HECToR

HIGH END COMPUTING TERASCALE RESOURCE

A Research Councils UK High End Computing Service

Updating Domain Decomposition Algorithm for Incompact3D

Dr Ning Li

Numerical Algorithms Group Ltd, HECToR CSE

23-24 September 2009

HECToR dCSE support technical meeting, Oxford

Presentation Outline

- Incompact3D - Background information
- Old 1D domain decomposition
- New 2D domain decomposition
  - Concept
  - Implementation details
  - Library design
  - Performance issues
  - Ongoing and future works

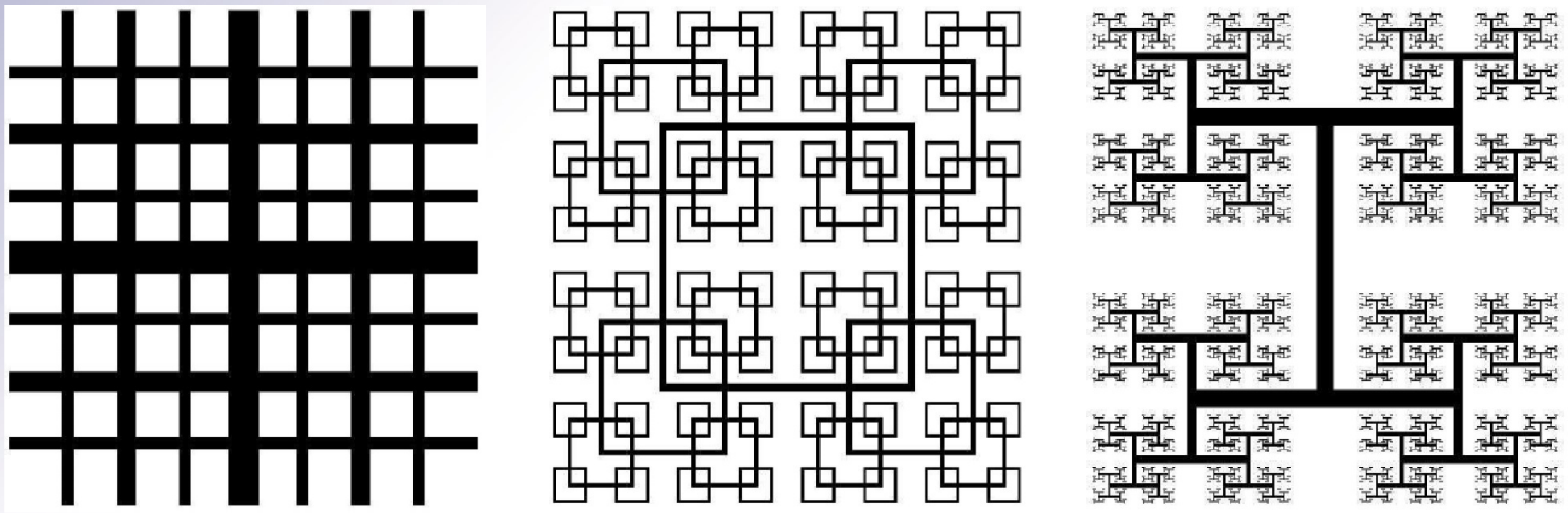
About This dCSE Project

- CFD code Incompact3D
- Turbulence, Mixing and Flow Control group at Imperial College
- PI - Prof. Christos Vassilicos
- Main code author – Dr. Sylvain Laizet
- 16-month work funded

HECToR

RESEARCH COUNCILS UK

Incompact3D - Background Information



- Direct Numerical Simulation (DNS)
- Flow passing through fractal geometry
- Billions of mesh points required to resolve smallest scale

HECToR

RESEARCH COUNCILS UK

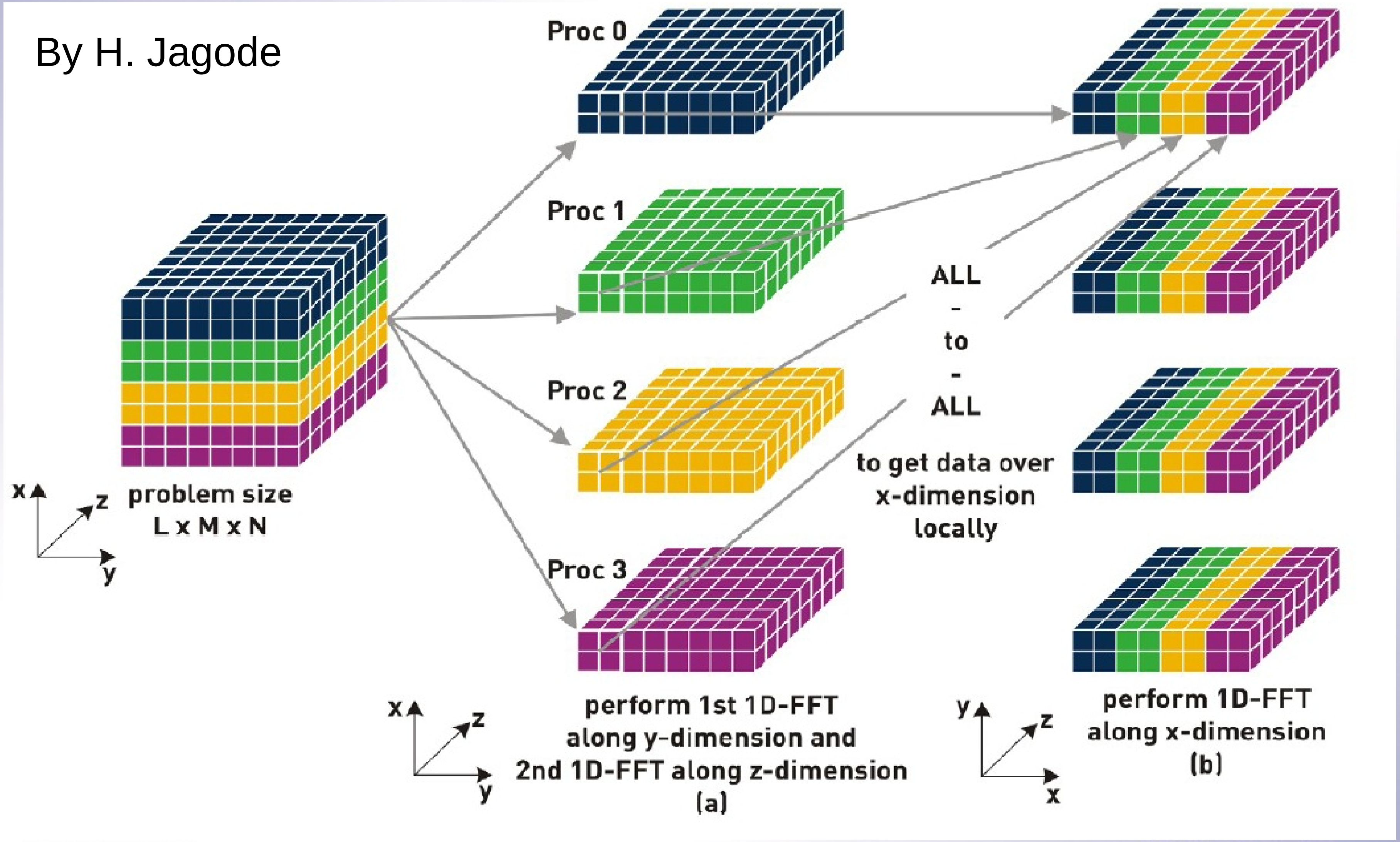
Implicit Schemes – Compact Finite Difference

