Computational Efficiency of the Aerosol Scheme in the Met Office Unified Model

Mark Richardson, Ph.D.

University of Leeds, NCAS

Acknowledgements

UNIVERSITY OF LEEDS

- NCAS funding in association with NERC, https://www.ncas.ac.uk/
- JWCRP collaboration with UK Met Office, http://www.jwcrp.org.uk/
- University of Leeds School of Earth and Environment for hosting the researcher http://see.leeds.ac.uk/
- Graham Mann, University of Leeds
- Fiona O’Connor, Earth Systems and Mitigation Science, UK Met. Office
- Paul Selwood, HPC Optimisation, UK Met. Office
- UK Met Office Collaborative Service for access to MONSooN HPC system
    - A Cray XC40 reached through a secured gateway

Background

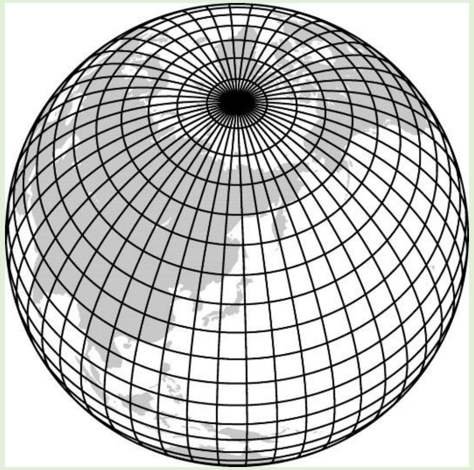
UNIVERSITY OF LEEDS

- What is the UM
  - A computer simulation for weather prediction
  - Primarily short range forecast
  - Increasingly used for climate simulations
  - Some local area simulations such as regional air quality effects
- Resolution
  - All this work is with N96 (192x144x85 grid boxes) ~ 2degree
- MPI configuration
  - 128 MPI tasks in 2D topology (16x8)
- My job is funded by JWCRP (NCAS) in collaboration with Met. Office
  - 33 months to assess computational efficiency and address places in code where it appears inefficient
  - Introduce OpenMP to UKCA for enhanced parallelism

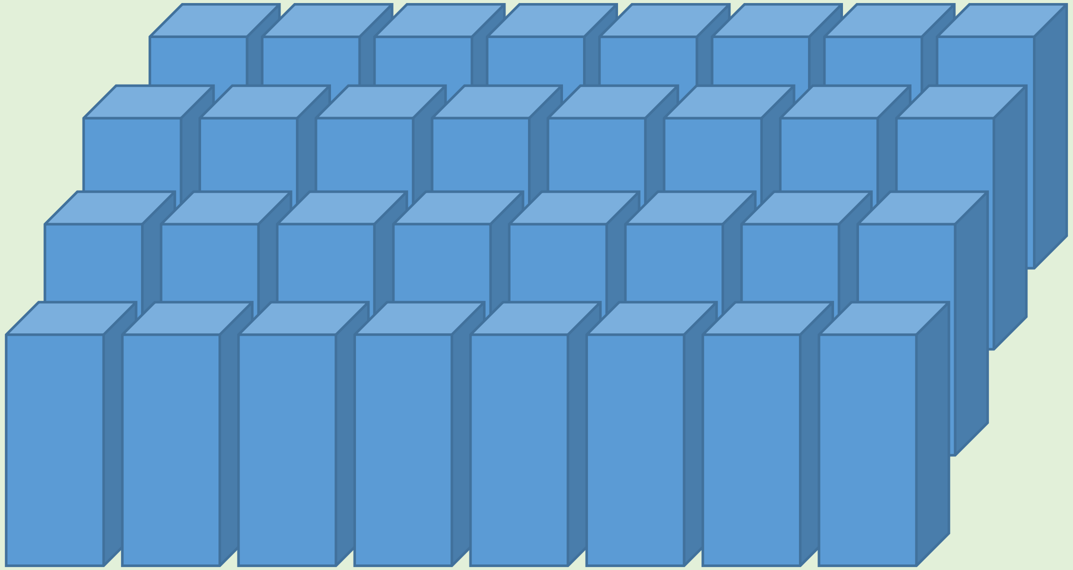
Motivation

- When activated UKCA adds overhead
  - Mode 1, full chemistry and aerosol
    - 30% of run is >40% overhead
  - Mode 2: reduced chemistry
    - with pre-calculated concentrations, i.e. offline oxidants
    - 20% of run is ~25% overhead
- Climate simulations
  - Years 1850-1950 and 1950-2050 (e.g pre-industrial, post-industrial)
  - At best 15 months per day
  - Resolution is N96L85 (192x144x85 grid boxes)
  - 448 cpus
- Options
  - Reduced complexity e.g. reduced GLOMAP and fewer chemical species more parameterisation (some groups are doing this)
  - Limit the frequency of calculation, typical only call UKCA every third time step
  - Improve computational efficiency

