HECToR enabled Step Change in Turbulent Multiphase Combustion Simulation

Lucian Anton, Ning Li,
NAG,
Kai Luo,
University of Southampton

HECToR DCSE workshop
Manchester, October 4-5, 2011

HECToR

RESEARCH COUNCILS UK

Project objectives (I)

WP1: Implementation of a 2D domain decomposition algorithm

(a) To convert the base data structures for the new domain decomposition.

(b) To update the core fluid solver and run a basic validation test.

(c) To update the remaining multi-physics modules.

(d) To benchmark the new DSTAR code.

Objective: DSTAR should scale to at least 10,000 cores with 80% scaling efficiency.

WP2: Improved I/O for capability size data sets

(a) To refactor the checkpoint procedures and statistics procedures for handling large data sets

(b) using MPI-IO type of parallel I/O solution.

(c) To re-engineer the logging procedures.

(d) To port some of the common I/O routines back to the 2DECOMP&FFT library. But only if this is worthwhile.

Objective: DSTAR should be able to efficiently handle datasets for cubic grids of at least size 500.

Project objectives(II)

WP3: Code refactoring. Most of this will be done along with the code update in WP1. The remaining routines, in particular 'user routines' will be modernised using Fortran 95 as these are regularly updated by scientists who use DSTAR. Some legacy library routines may be kept in F77 form unless any significant performance issue is identified.
Objective: All user routines of DSTAR should be modernised.

WP4: Dissemination
Objective: All user routines of DSTAR should be modernised and a brief user document provided for future DSTAR users.

http://www.hector.ac.uk/cse/distributedcse/reports/dstar/

HECToR

RESEARCH COUNCILS UK

Outline

- Introduction
  - Problem description & algorithm
  - Code overview
- Mixed Mode
  - Computation acceleration
    - 2D decomposition cross over
  - Computation-Communication overlap
- Conclusions

Introduction I

- DSTAR is a combustion code (gas flow + chemical reaction, liquid droplets)

$$\frac{\partial \overline{\rho_g}}{\partial t} + \frac{\partial \left(\overline{\rho_g}\widetilde{u_{g,i}}\right)}{\partial x_i} = \overline{S_{ms}}$$

$$\frac{\partial \left(\overline{\rho_g}\widetilde{u_{g,i}}\right)}{\partial t} + \frac{\partial}{\partial x_j}\left(\overline{\rho_g}\widetilde{u_{g,i}}\widetilde{u_{g,j}} + \overline{p}\delta_{ij} - \widehat{\sigma_{ij}}\right) = \overline{S_{mo,i}} + SG_{mo,i}$$

$$\frac{\partial \widehat{E_T}}{\partial t} + \frac{\partial}{\partial x_i}\left[\left(\widehat{E_T} + \overline{p}\right)\widetilde{u_{g,i}} + \widehat{q_i} - \widetilde{u_{g,j}}\widehat{\sigma_{ij}}\right] = Q_h\overline{\omega_T} + \overline{S_{en}} + SG_{en}$$