- A compact scheme is inherently implicit
  - This applies to spatial derivative and interpolation calculations
  - $af'_{i-1} + bf'_{i} + cf'_{i+1} = RHS$
  - All values along a global mesh line has to be solved together
  - Tri-diagonal linear solver is fast and easy in local memory.
  - Not so in parallel
  - Relies on parallel library (such as ScaLAPACK)
  - Not easy to use and not so efficient

Implicit Schemes – FFT

- FFT applies to spectral method
- Many finite difference/volume CFD codes use FFT to solve the pressure Poisson problem
  - Multiple-dimension FFT equivalent to a family of 1D FFTs.
  - 1D FFT has to go through all values along a global mesh line.
  - If they are not all local, parallel 1D FFT library required.

Exisiting 1D Decomposition



1D Decomposition Limitation

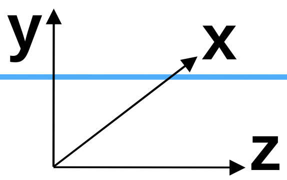
- For N^3 mesh, N_proc < N
- Planned simulations
  - Typical mesh size 2048*512*512
  - N_proc up to 512 only
  - 200000 time steps at 4 seconds per step
  - 26 days (excluding queueing time)

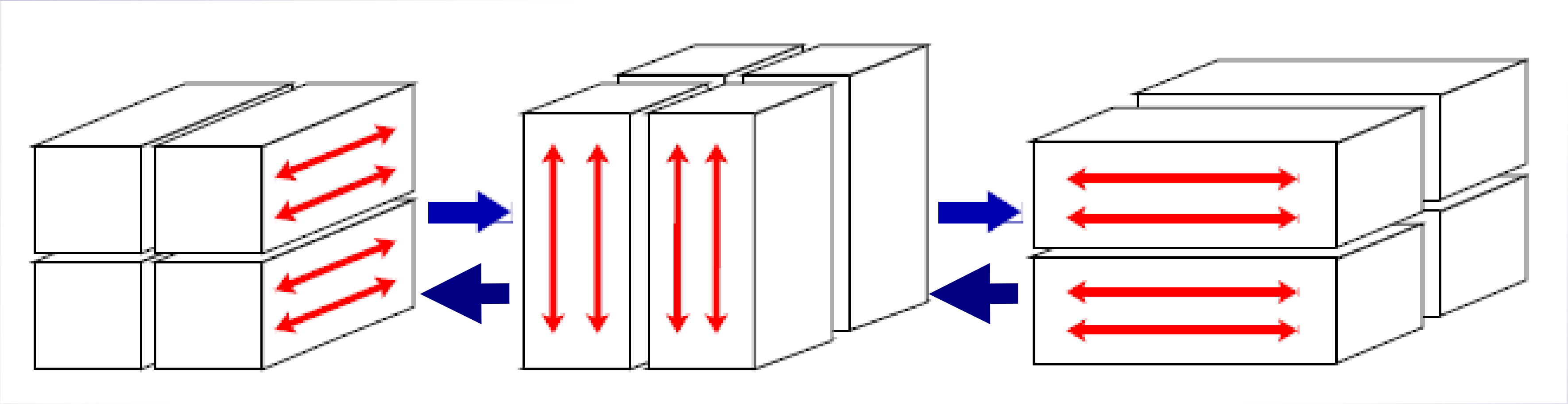
  - For larger problems, it is also likely to hit the memory limit

HECToR

RESEARCH COUNCILS UK

2D Decomposition

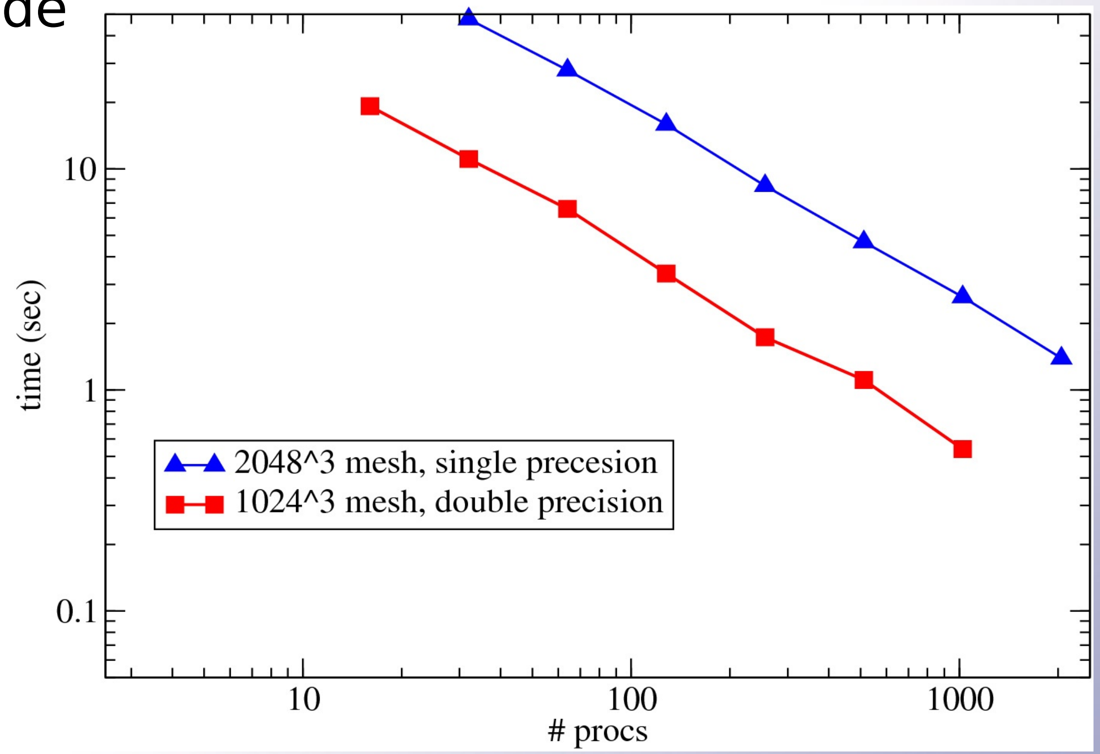




- Also known as pencil decomposition
- Local operations in one dimension at a time; then transpose
- Repeat to form a loop
- Constraint relaxed to N_proc < N^2

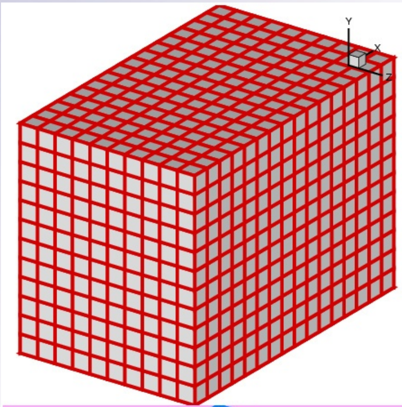
Related Works

- Open-source P3DFFT library by Pekurovsky
  - 3D FFT interface for applications
  - Using 2D decomposition internally
  - Delegate 1D FFT to established 3rd party library
- Turbulence research by Yeung, et al.
  - Spectral DNS code
  - Using P3DFFT