Relating physical space, computational domain and arrangement in memory

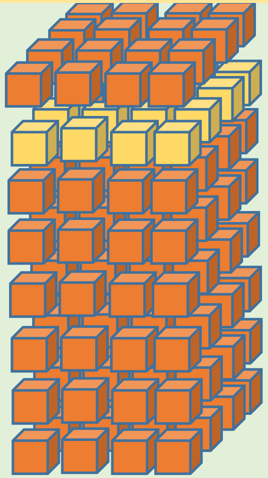


8x4 MPI tasks

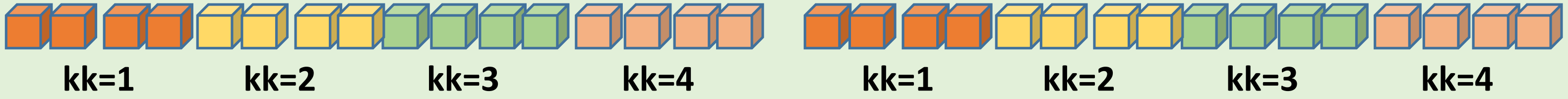


One MPI task

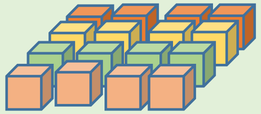
L = 1, model_levels



Two layers in memory



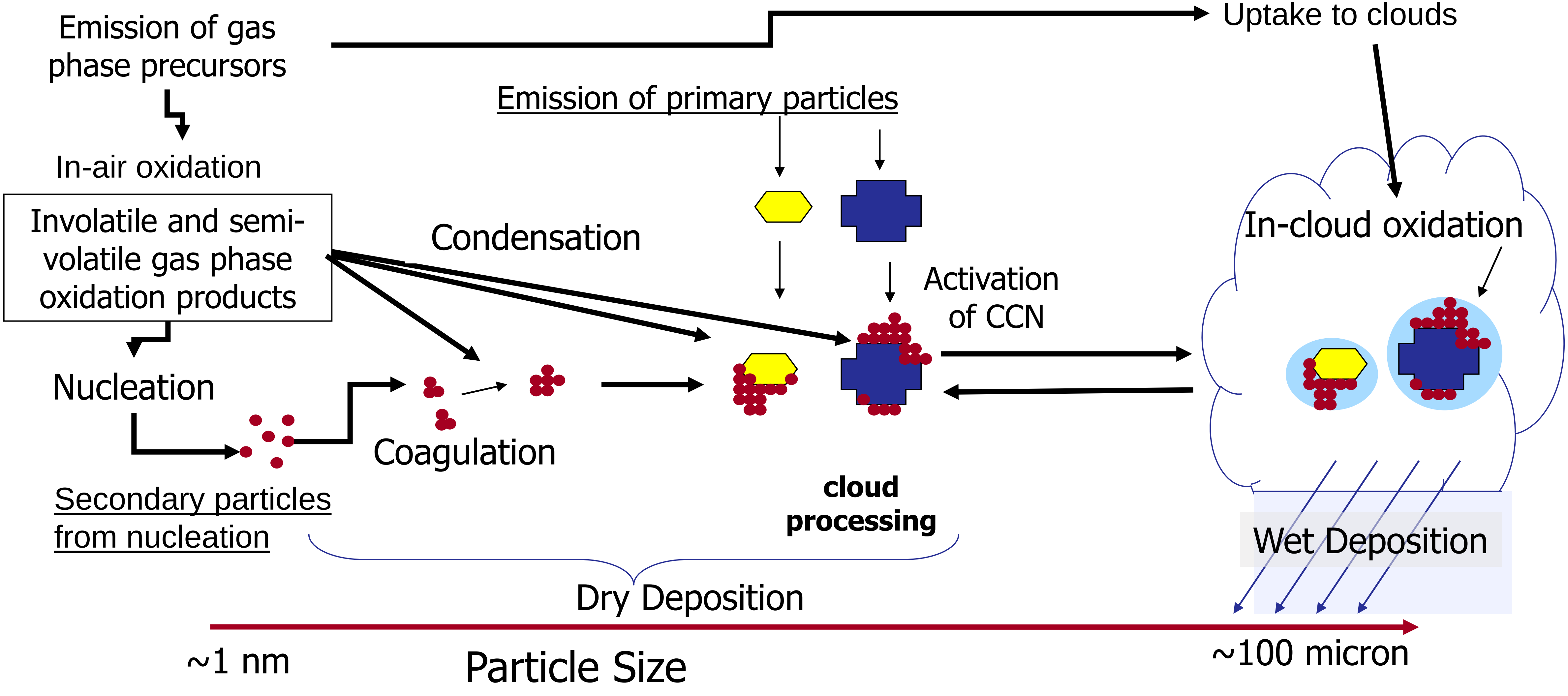
kk=1, row_length



Globe Graphic: Warren M Washington et al. Phil. Trans. R. Soc. A 2009;367:833-846