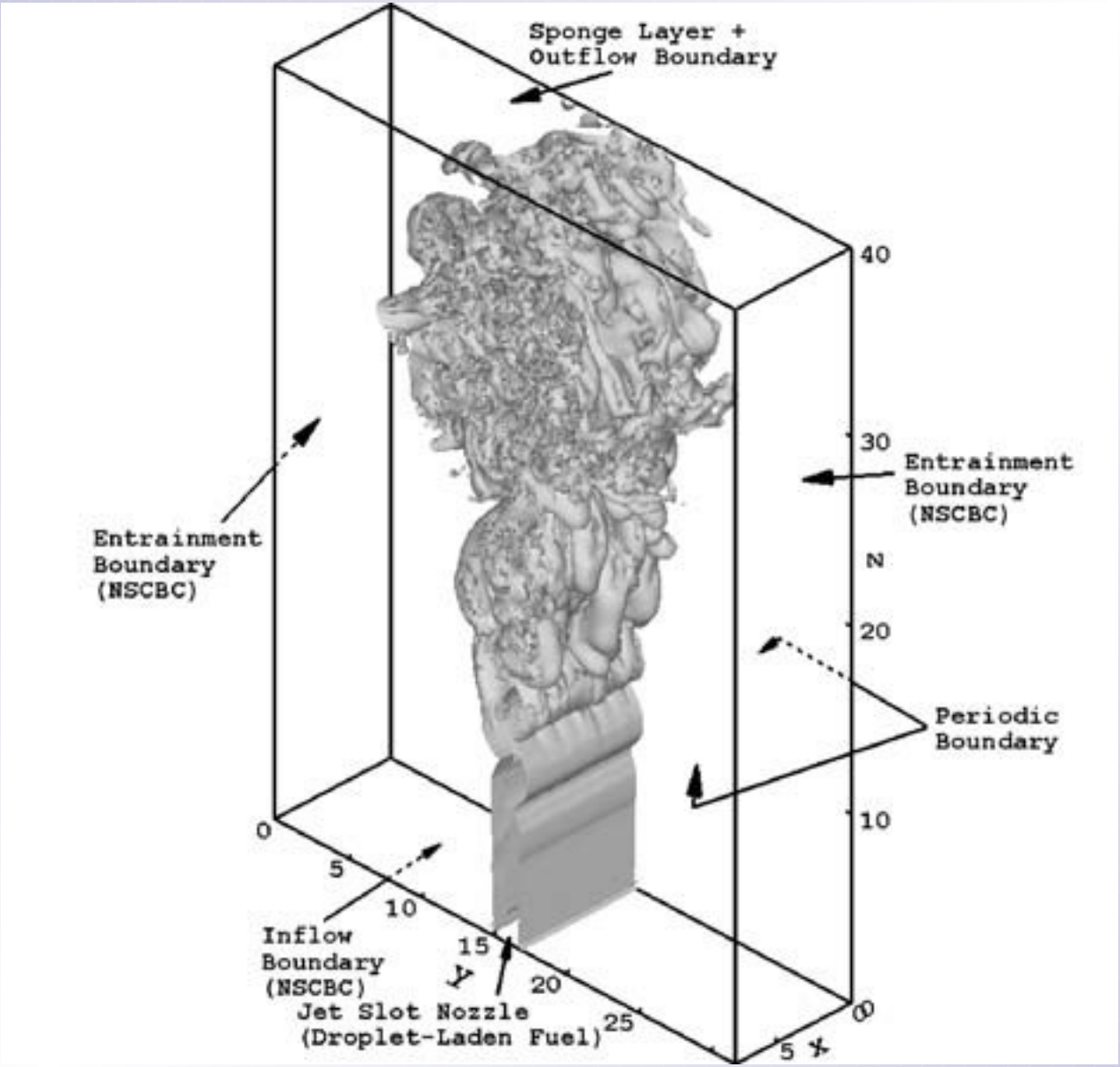
$$\frac{\partial \left(\overline{\rho_g}\widetilde{Y_n}\right)}{\partial t} + \frac{\partial}{\partial x_i}\left[\overline{\rho_g}\widetilde{u_{g,i}}\widetilde{Y_n} - \frac{1}{ReSc}\left(\widehat{\mu}\frac{\partial \widetilde{Y_n}}{\partial x_i}\right)\right] = -\upsilon_n W_n \overline{\omega_T} + SG_{sp,n}$$

$$\frac{\partial \left(\overline{\rho_g}\widetilde{Y_v}\right)}{\partial t} + \frac{\partial}{\partial x_i}\left[\overline{\rho_g}\widetilde{u_{g,i}}\widetilde{Y_v} - \frac{1}{ReSc}\left(\widehat{\mu}\frac{\partial \widetilde{Y_v}}{\partial x_i}\right)\right] = \overline{S_{ms}} + SG_{sp,v}$$

HECToR

RESEARCH COUNCILS UK

Introduction II



Algorithm(I)

- Numerical algorithm: direct numerical simulation
  - Implicit compact difference scheme for spatial derivatives
  - 3rd-order Runge-Kutta for time integration

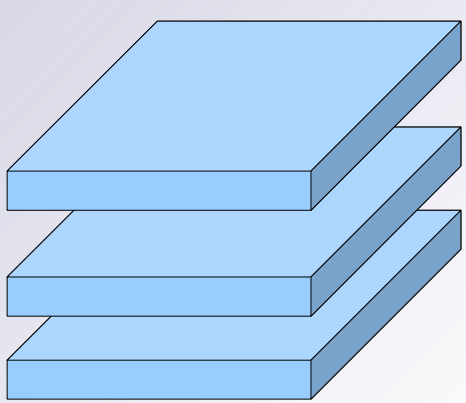
References:

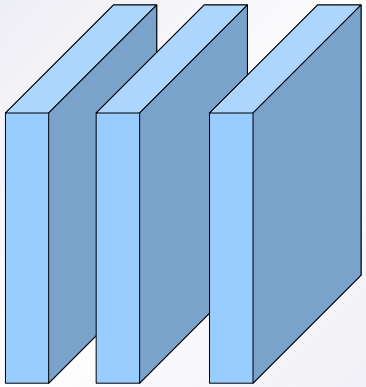
- Jun Xia, Kai H. Luo, Suresh Kumar, Flow Turbulence Combust (2008) 80:133-153
- Xia, J. and Luo, K. H.(2009) 'Conditional statistics of inert droplet effects on turbulent combustion in reacting mixing layers', Combustion Theory and Modelling, 13: 5, 901 - 920

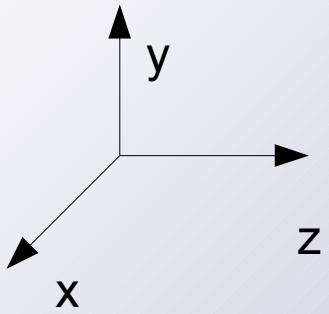
Algorithm(II)

- Implicit scheme for spatial derivatives requires boundary to boundary domains

$$\beta f'_{i-2} + \alpha f'_{i-1} + f'_i + \alpha f'_{i+1} + \beta f'_{i+2} = c\frac{f_{i+3} - f_{i-3}}{6h} + b\frac{f_{i+2} - f_{i-2}}{4h} + a\frac{f_{i+1} - f_{i-1}}{2h}$$