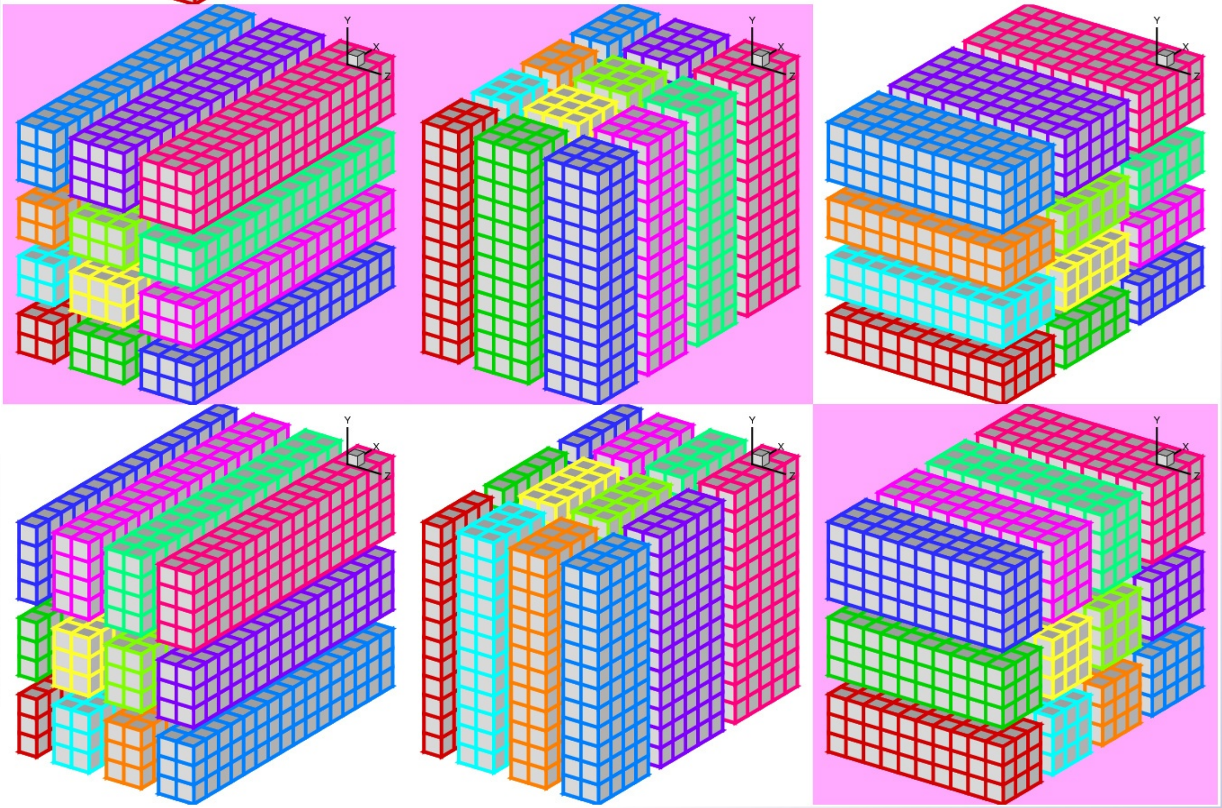


HECToR

2D Decomposition Example

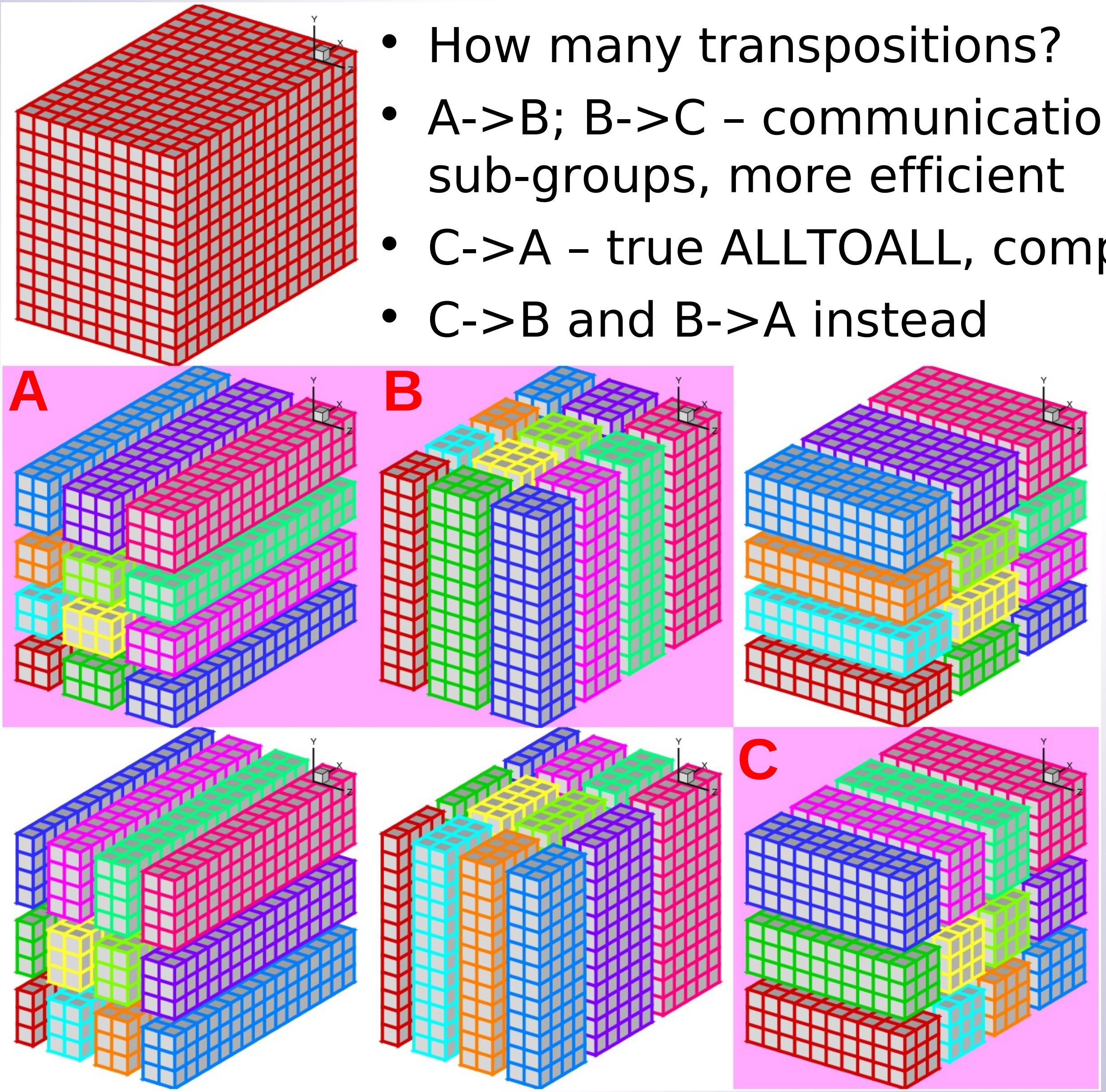


- 17*13*11 global mesh
- 4*3 processor grid
- Uneven distribution fully supported
- However, consider load balance
- MPI_ALLTOALL faster than ALLTOALLV?