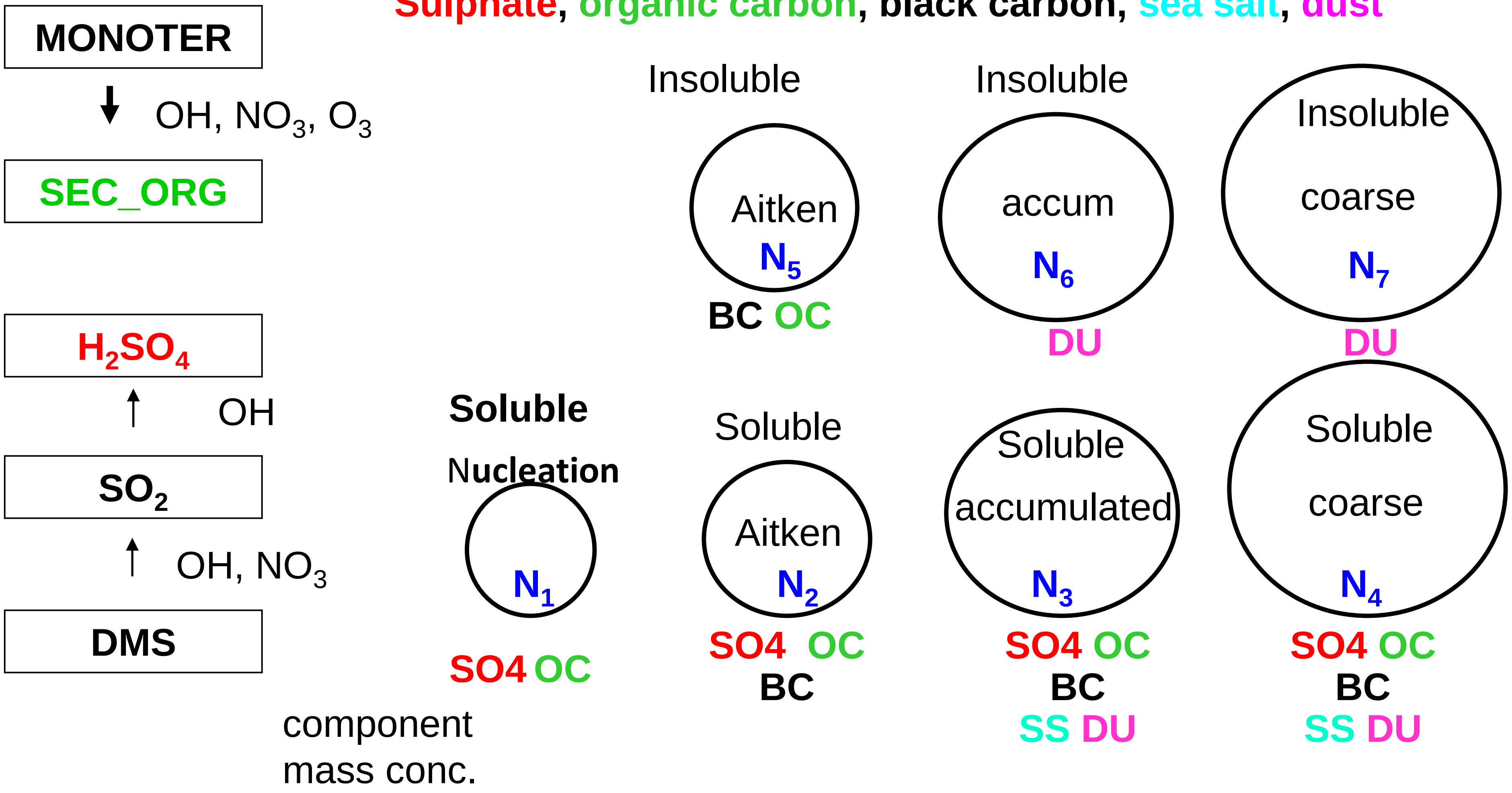
Aerosol processes, size and composition

UNIVERSITY OF LEEDS



GLOMAP-mode standard configuration (with dust)

Sulphate, organic carbon, black carbon, sea salt, dust



MONOTER

OH, NO_3, O_3

SEC_ORG

H_2SO_4

OH

SO_2

OH, NO_3

DMS

Insoluble Aitken N_5 — **BC OC**

Insoluble accum N_6 — **DU**

Insoluble coarse N_7 — **DU**

Soluble Nucleation N_1 — **SO4 OC**

Soluble Aitken N_2 — **SO4 OC BC**

Soluble accumulated N_3 — **SO4 OC BC SS DU**

Soluble coarse N_4 — **SO4 OC BC SS DU**

component mass conc.