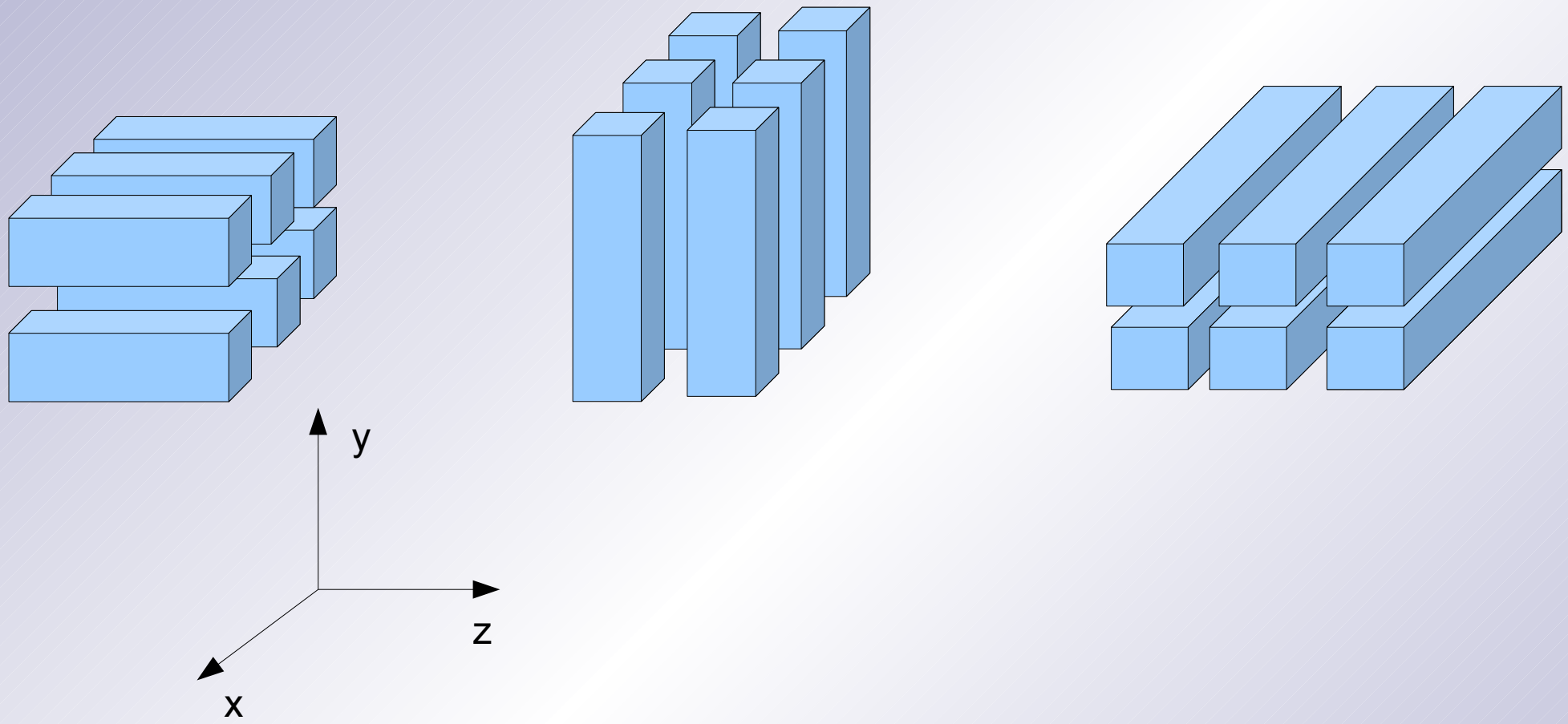


- 1D decomposition limits the number of MPI tasks to `min(`N_y`,` N_z`)`

Code overview

- Most of the time is spent in computing right hand side (rhs) terms
  - Loops updating grid values, boundary conditions
  - Calls to derivatives subroutines
  - Calls to communication subroutines

2D decomposition



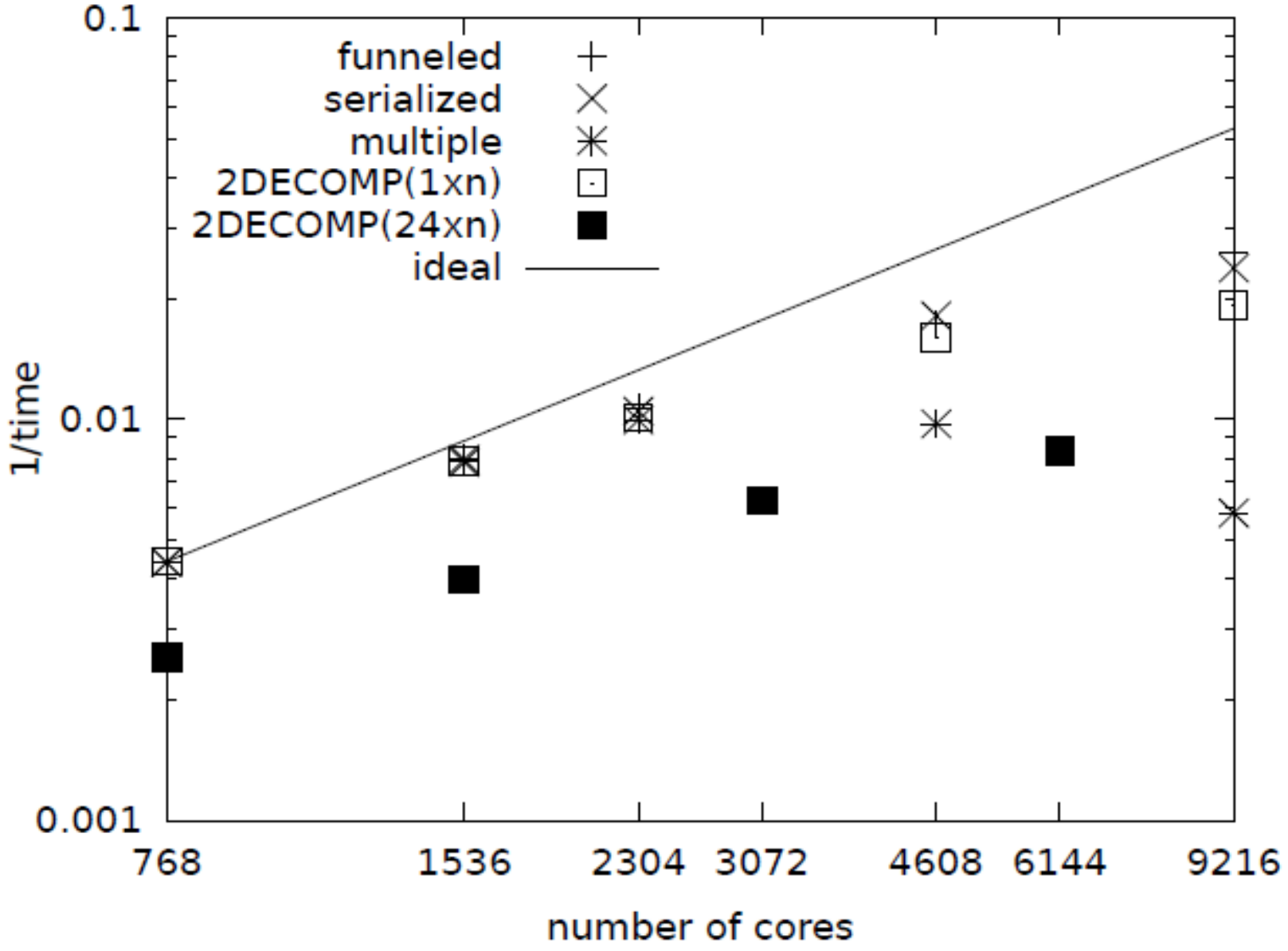
2DECOMP&FFT available at http://www.2decomp.org/

