Scaling Turbulence Applications

to Thousands of Cores

David Scott
EPCC
The University of Edinburgh

Outline

Background (Collaborators)

Working with Gary Coleman and Roderick Johnstone of Southampton University

They are members of UKTC

Joachim Hein (EPCC) who proposed the work and discusses the work with me from time to time

Background (History)

3D simulation of the Ekman Boundary Layer (EBL)

Originally developed 20 years ago (G. Coleman)

There was a vector version

Then a version for HPCx
(R. Johnstone and G. Coleman)

Background (Current Domain Sizes)

x, y: horizontal

z: vertical

Nx, Ny, Nz = 420, 1260, 151 or

768, 2304, 204

Background (Transforms)

A Jacobi transform is performed on the vertical dimension (z)

Fourier transforms are performed on the horizontal dimensions (x and y)

Current Code (Vertical Transform)

Transformation of the z dimension

The y dimension is distributed across processors

On each processor loops over the y sub-range and the x dimension are performed

Within these loops a Jacobi transform is performed along the z direction

Current Code (Horizontal Transforms)

Transformation of the x and y dimensions

The z dimension is distributed across processors

On each processor a loop over the z sub-range is performed

Within this loop

* For each x a FT along the y direction is performed, then
* For each y a FT along the x direction is performed

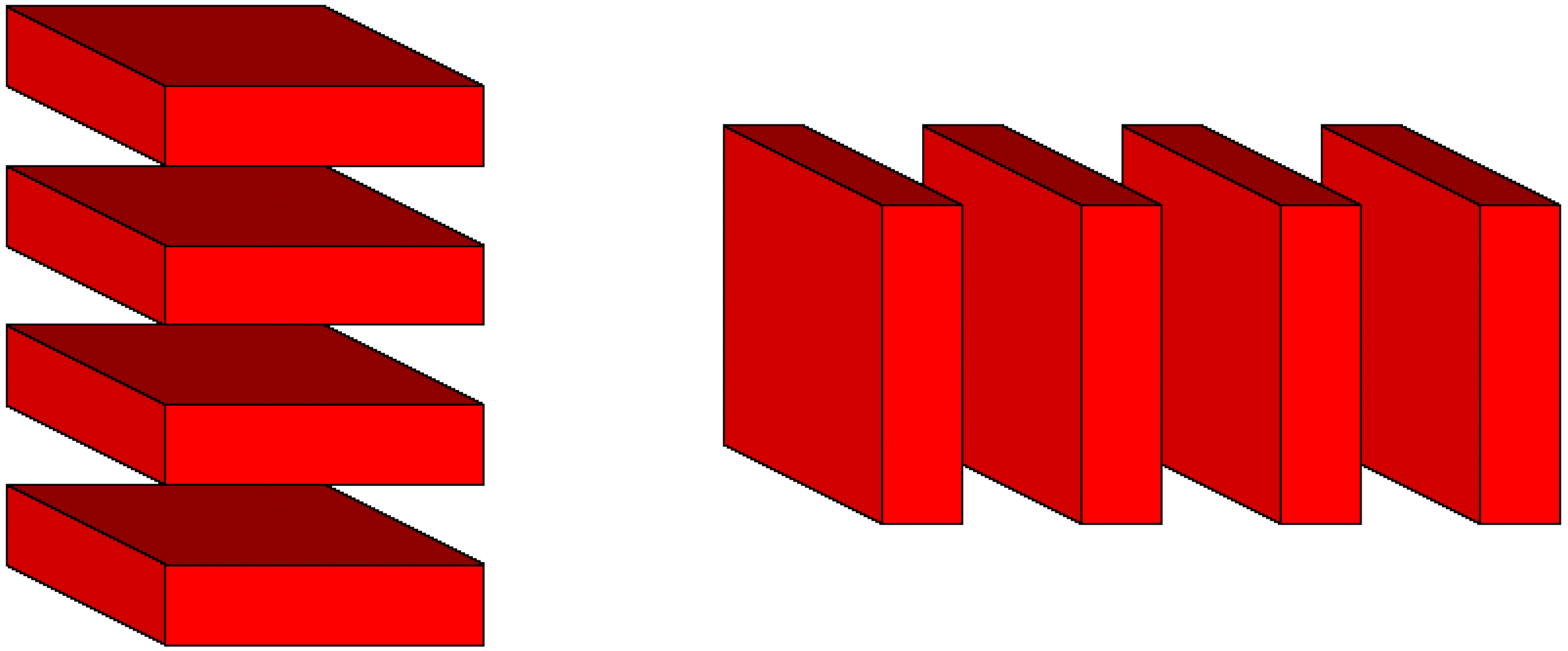
Current Code (Decomposition)