Aerosol mass as “components” in internally mixed modes

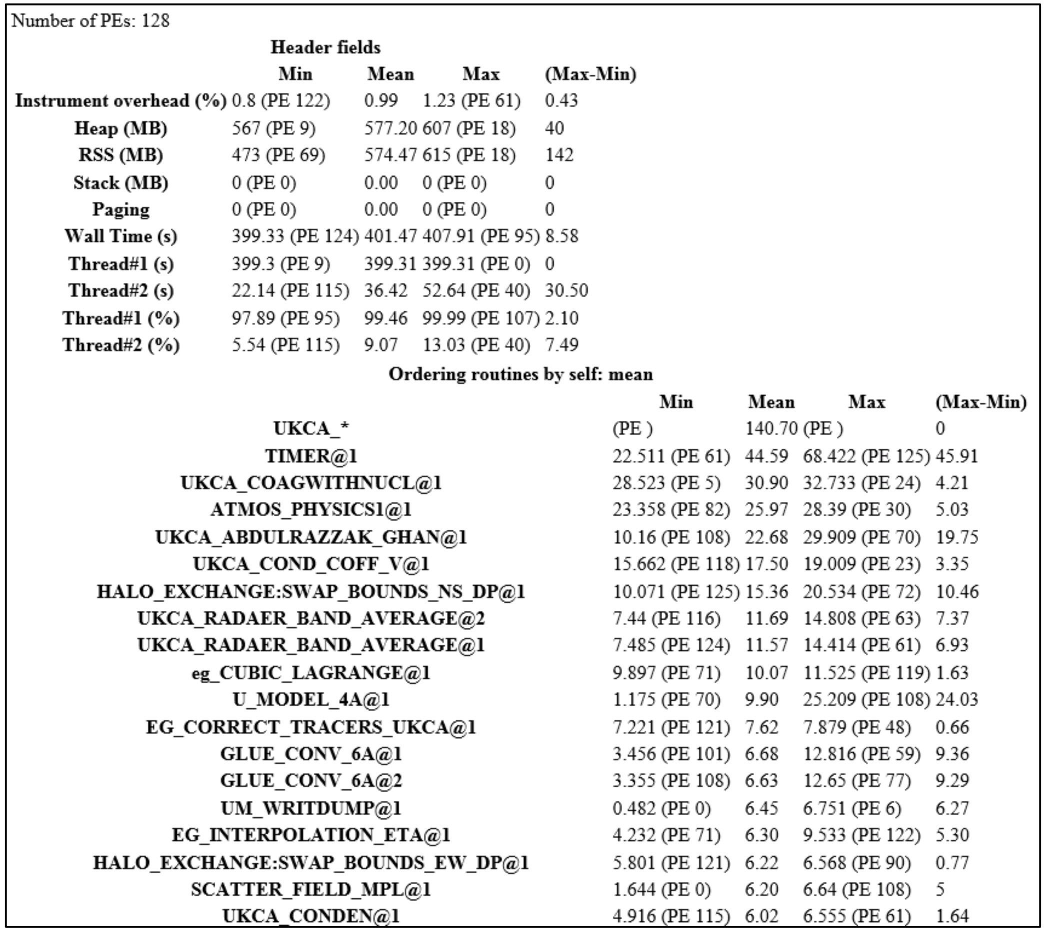
19 mass

7 number

Transported tracers=26

DrHook reports

Reference code 2 threads



Number of PEs: 128

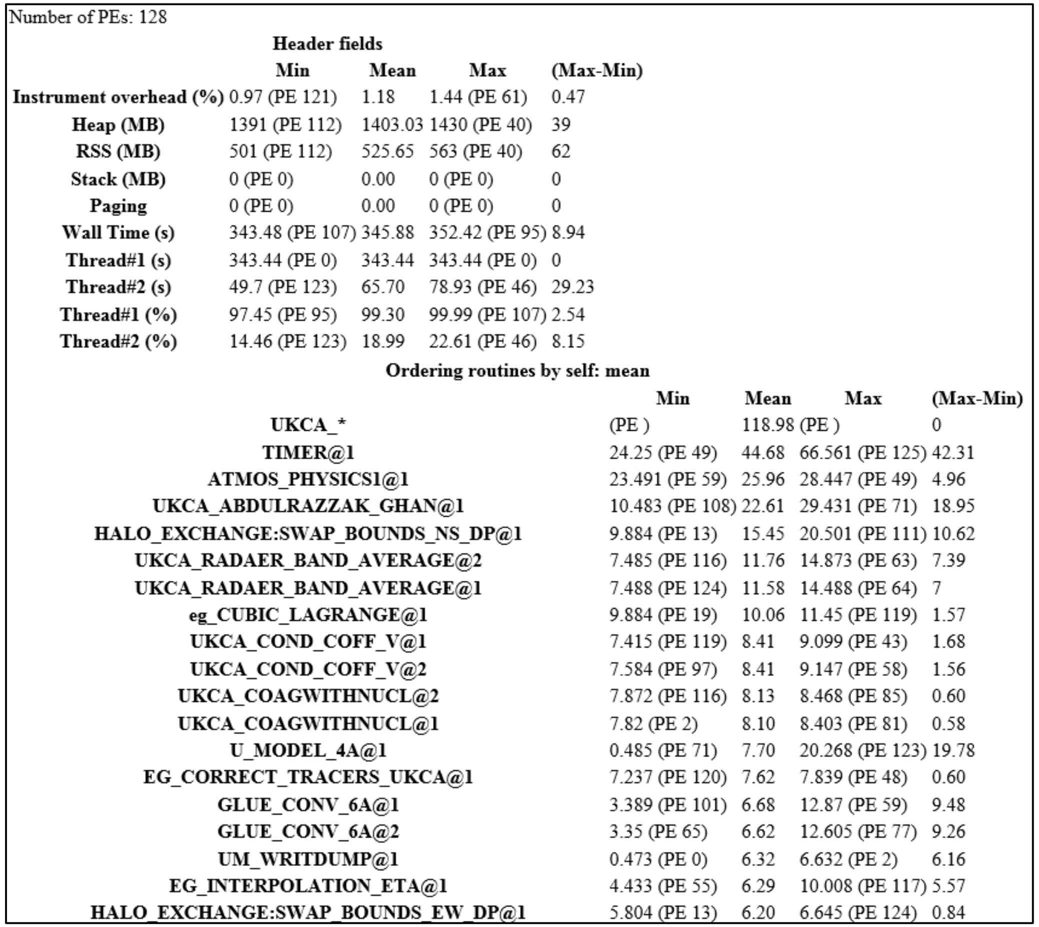
Header fields

	Min	**Mean**	**Max**	**(Max-Min)**
Instrument overhead (%)	0.8 (PE 122)	0.99	1.23 (PE 61)	0.43
Heap (MB)	567 (PE 9)	577.20	607 (PE 18)	40
RSS (MB)	473 (PE 69)	574.47	615 (PE 18)	142
Stack (MB)	0 (PE 0)	0.00	0 (PE 0)	0
Paging	0 (PE 0)	0.00	0 (PE 0)	0
Wall Time (s)	399.33 (PE 124)	401.47	407.91 (PE 95)	8.58
Thread#1 (s)	399.3 (PE 9)	399.31	399.31 (PE 0)	0
Thread#2 (s)	22.14 (PE 115)	36.42	52.64 (PE 40)	30.50
Thread#1 (%)	97.89 (PE 95)	99.46	99.99 (PE 107)	2.10
Thread#2 (%)	5.54 (PE 115)	9.07	13.03 (PE 40)	7.49

Ordering routines by self: mean

	Min	**Mean**	**Max**	**(Max-Min)**
UKCA_*	(PE)	140.70	(PE)	0
TIMER@1	22.511 (PE 61)	44.59	68.422 (PE 125)	45.91
UKCA_COAGWITHNUCL@1	28.523 (PE 5)	30.90	32.733 (PE 24)	4.21
ATMOS_PHYSICS1@1	23.358 (PE 82)	25.97	28.39 (PE 30)	5.03
UKCA_ABDULRAZZAK_GHAN@1	10.16 (PE 108)	22.68	29.909 (PE 70)	19.75
UKCA_COND_COFF_V@1	15.662 (PE 118)	17.50	19.009 (PE 23)	3.35
HALO_EXCHANGE:SWAP_BOUNDS_NS_DP@1	10.071 (PE 125)	15.36	20.534 (PE 72)	10.46
UKCA_RADAER_BAND_AVERAGE@2	7.44 (PE 116)	11.69	14.808 (PE 63)	7.37
UKCA_RADAER_BAND_AVERAGE@1	7.485 (PE 124)	11.57	14.414 (PE 61)	6.93
eg_CUBIC_LAGRANGE@1	9.897 (PE 71)	10.07	11.525 (PE 119)	1.63
U_MODEL_4A@1	1.175 (PE 70)	9.90	25.209 (PE 108)	24.03
EG_CORRECT_TRACERS_UKCA@1	7.221 (PE 121)	7.62	7.879 (PE 48)	0.66
GLUE_CONV_6A@1	3.456 (PE 101)	6.68	12.816 (PE 59)	9.36
GLUE_CONV_6A@2	3.355 (PE 108)	6.63	12.65 (PE 77)	9.29
UM_WRITDUMP@1	0.482 (PE 0)	6.45	6.751 (PE 6)	6.27
EG_INTERPOLATION_ETA@1	4.232 (PE 71)	6.30	9.533 (PE 122)	5.30
HALO_EXCHANGE:SWAP_BOUNDS_EW_DP@1	5.801 (PE 121)	6.22	6.568 (PE 90)	0.77
SCATTER_FIELD_MPL@1	1.644 (PE 0)	6.20	6.64 (PE 108)	5
UKCA_CONDEN@1	4.916 (PE 115)	6.02	6.555 (PE 61)	1.64