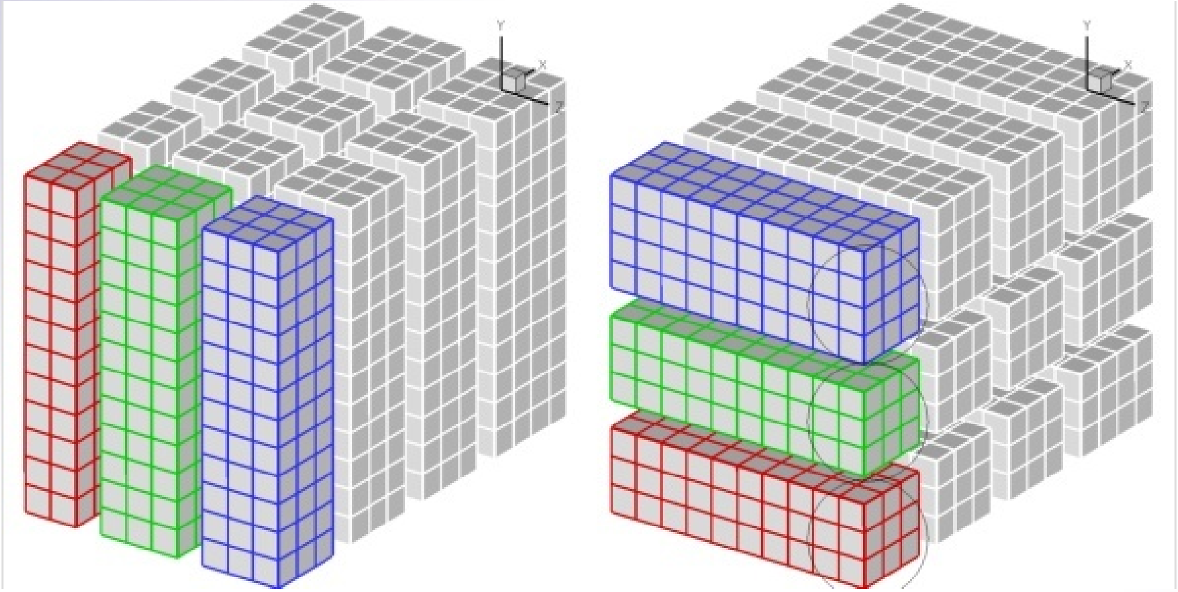


2D Decomposition Example

- How many transpositions?
- A->B; B->C – communication among sub-groups, more efficient
- C->A – true ALLTOALL, complex to code
- C->B and B->A instead



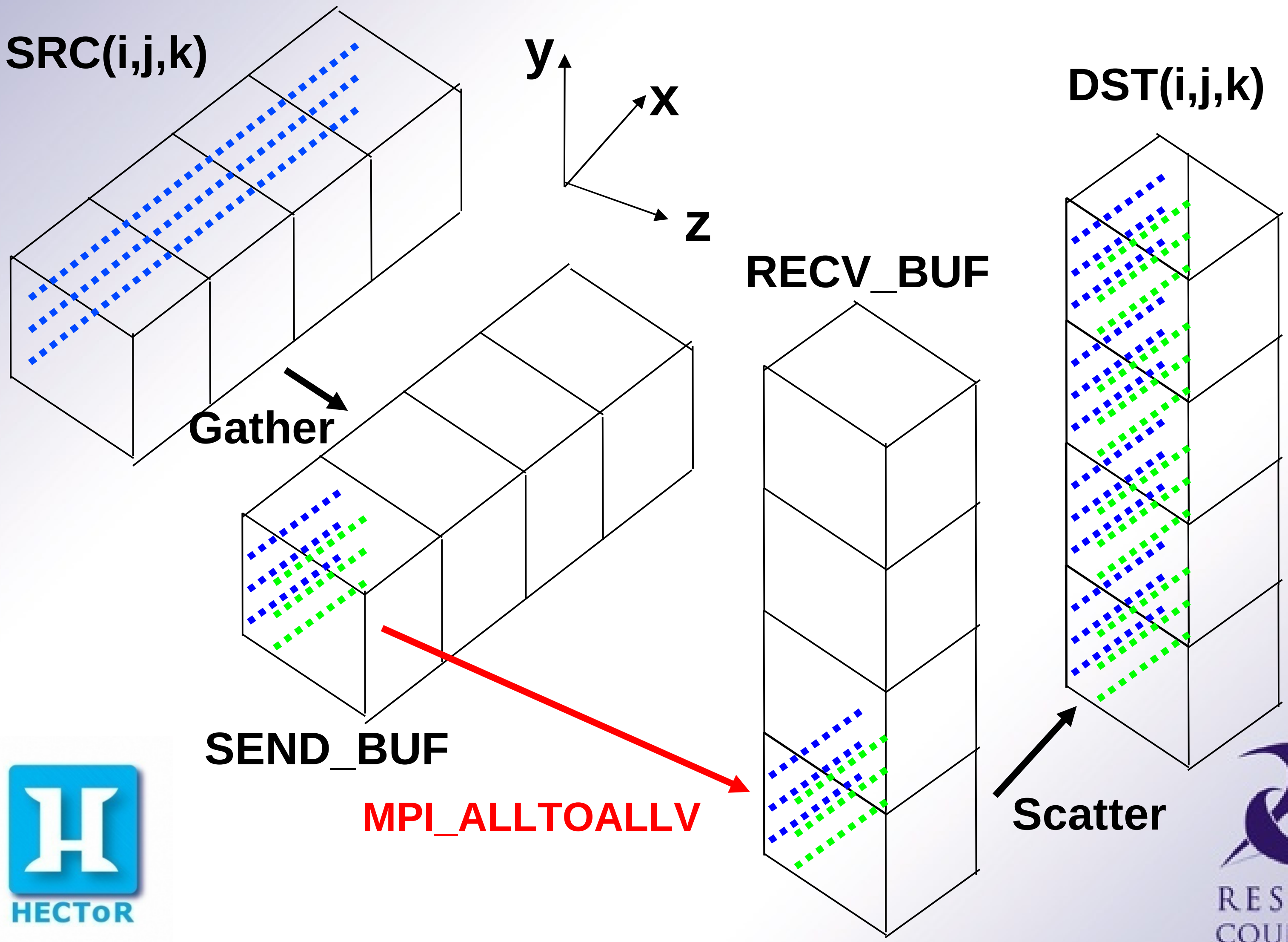
Using MPI_ALLTOALL(V)



MPI_ALLTOALL(sendbuf, sendcount, sendtype,
recvbuf, recvcount, recvtype, comm)