Mixed Mode strategy

- Open parallel region at the top of rhs
- Use DO and WORKSHARE directives for array operations
- Use orphaned directives in called subroutines
- Communication for transpose operation
  - Funneled
  - Serialised
  - Multiple
- Overlap communication with computation

Mixed Mode Scaling (I)

768^3 grid



Mixed Mode Scaling(II)

model	threads				
	1	2	3	6	12
funneled	232	127	98	58	39
serialized	233	128	97	58	42
multiple	232	136	104	106	172
2DECOMP	229	144	103	65	52

Table 1: Average computing time in seconds in RHS subroutine for each mixed mode model described in text.

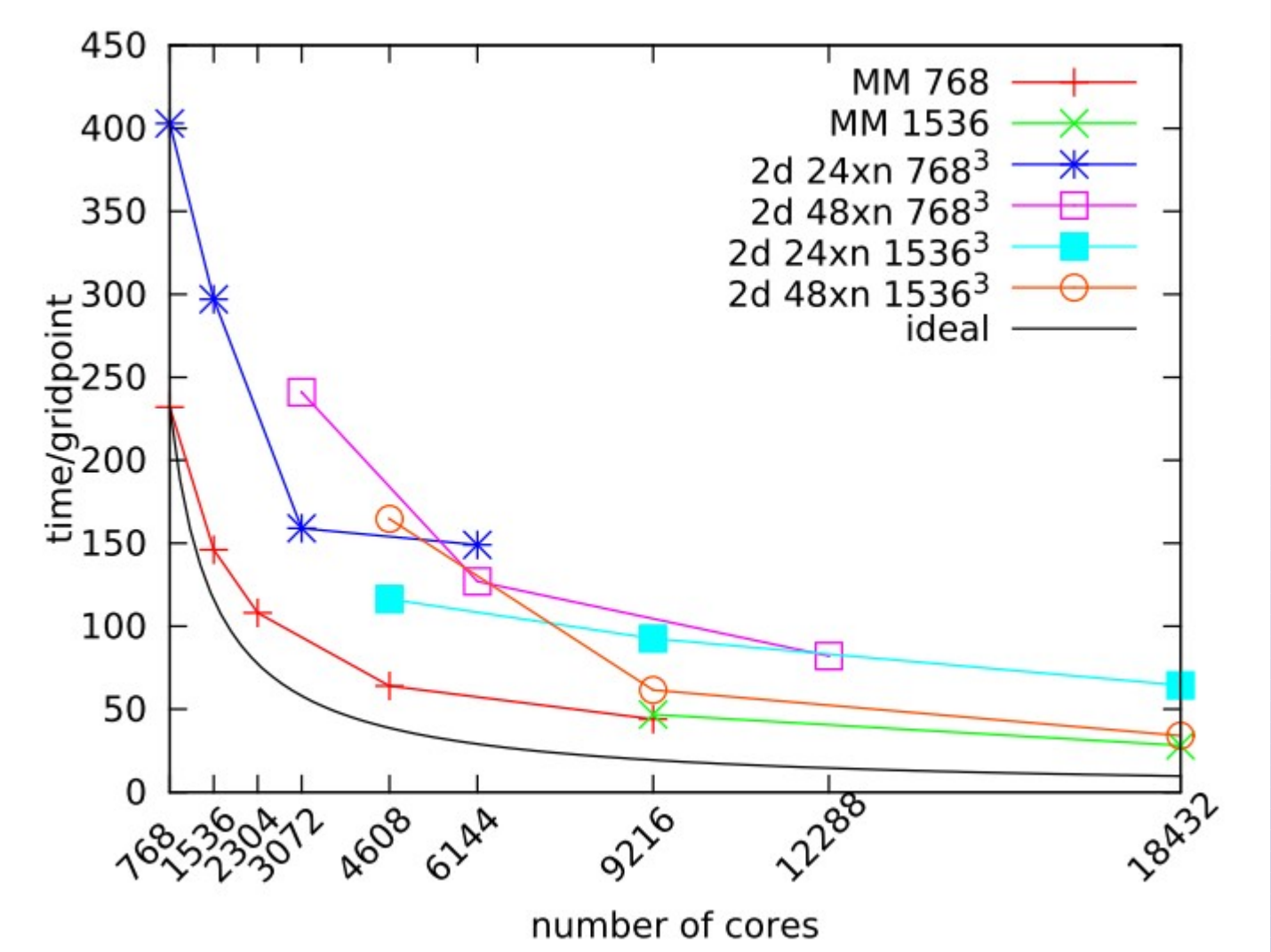
	ranks per NUMA node			
	6	3	2	1
all to all	0.832	0.353	0.366	0.208
send-recv	2.44	0.824	0.480	0.244