Development code 2 threads



Number of PEs: 128

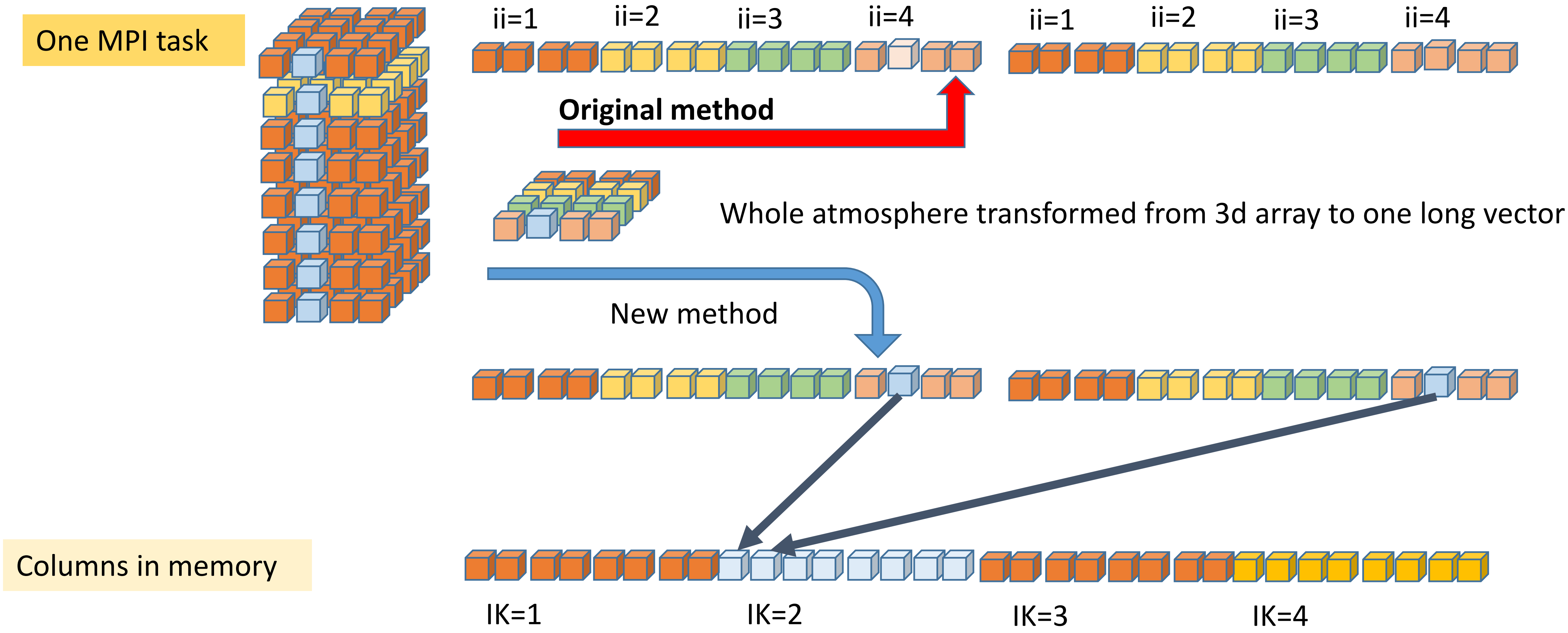
Header fields

	Min	**Mean**	**Max**	**(Max-Min)**
Instrument overhead (%)	0.97 (PE 121)	1.18	1.44 (PE 61)	0.47
Heap (MB)	1391 (PE 112)	1403.03	1430 (PE 40)	39
RSS (MB)	501 (PE 112)	525.65	563 (PE 40)	62
Stack (MB)	0 (PE 0)	0.00	0 (PE 0)	0
Paging	0 (PE 0)	0.00	0 (PE 0)	0
Wall Time (s)	343.48 (PE 107)	345.88	352.42 (PE 95)	8.94
Thread#1 (s)	343.44 (PE 0)	343.44	343.44 (PE 0)	0
Thread#2 (s)	49.7 (PE 123)	65.70	78.93 (PE 46)	29.23
Thread#1 (%)	97.45 (PE 95)	99.30	99.99 (PE 107)	2.54
Thread#2 (%)	14.46 (PE 123)	18.99	22.61 (PE 46)	8.15