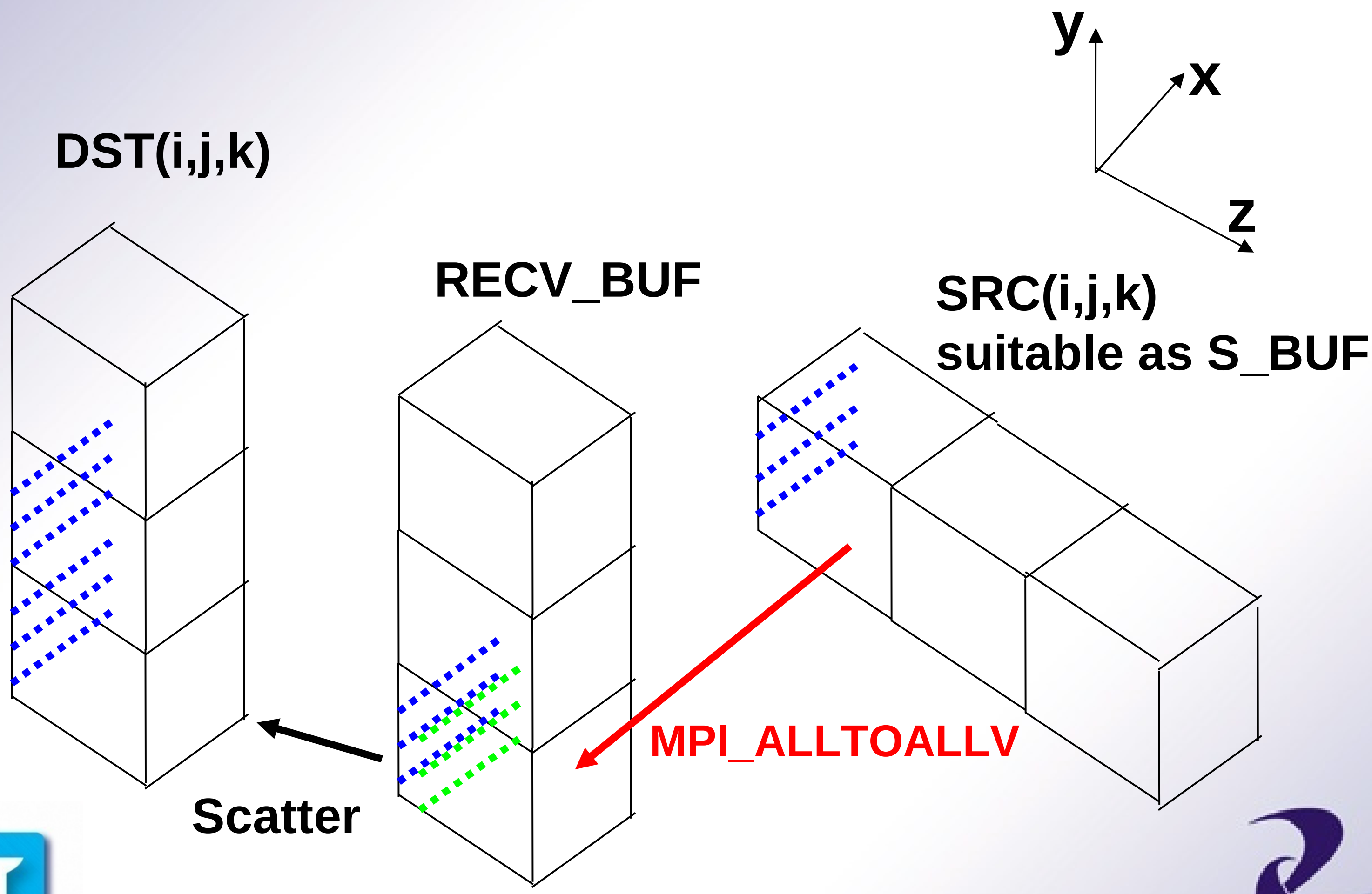
MPI_ALLTOALLV(sendbuf, sendcounts, sdispls,
sendtype, recvbuf, recvcounts, rdispls,
recvtype, comm)

Gather/Scatter Data for ALLTOALLV Buffers



Transpose from X-pencil to Y-pencil

Gather/Scatter Data for ALLTOALLV Buffers



Transpose from Z-pencil to Y-pencil

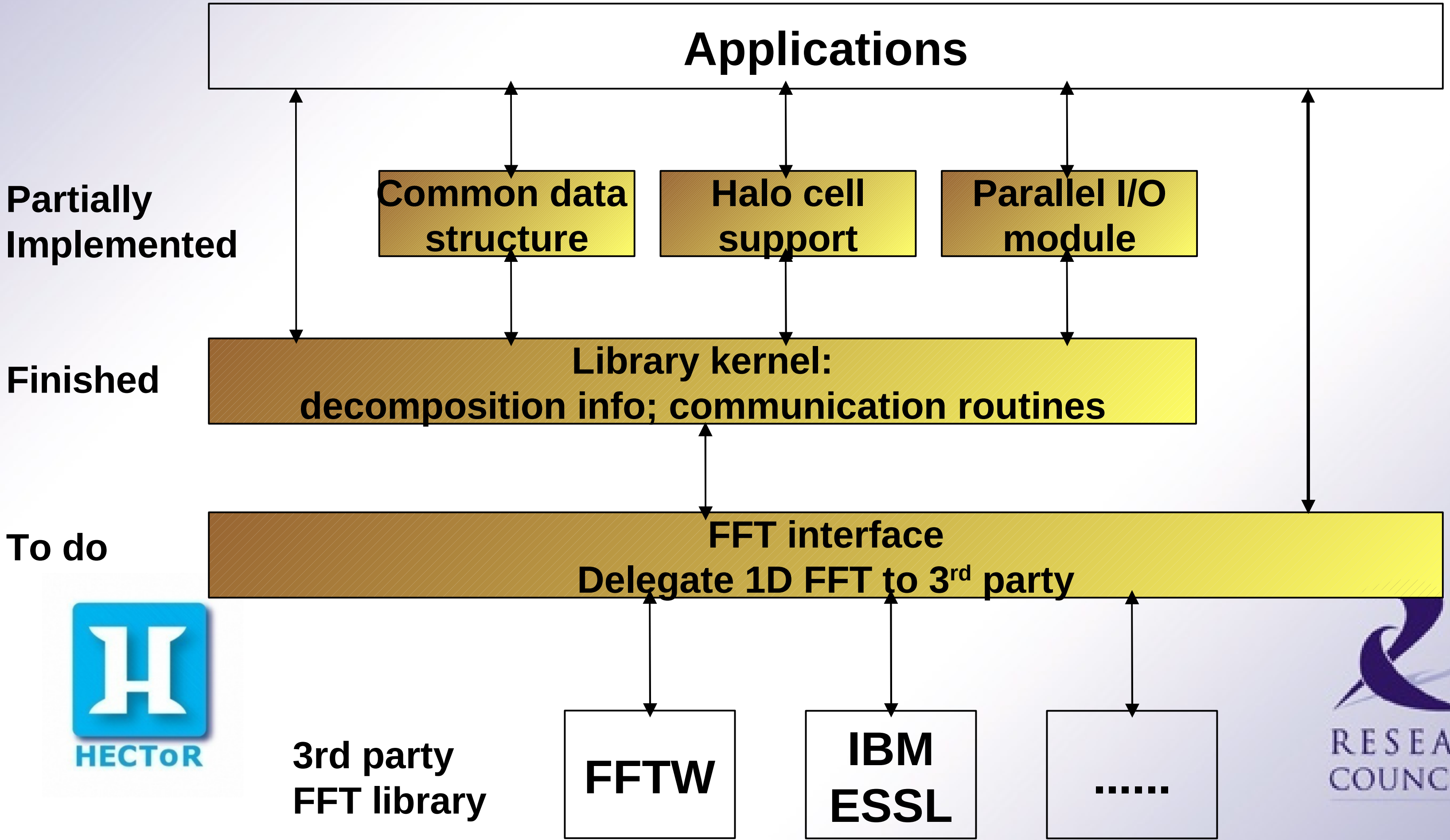
HECToR

RESEARCH COUNCILS UK

Library Design

Implement as a library:

Reusable; hide communication details



Sample Application

```
USE decomp_2d
INTEGER, PARAMETER :: nx=96,ny=48,nz=48
INTEGER, PRRAMETER :: p_row=4,p_col=3
REAL, ALLOCATABLE, DIMENSION(:,:,:) :: ux,uy,uz

CALL MPI_INIT(ierror)
CALL SETUP_2D_DECOMP(nx,ny,nz,p_row,p_col)

ALLOCATE(ux(xstart(1):xend(1),xstart(2):xend(2),xstart(3):xend(3))
ALLOCATE(uy(ystart(1):yend(1),ystart(2):yend(2),ystart(3):yend(3))
ALLOCATE(uz(zstart(1):zend(1),zstart(2):zend(2),zstart(3):zend(3))

!......
```

Global data size

2D processor grid

Set up 2D decomposition

Initialise application data structure

HECToR

RESEARCH COUNCILS UK

Sample Application (continued)

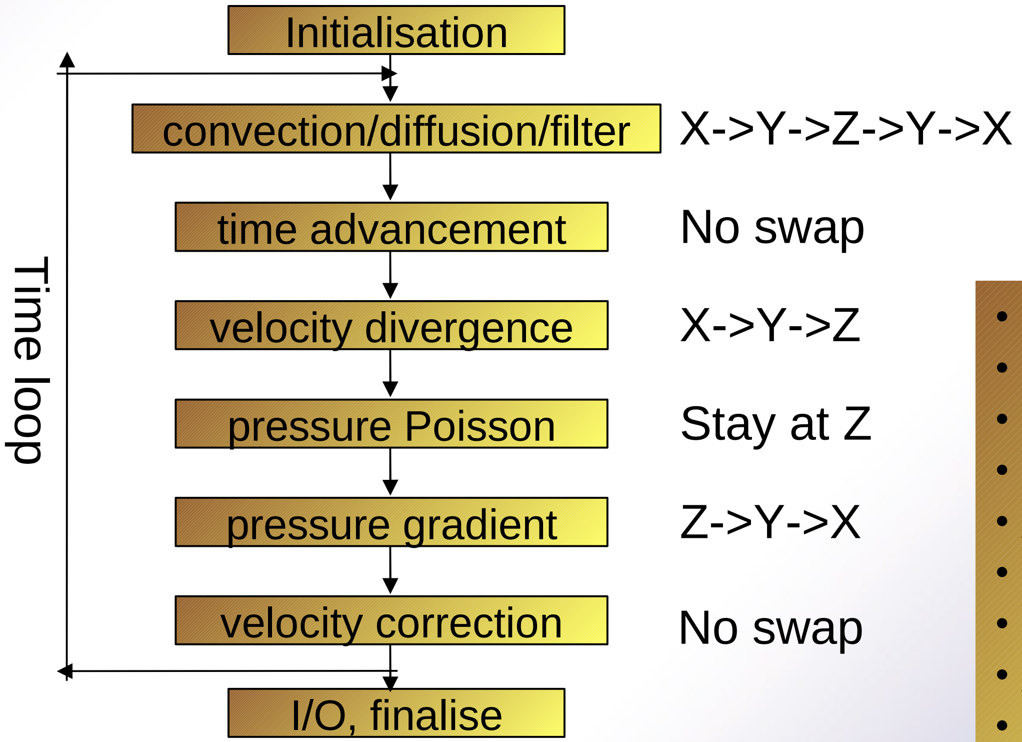
```
!......

! do something on ux
CALL TRANSPOSE_X_TO_Y(ux,uy)

! do something on uy
CALL TRANSPOSE_Y_TO_Z(uy,uz)

! do something on uz
CALL TRANSPOSE_Z_TO_Y(uz,uy)

! do something on uy
CALL TRANSPOSE_Y_TO_X(uy,ux)
CALL MPIIO_WRITE(nx,ny,nz,ux,'ux.dat')

CALL CLEAN_2D_DECOMP
DEALLOCATE(ux,uy,uz)
CALL MPI_FINALIZE(ierror)

END
```

HECToR

RESEARCH COUNCILS UK

What Application Developers Need to Do?

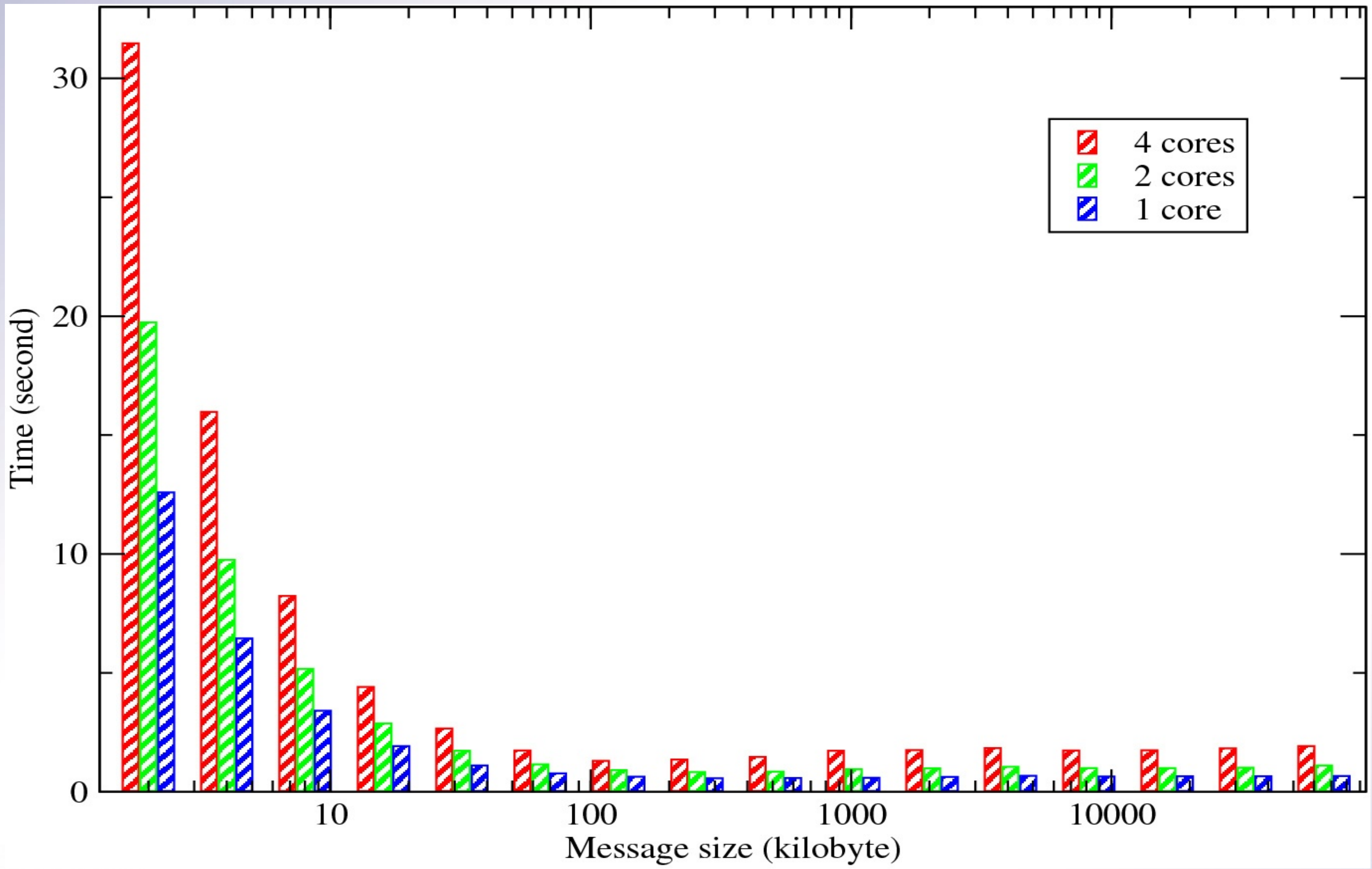
- Understand decomposition concept
- Understand library interface
- Group calculations based on decomposition
- Minimise the number of transpositions
- Incompact3D flow chart