1 dimensional processor grid

Each processor gets several planes

Perform FFT in two of the three directions

Single All-to-all before performing Jacobi in third direction



Current Code (Processors)

The maximum number of processors that can be used

= max(Ny, Nz)

= Nz (for the domains chosen)

= 151 or 204

The Problem

Investigating significantly larger domains with the current code is impractical because:

* It takes too long

* Too much memory per processor is required

The Proposed Solution (Overview)

J. Hein

Use more processors and hence divide the calculation into smaller chunks in order to:

* Speed up the calculation
* Reduce the amount of memory required per processor

But this will require more data between processors

* The communication overhead will be increased

Use a 2D decomposition

The Proposed Solution (Previous Work - 1)

Cubic case has already been investigated by Hagode

- Used IBM BlueGene/L
- Fourier Transform in all directions
- Not a problem solving code
- Similar mappings for all dimensions
- Mapping depends on 2 things
  - No. of processors
  - Linear dimension (a common power of 2)

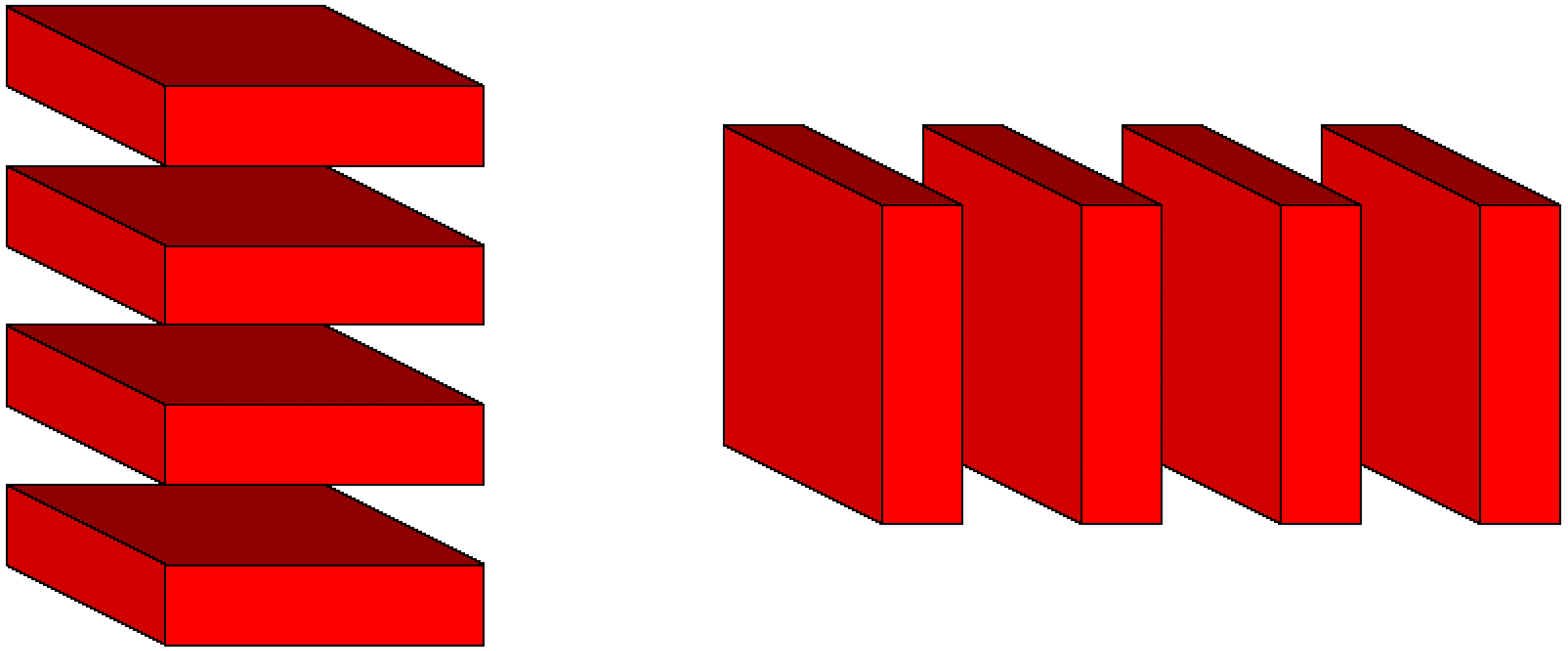
The Proposed Solution (Previous Work - 2)