Ordering routines by self: mean

	Min	**Mean**	**Max**	**(Max-Min)**
UKCA_*	(PE)	118.98	(PE)	0
TIMER@1	24.25 (PE 49)	44.68	66.561 (PE 125)	42.31
ATMOS_PHYSICS1@1	23.491 (PE 59)	25.96	28.447 (PE 49)	4.96
UKCA_ABDULRAZZAK_GHAN@1	10.483 (PE 108)	22.61	29.431 (PE 71)	18.95
HALO_EXCHANGE:SWAP_BOUNDS_NS_DP@1	9.884 (PE 13)	15.45	20.501 (PE 111)	10.62
UKCA_RADAER_BAND_AVERAGE@2	7.485 (PE 116)	11.76	14.873 (PE 63)	7.39
UKCA_RADAER_BAND_AVERAGE@1	7.488 (PE 124)	11.58	14.488 (PE 64)	7
eg_CUBIC_LAGRANGE@1	9.884 (PE 19)	10.06	11.45 (PE 119)	1.57
UKCA_COND_COFF_V@1	7.415 (PE 119)	8.41	9.099 (PE 43)	1.68
UKCA_COND_COFF_V@2	7.584 (PE 97)	8.41	9.147 (PE 58)	1.56
UKCA_COAGWITHNUCL@2	7.872 (PE 116)	8.13	8.468 (PE 85)	0.60
UKCA_COAGWITHNUCL@1	7.82 (PE 2)	8.10	8.403 (PE 81)	0.58
U_MODEL_4A@1	0.485 (PE 71)	7.70	20.268 (PE 123)	19.78
EG_CORRECT_TRACERS_UKCA@1	7.237 (PE 120)	7.62	7.839 (PE 48)	0.60
GLUE_CONV_6A@1	3.389 (PE 101)	6.68	12.87 (PE 59)	9.48
GLUE_CONV_6A@2	3.35 (PE 65)	6.62	12.605 (PE 77)	9.26
UM_WRITDUMP@1	0.473 (PE 0)	6.32	6.632 (PE 2)	6.16
EG_INTERPOLATION_ETA@1	4.433 (PE 55)	6.29	10.008 (PE 117)	5.57
HALO_EXCHANGE:SWAP_BOUNDS_EW_DP@1	5.804 (PE 13)	6.20	6.645 (PE 124)	0.84

Comparison of memory layout for segment method



One MPI task

Original method

Whole atmosphere transformed from 3d array to one long vector

New method

Columns in memory

Code restructure for columns

UNIVERSITY OF LEEDS

Original

```
nbox = ni*nk*nl
t1d = reshape(t3d, nbox)
…
call aero_step(t1d,nbox)
…
mode_tracers = reshape(ae_nd, nk,ni,nl)
…
! mode_tracers returned to atmosphere code
```

Modified

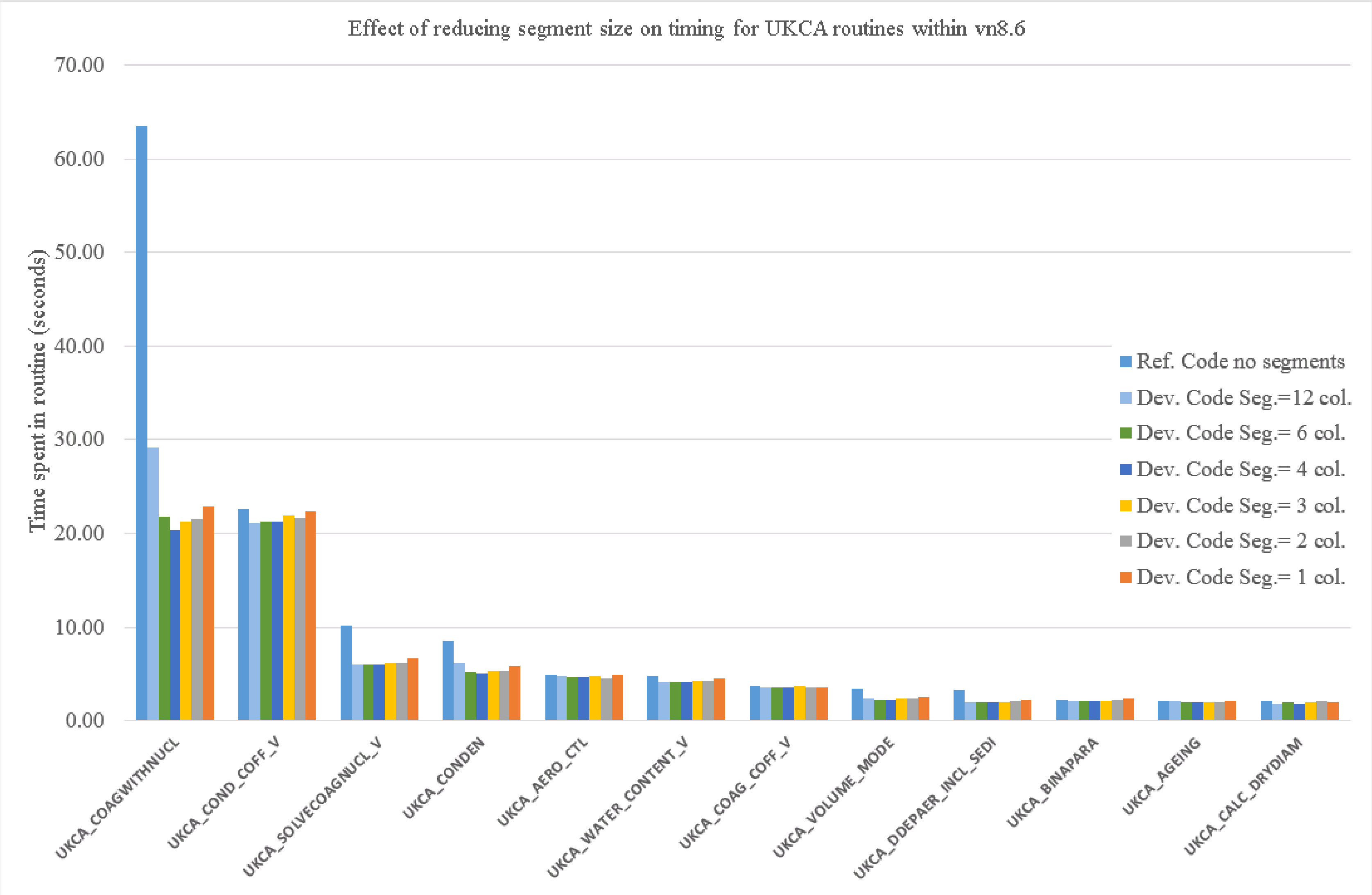
```
DO ii = 1,  rows
  DO ik = 1, num_segs
     ! Extract columns for this seg
     jl =0
     DO kk = k_lo, k_up
        DO L = 1, model_levels
          jl = jl +1
          t1d(jl) = t3d(kk,ii,l)
        END DO
     END DO
     nbox_seg = jl
     call aero_step(t1d, nbox_seg)
     jl =0
     DO kk = k_lo, k_up
       DO L = 1, model_levels
         jl = jl +1
         mode_tracers(kk,ii,l) = ae_nd(jl)
       END DO
     END DO
  END DO
END DO
! Mode_tracers returned to atmosphere code
```