- Calculate X derivatives
- Combine results in temp
- Swap temp to Y pencil
- Calculate Y derivatives
- Add to temp
- Swap temp to Z pencil
- Calculate Z derivatives
- Add to temp
- Results stored in Z pencil

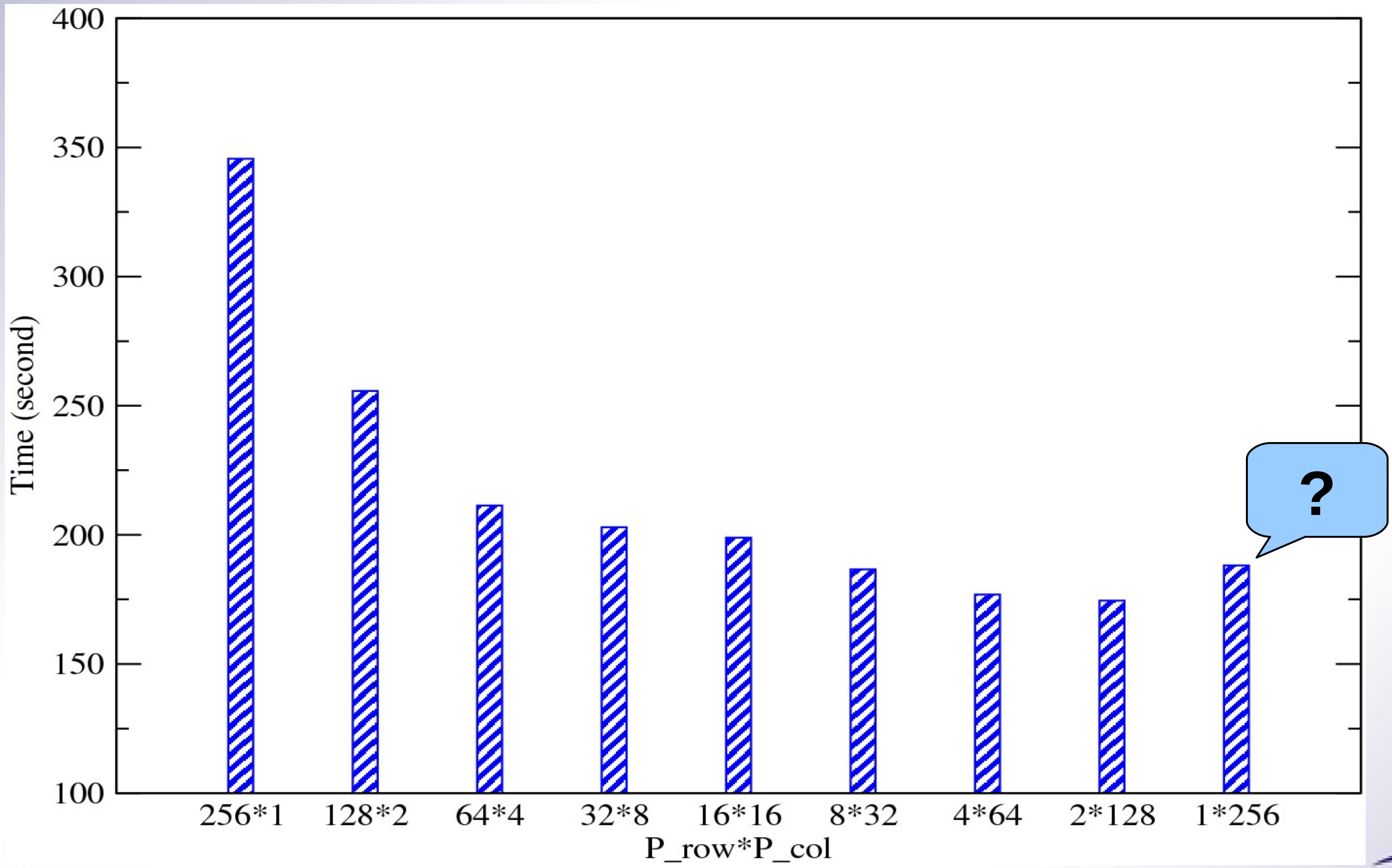
HECToR

Performance – Core per Node



- Network bandwidth is not enough.
- Possibly memory bandwidth issue as well
- Still worthwhile to use 4 cores unless application requires more memory

Performance – Shape of 2D Processor Grid



- P_row << P_col recommended on HECToR
- Ideally, P_row ≤ 4
- Cache efficiency: nx/P_row largest possible