Table 2: Maximum time in seconds for a transpose operation from a $768 \times 768 \times 1$ grid to a $768 \times 1 \times 768$ over 768 MPI ranks placed on NUMA nodes as in the case of mixed mode computations with 1, 2, 3 and 6 OpenMP threads. Placement done using `-S` flag of `aprun` command.

- 2DECOMP optimisations:
  - No internal copy for y->z transpose if slab thickness is 1
  - Use OpenMP threads to copy data to internal buffer

MPICH_GNI_MAX_EAGER_MSG_SIZE set to maximum value

2D decomposition cross over



Communication-Computation Overlap with MPI

	XE6				XT4			
threads	t_{tr}	$\bar{t}_{tr}$	t_c	$\bar{t}_c$	t_{tr}	$\bar{t}_{tr}$	t_c	$\bar{t}_c$
1	2.52	16.68	13.15	13.21	2.95	17.50	15.07	15.13
2	1.57	7.81	6.64	6.62	1.51	8.63	7.37	7.39
3	1.28	5.58	4.55	4.54				
4					0.81	4.36	3.66	3.69

Table 2: Average time in seconds for transpose operation that is done as a compact block (t_{tr}) and overlapping a computation block ($\bar{t}_{tr}$) for XE6 and XT4 machines. The respective computation times t_c, $\bar{t}_c$ is approximately the same in both cases, as expected. $\bar{t}_{tr} \approx t_{tr} + t_c$ implies that none or little communication is done during the computation. The bulk of communication is done in the call of `MPI_WAITALL`.

1	1.01	-0.53
2	-0.40	-0.25
3	-0.25	
4		-0.11

$$\bar{t}_{tr} - (t_{tr} + t_c)$$

More on CCO

Listing 2: A simple benchmark to determine the capability of the MPI library to perform asynchronous nonblocking point-to-point communication for large messages (receive variant).

```
1 if(rank==0) {
2   stime = MPI_Wtime();
3   MPI_Irecv(rbuf,mcount,MPI_DOUBLE,1,0,MPI_COMM_WORLD,&req);
4   do_work(calctime);
5   MPI_Wait(req, &status);
6   etime = MPI_Wtime();
7   cout << calctime << " " << etime-stime << endl;
8 } else MPI_Send(sbuf,mcount,MPI_DOUBLE,0,0,MPI_COMM_WORLD);
```