Replace intrinsic RESHAPE with double nested loops sized to the segment
Additional loops for rows and segments per row

Benefit from compiler "in-line" optimisations
Preparation for OpenMP region

Effect of reducing the segment size, no OpenMP

- **Each group of columns relate to a single function within UKCA**
- **First column in group is reference code**
- **Top 12 shown**
- **Significant effect seen on highest workload function**
- **Not all as dramatic but generally all benefit**
- **Appears that choosing 4 columns per segment is optimal**



Add Open MP parallelism

Original

```
nbox = ni*nk*nl

t1d=reshape(t3d, nbox)

call aero_step(t1d,nbox)

mode_tracers=reshape(ae_nd, nk,ni,

! mode_tracers returned to atmosph
```

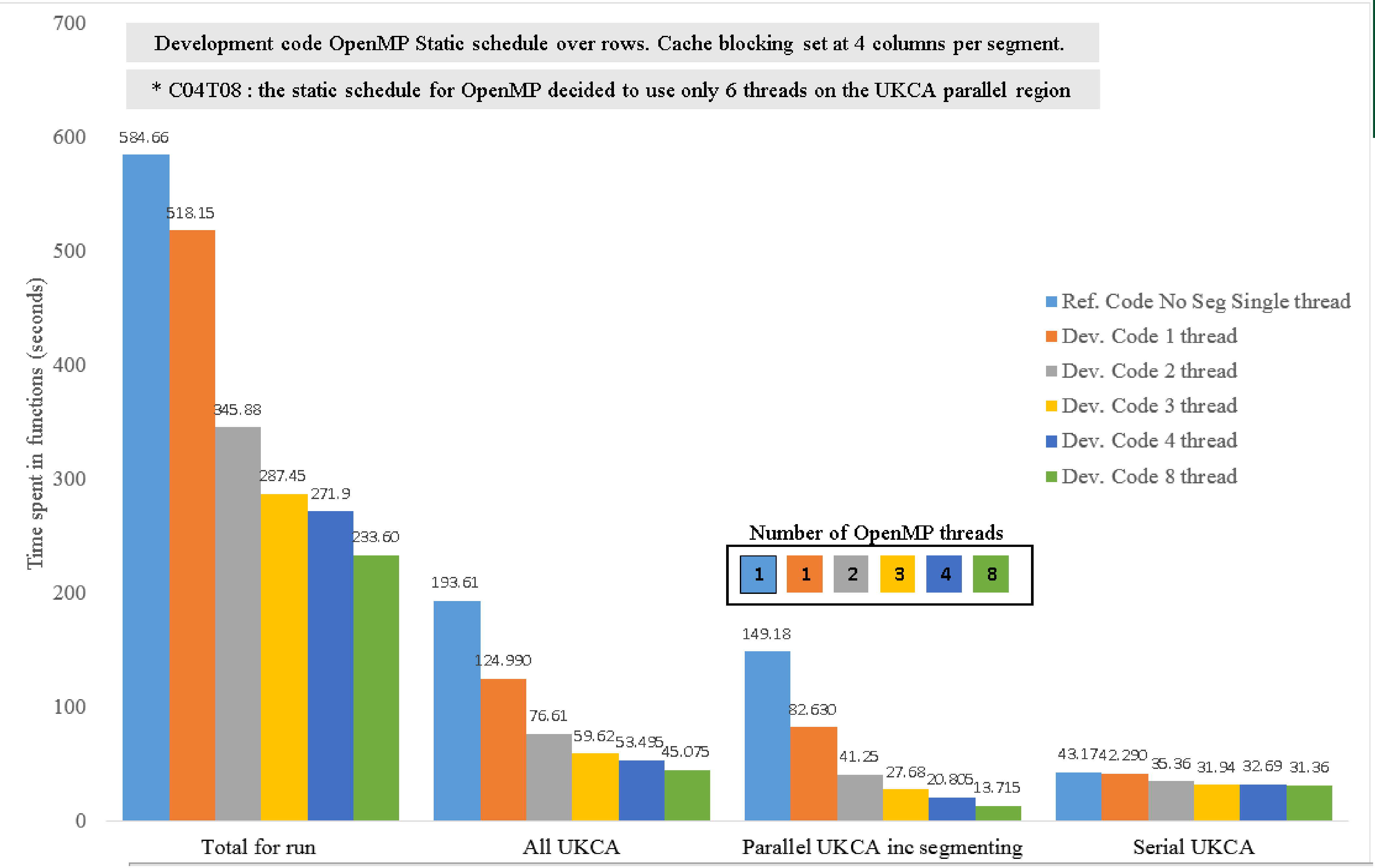
Significant effort in code rewriting to make the gains described here.

Modified for cache blocking

```
DO ii = 1,  rows
  DO ik = 1, num_segs
     ! Extract columns for this seg
     jl =0
     DO kk = k_lo, k_up
        DO L = 1, model_levels
          jl = jl +1
          t1d(jl) = t3d(kk,ii,l)
        END DO
     END DO
     nbox_seg = jl
     call aero_step(t1d, nbox_seg)
     jl =0
     DO kk = k_lo, k_up
       DO L = 1, model_levels
         jl = jl +1
         mode_tracers(kk,ii,l) = ae_nd(
       END DO
     END DO
  END DO
END DO
! Mode_tracers returned to atmosphere c
```

Modified for OpenMP

```
!$OMP PARALLEL
<some work>
!$OMP DO COLLAPSE(2)
DO ii = 1,  rows
  DO ik = 1, num_segs
    ! Extract columns for this segment
    jl =0
    Do kk = k_lo, k_up
      DO L = 1, model_levels
        jl = jl +1
        t1d(jl) = t3d(k,ii,l)
      END DO
    END DO
    nbox_seg = jl
    call aero_step(t1d, nbox_seg)
    ...
  END DO
END DO
!$OMP END DO
<some work round up>
!$OMP END PARALLEL
! Mode_tracers returned to atmosphere
```



Development code OpenMP Static schedule over rows. Cache blocking set at 4 columns per segment.