HECToR

RESEARCH COUNCILS UK

Shared-Memory Implementation

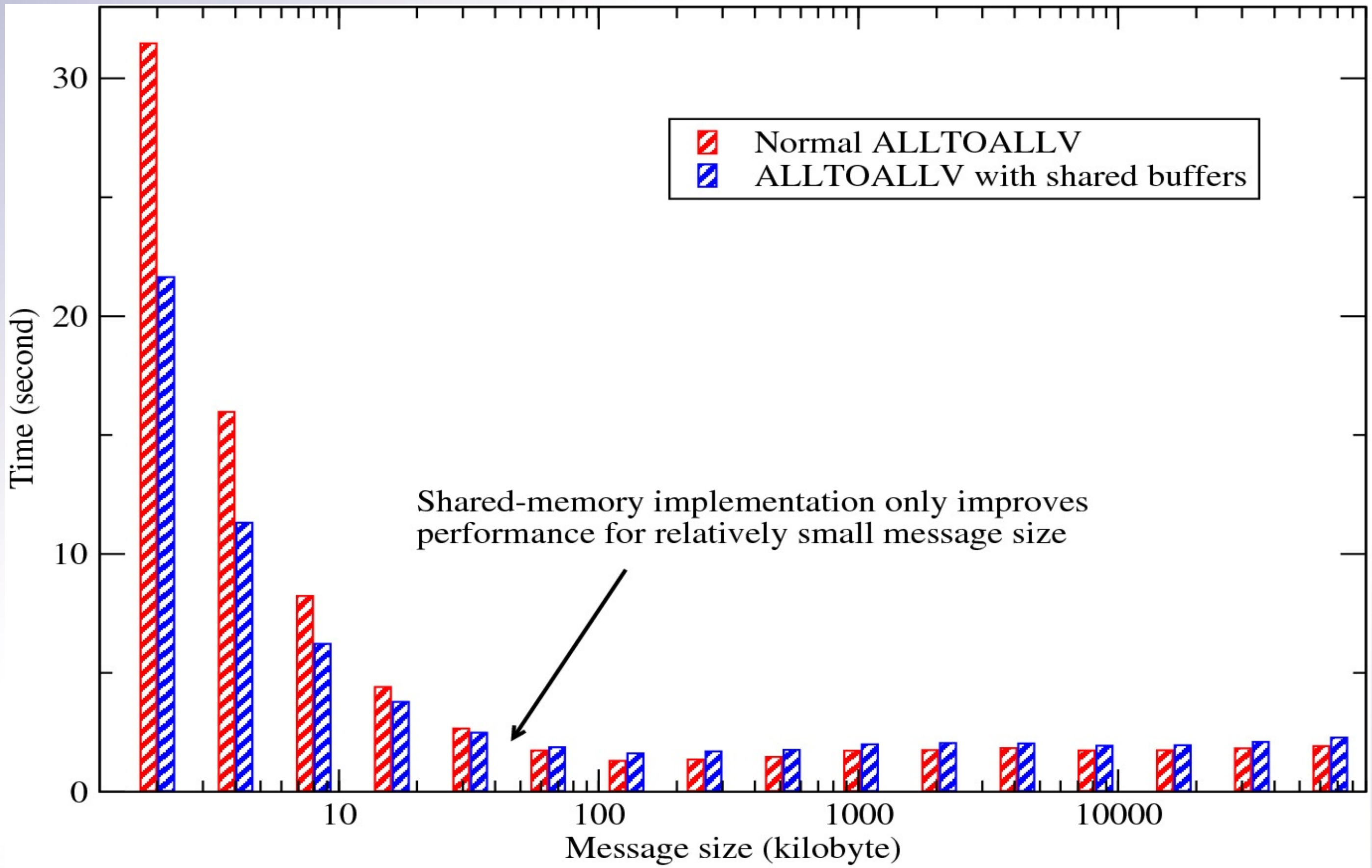
Shared-memory code (D. Tanqueray, Cray)

- ALLTOALL(V) among large number of nodes expensive.
- HECToR prefers small number of large messages.
- HECToR phase 2 has 8GB memory shared by 4 cores.
- Memory addressable by all cores.
- Cores on same node copy data to/from a shared buffer.
- Only leaders of the nodes participate communication
- This results in fewer but larger messages.
- Communication routine interface remains the same.
- Not automatically portable, but can be so.

HECToR

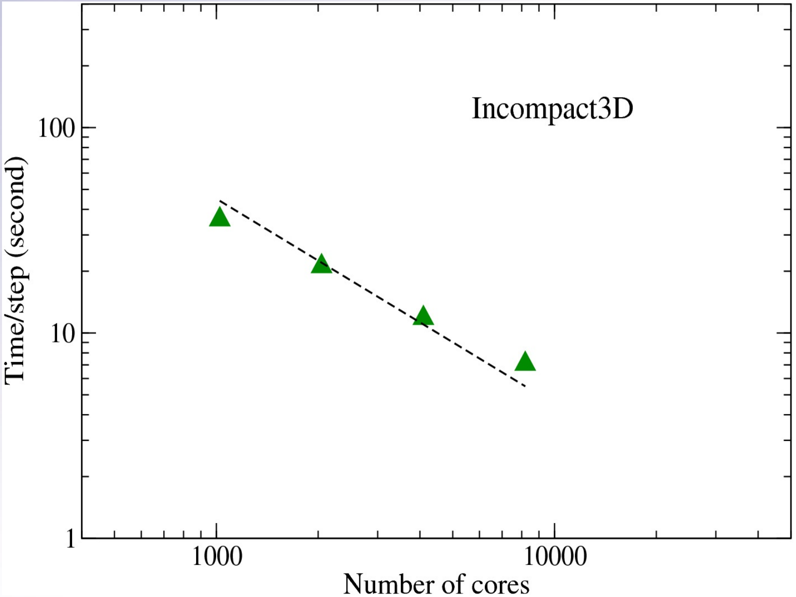
RESEARCH COUNCILS UK

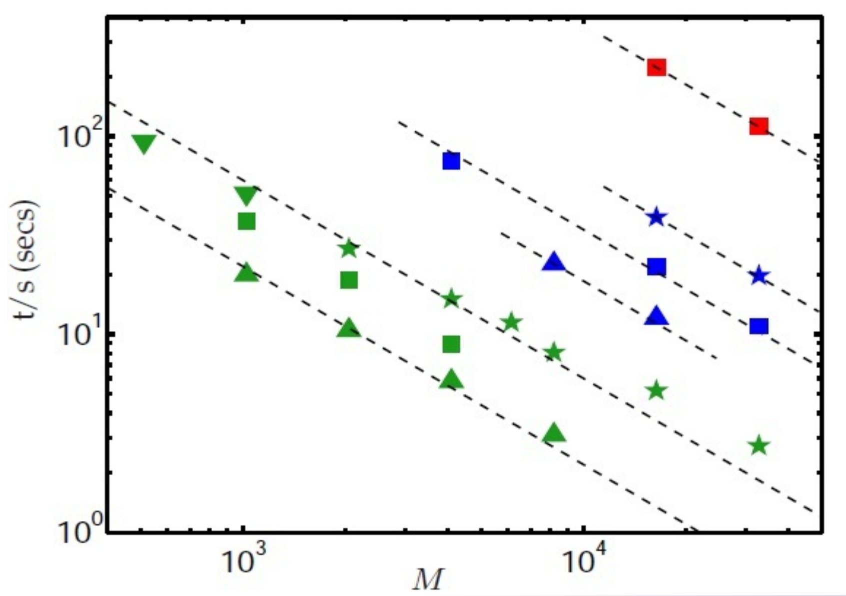
Performance - Shared-Memory



- Performance improvement for message size smaller than 64K only
- Must be more useful on HECToR Phase 2B - 12-core system

Performance - Scaling





- Going through all Incompact3D algorithms except pressure solver.
- 8 billion (2048^3) mesh points.
- Scaling factor 85%-90%.
- Application code to be optimised.

Ongoing and Future Work

- Implement FFT interface
- Validation of new Incompact3D
- Performance benchmark of new Incompact3D
- Parallel I/O and other library improvement
- Other algorithm improvements – stretching grid; filtering; new boundary conditions; etc.
- MPI/OpenMP hybrid programming?
- Other applications
  - SoFTaR – computational combustion code at Brunel University
  - Several CFD codes within UKTC

HECToR

RESEARCH COUNCILS UK