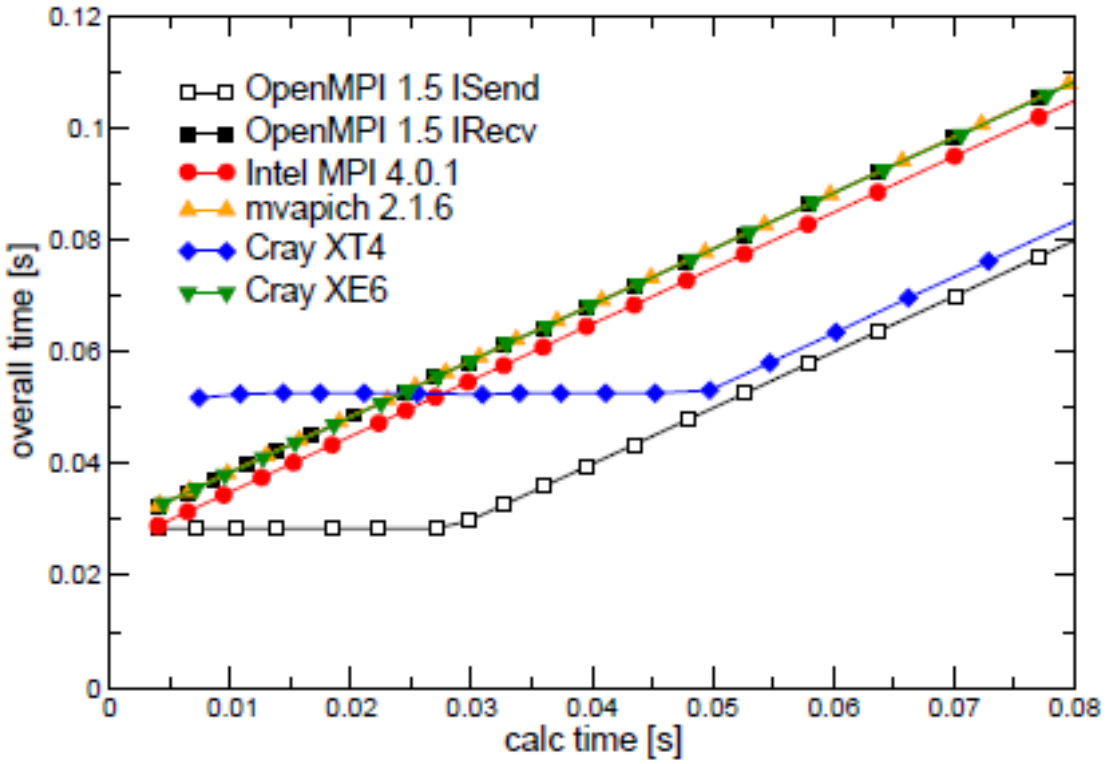


Figure 4: Internode results for the nonblocking MPI benchmark on the Westmere-based test cluster and on Cray XT4 and XE6 systems. Unless indicated otherwise, results for nonblocking send and receive are almost identical.

G. Schubert, H. Fehske, G. Hager, and G. Wellein, Parallel Processing Letters 21(3), 339-358 (2011). Preprint: arXiv:1106.5908

HECToR

RESEARCH COUNCILS UK

CCO with OpenMP

```
 myth=omp_get_num_thread()
 k=mod(nxl(2),nth_max-1)
 isx=(myth-1)*(nxl(2)/(nth_max-
1))+1+min(k,myth-1)
 iex=isx+(nxl(2)/(nth_max-1))-1
 if ( myth <= k )iex=iex+1
 ...
!$OMP BARRIER
If ( myth == 0) then
 Call mpi_transpose(...)
Else
  Do i=isx, iex
! work
    ...
  Enddo
Endif
 ...
```