1 dimensional processor grid

Each processor gets several planes

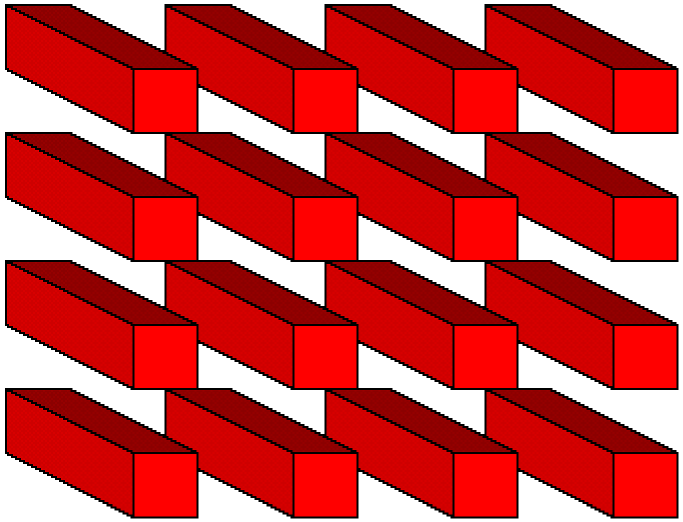
Perform FFT in two of the three directions

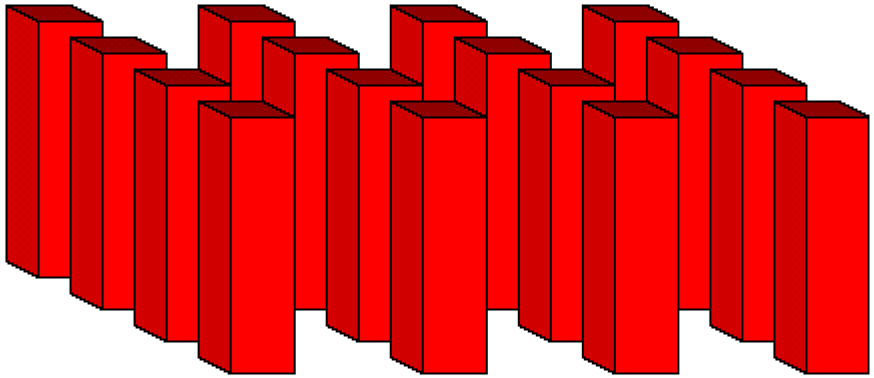
Single All-to-all before performing Jacobi in third direction



Proposed Solution (Previous Work – 3)

