cut

$\longrightarrow$ all-to-all exchange

Data transpose technique in spectral methods (2)

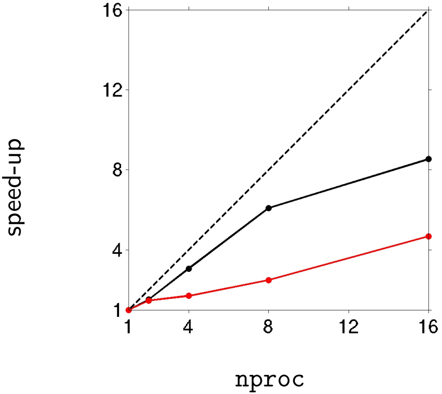
Slice data model

- improved ordering (no "cascading")




```
do ip=0,size-1
   ips=mod(rank+ip,size)
   ipr=mod(size+rank-ip,size)
   tmp=u(1:nx,kbeg(ips):kend(ips),jbeg(rank):jend(rank))
   if(ips.ne.rank)then
      • send work to ips
      • receive u(1:nx,kbeg(rank):kend(rank),jbeg(ipr):jend(ipr))
        from ipr
      → MPI_SENDRECV
   else
     • store work locally
   endif
```

Plane channel flow spectral code – speed-up



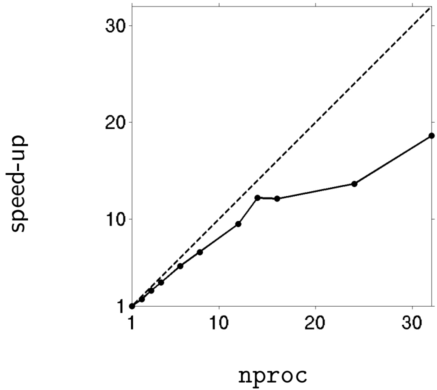
Fourier(2)-Chebyshev:

$128 \times 97 \times 128$

Xeon-Infiniband
Xeon-Gbit Ethernet
`lince.ciemat.es`

$\Rightarrow$ bandwidth is important due to high message volume

Square duct flow spectral code – speed-up



Fourier-Chebyshev(2):

128×97×97

Xeon-Infiniband
`lince.ciemat.es`

Discrete orthogonal wavelet transform

Purpose

- compression of data (MPEG-4)
- basis functions for PDE solving
- data analysis w.r.t. space and scale simultaneously (MRA)

Definition (in 3 dimensions)

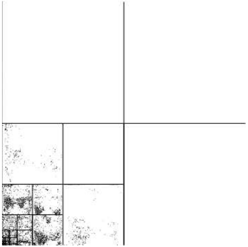
- $f(\mathbf{x}) = \sum_{j=0}^{J-1} \sum_{q=1}^{7} \sum_{s=0}^{2^j-1} \sum_{k=0}^{2^j-1} \sum_{l=0}^{2^j-1} d_{s,k,l}^{q,j} \cdot \psi_{s,k,l}^{q,j}(\mathbf{x}) + \text{average}$
- transform of $N^3 = 2^{3J}$ data $\rightarrow$ N^3 coefficients

DWT – compression

Fingerprint example (2 space dimensions)

signal

coefficients

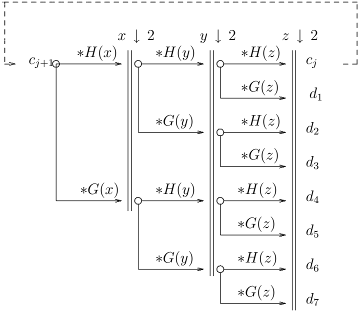


reconstruction

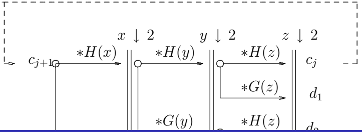
- reconstruction uses only 1.5% of the coefficients

DWT – fast algorithm

Mallat algorithm – parallel



- convolution with filters H, G (involves FFT)
- down-sampling by factor 2
- directions treated sequentially
- $\mathcal{O}(N^3 \log_2(N))$ operations
- use the slice model
- need global transpose (●)



Parallel 3D DWT – active processors

Example with 8 processors:

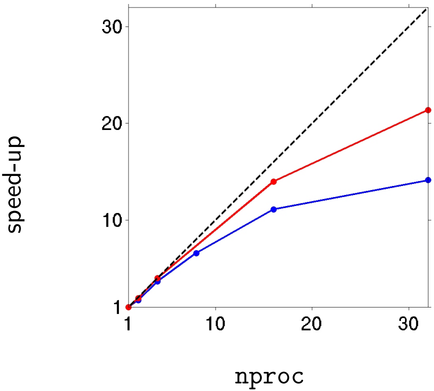
	0	1	2	3	4	5	6	7	*ip*
$J-1$	●	●	●	●	●	●	●	●	
				⋮					
3	●	●	●	●	●	●	●	●	
2	●	○	●	○	●	○	●	○	
1	●	○	○	○	●	○	○	○	
0	●	○	○	○	○	○	○	○	
voice j									

- suppose processors are powers of 2: `nproc`$=2^P$
- when $j < P$ need to “retire” processors

$\longrightarrow$ additional communication

& complex storage scheme

3D discrete wavelet transform – speed-up



no of modes:

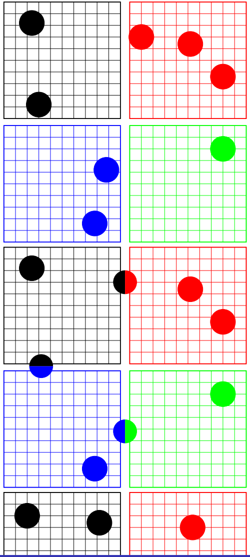
128^3

Cray T3E

IBM SP2

- reasonable speed-up can be achieved

Fluid flow with suspended particles



Finite-difference method

- 3D Cartesian domain decomposition
- each process treats particles in its own sub-domain
- particles can cross domain boundaries

$\longrightarrow$ additional communication

& shared work while overlapping

$\rightsquigarrow$ possible load-balancing problems

Fluid flow with suspended particles – protocol

Particle communication in two phases

- Phase I: exchange single integer with all neighbors
  $\longrightarrow$ number of particles which need to be communicated between pairs of processes
- Phase II: exchange actual data with corresponding neighbors
  $\longrightarrow$ data packed into single buffer, length is known

Fluid flow with suspended particles – protocol

Implementation of Phase I

• `nsend(1:8)` number of particles to be sent

```
do i=1,8
   idest=iseq_send(i)
   call MPI_ISEND(nsend(idest),1,...
          neighbor(idest),...)
   isrc=iseq_recv(i)
   call MPI_IRECV(nrecv(isrc),1,...
          neighbor(isrc),...)
enddo
```

→ `nrecv(1:8)` number of particles to receive

6	**7**	**8**
5	**myrank**	**1**
4	**3**	**2**

Fluid flow with suspended particles – protocol (2)

Implementation of Phase II

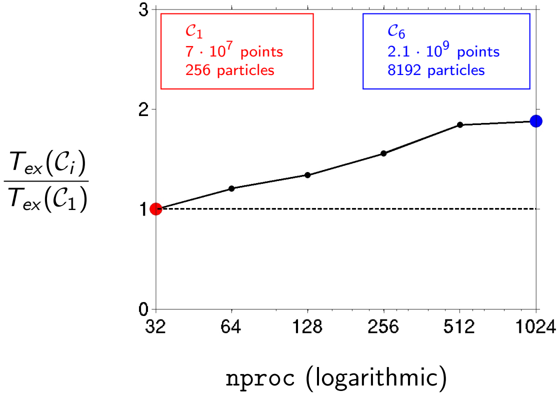
- `outbuf(nmax,1:8)` particle data to send

```
do i=1,8
   idest=iseq_send(i)
   if(nsend(idest).gt.0)
 & call MPI_ISEND(outbuf(1,idest),1,...
 &        neighbor(idest),...)
   isrc=iseq_recv(i)
   if(nrecv(isrc).gt.0)
 & call MPI_IRECV(inbuf(1,isrc),1,...
 &        neighbor(isrc),...)
enddo
```

→ `inbuf(nmax,1:8)` received data

6	**7**	**8**
5	**myrank**	**1**
4	**3**	**2**

Channel flow with suspended particles – scaling



MareNostrum

IBM PowerPC & Myrinet

- reasonable increase over a 32-fold scale-up

Where to go from here

Other parallel linear algebra tools

- SuperLU: sparse linear system solver
- Hypre: sparse linear system solver

Parallel PDE solver packages

- PETSc: Portable, Extensible Toolkit for SciComp
- SAMRAI: Structured Adaptive Mesh Refinement

Other parallel tools

- ParMETIS: unstructured mesh partitioning library

Acknowledgement

This lecture heavily draws from the following resources:

- NCSA online course (Introduction to MPI)
- MPI: the complete reference (online book)
- Documentation of the MPICH implementation of the MPI-1 standard (html) and the MPI-2 standard (pdf)