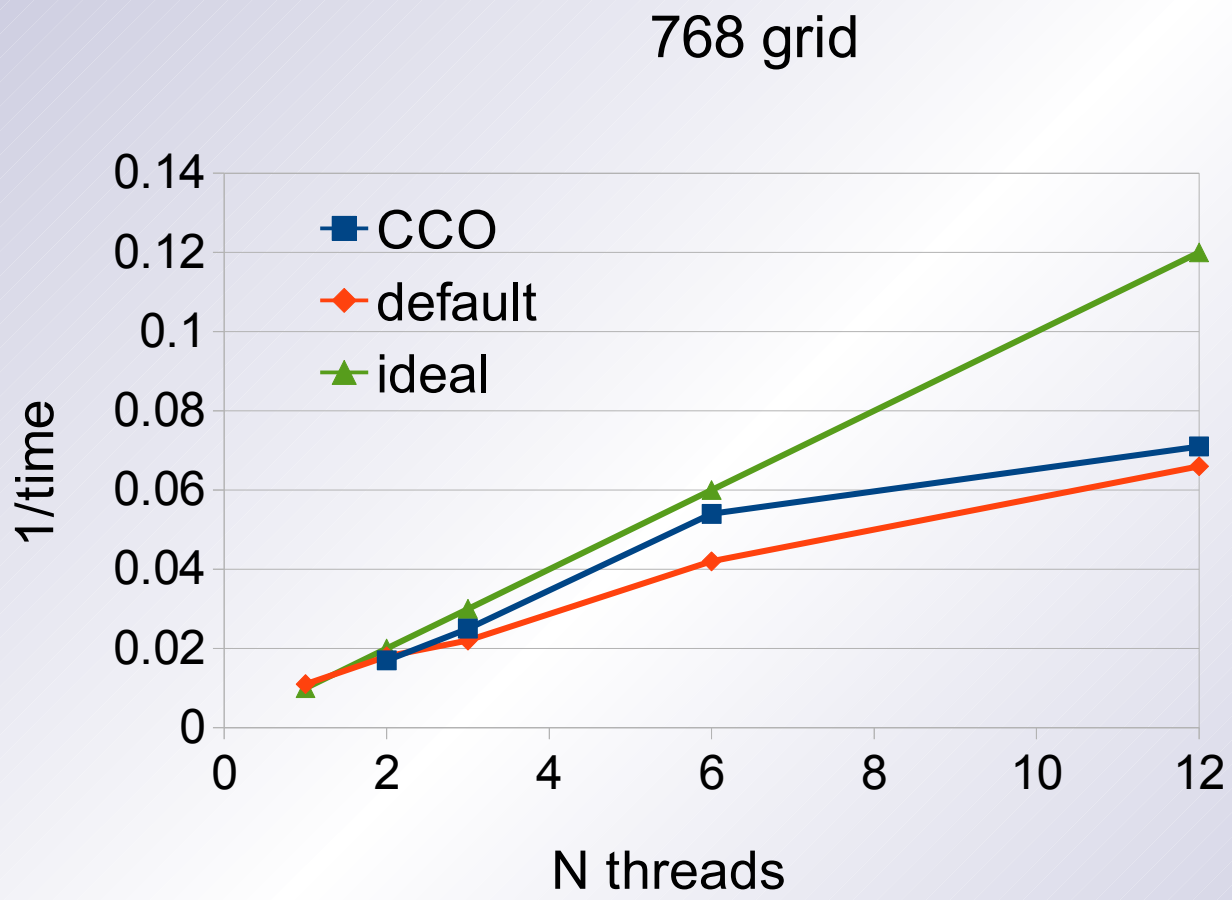
```
...
!$OMP BARRIER
!$OMP MASTER
 Call mpi_transpose(...)
!$OMP END MASTER
!$OMP DO COLLAPSE (2)
SCHEDULE(dynamic,ni*nj/(nth-1))
  Do i=1,ni
   Do j=1,nj
! work
      ...
   Enddo
  Enddo
!$OMP ENDDO NOWAIT
 ...
```

Mixed mode CCO timing

threads	1	2	3	6	12
communication time					
			786^3 grid		
t_{no}	53.80	25.32	24.61	11.08	10.21
t_{wo}		27.18	25.19	11.58	10.83
				1536^3 grid	
t_{no}				73.09	38.10
t_{wo}				71.68	41.88
non-overlapped computation time					
			786^3 grid		
t_{no}	58.97	30.70	21.40	12.62	7.01
t_{wo}		30.13	15.12	6.08	3.33
				1536^3 grid	
t_{no}				61.70	39.22
t_{wo}				34.86	17.88

Table 3: Average time in seconds for transpose operations and computation time for regions that are not covered by the communication for the initial version of the code (t_{no}) and the code that uses CCO with help of OpenMP threads (t_{wo}). Data collected from runs on two cubic grids with linear dimensions 768 and 1536. Please note the scaling of the communication time with the number of threads, the amount of data transferred is independent of number of threads.

CCO scaling


768 grid
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
1/time
CCO
default
ideal
0
2
4
6
8
10
12
N threads

CCO on sectors
6, 12 OpenMP threads

		static		dynamic	
	threads	total	comm	total	comm
pq	6	7.10	7.10		
	12	4.98	4.98		
s1w1	6	1.93	1.93	1.94	1.93
	12	1.30	1.29	1.30	1.30
wei	6	1.54	0.91	2.21	0.89
	12	0.79	0.77	1.36	0.72
reaq	6	0.94	0.70		
	12	0.87	0.86		
s5w7	6	1.49	0.94		
	12	0.84	0.84		
s3w5	6	2.29	2.29		
	12	1.51	1.51		

Conclusions

- Mixed mode provides good scaling (50-60% efficiency).
  - 18,432 cores, 12 threads per node (1536x1536x1536 grid).
- Computation-communication overlap with specialised OpenMP threads could bring 10-15% speed up.
- MPI CCO does not work as yet, but communication is faster for underpopulated nodes.

Acknowledgements

HECToR

a Research Councils UK High End Computing Service.

Engineering and Physical Sciences Research Council Grant No.EP/I000801/1.

LA thanks Kevin Roy for useful discussions.