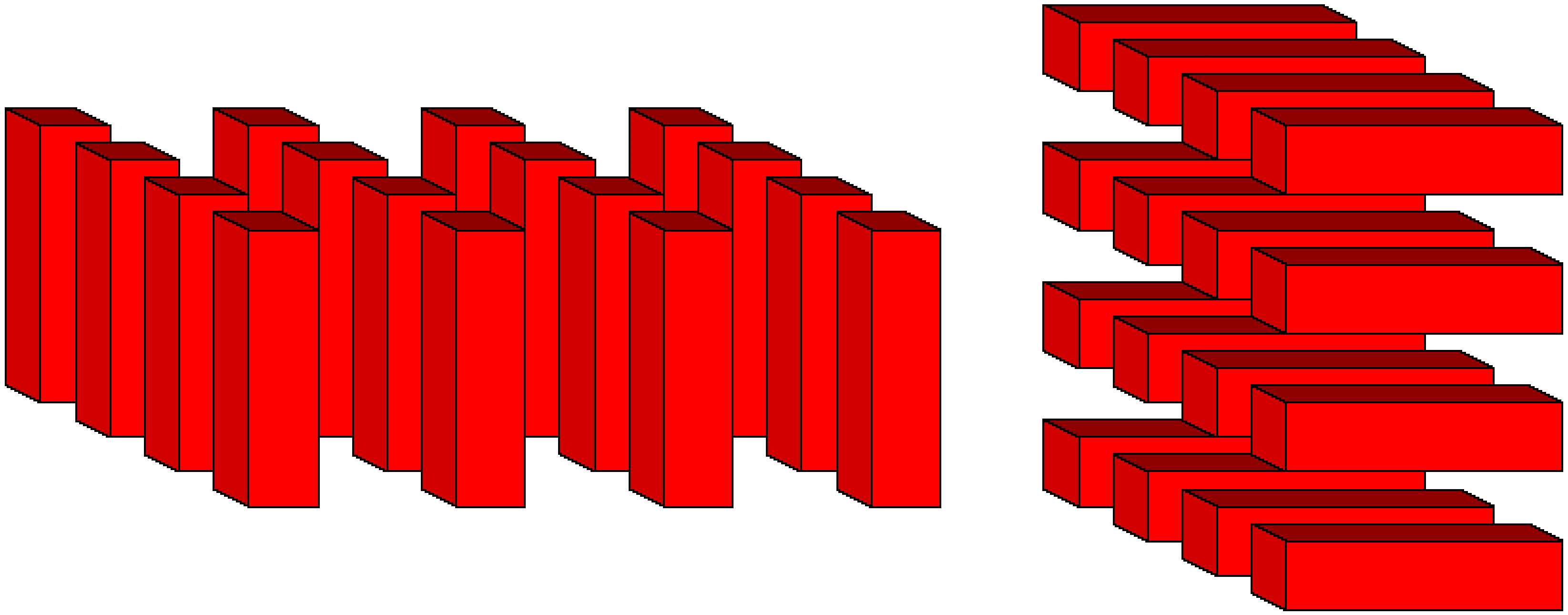
- Each processor gets a “stick” of the 3D array
- Perform FFT in 1st direction
- Perform several All-to-all transformation in the columns of the processor grid





Proposed Solution (Previous Work – 4)

- Perform FFT in the 2nd direction
- Perform All-to-all in the rows of the processor grid
- Perform 3rd FFT in the last direction



The Proposed Soln. (Mapping of a Cuboid)

EBL is a problem solving code

In the EBL case the domain is not cubic

- Distribution over processors will depend on the dimension being transformed
- 3 distinct mappings of the 3D domain to the set of processors
- Each mapping depends on 3 things
  - No. of processors
  - 2 linear dimensions (not powers of 2)

Refactoring the Code (Requirement)

The principal requirement is:

When transforming one dimension the loops over the two remaining dimensions should be at the top level.

Refactoring the Code (First Attempt)

Started from the current (EBL3) code (R. Johnstone and G. Coleman)

* Some loops were buried in subroutines
* The 1D decomposition obscured matters
* Difficult to reverse engineer
* BUT up to date

Refactoring the Code (Second Attempt)

Swapped to an earlier, serial version

* Some loops were buried in subroutines
* It needed cleaning up
* It needed updating
* BUT structure was clearer

Status (Done)

Have cleaned up and updated serial code

Have flattened code and reordered where necessary

Have written and tested code to shuffle data around on a cuboid

Have written stand alone code that:

* given the dimensions of the domain and the number of processors
* suggests processor array dimensions for each transformation

Status (To Be Done)

Partition the loops to enable mapping to processors (started)

Implement data movement (based on experimental code)

Aggregation of data on nodes

Investigation of SWT and SS3F