* C04T08 : the static schedule for OpenMP decided to use only 6 threads on the UKCA parallel region

Improvement for UKCA compared to whole simulation

- The percentage of the runtime that is spent in UKCA has been reduced
- Reducing perceived “overhead”

	Reference	Development				
	1 thread	1 thread	2 thread	3 thread	4 thread	8 thread
Time for whole simulation	584.7	518.2	345.9	287.5	271.9	233.6
Time spent in UKCA	193.6	125.0	76.6	59.6	53.5	45.1
Percentage of run spent in UKCA	33.1	24.1	22.1	20.7	19.7	19.3

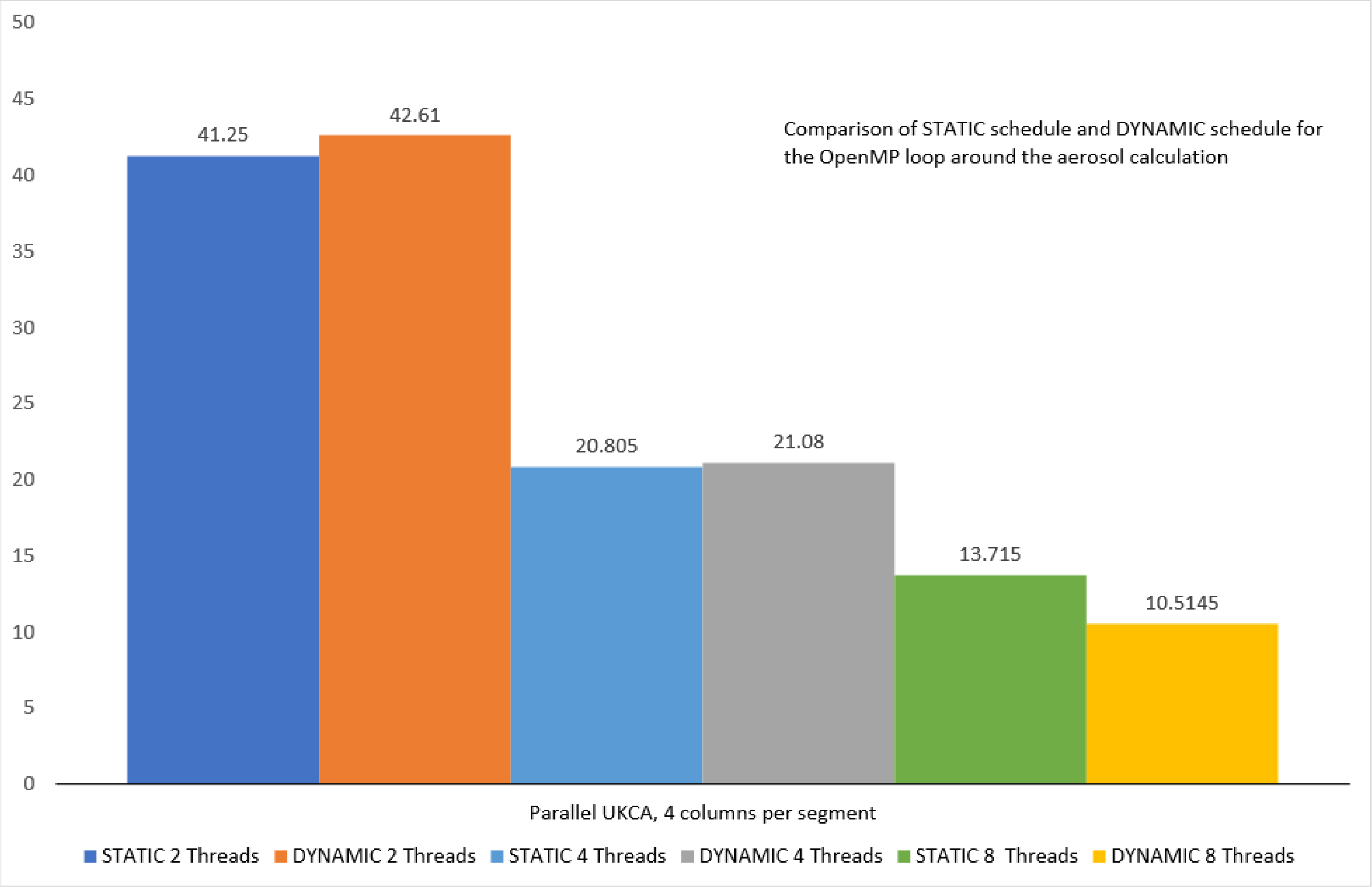
Comparing OpenMP Schedules: Static and Dynamic

- Limits of this implementation
  - OpenMP loop is over the rows of computational cells
  - Loop over segments occurs within the rows loop
    - “COLLAPSE” clause not working yet (need all UM to be upgraded)
- Matching threads to number of rows in sub-domain
  - At high thread count the number of threads should be a factor of the rows
- STATIC distributes rows evenly
  - Noticed that even when 8 threads assigned the scheduler chose only 6
  - Additional difficulty in prescribing 6 threads specifically without intervention
- DYNAMIC will allocate all threads some of the work (iterations)
  - Remaining iterations are allocating as threads become available

Comparison of STATIC and DYNAMIC schedules for region of OpenMP containing UKCA

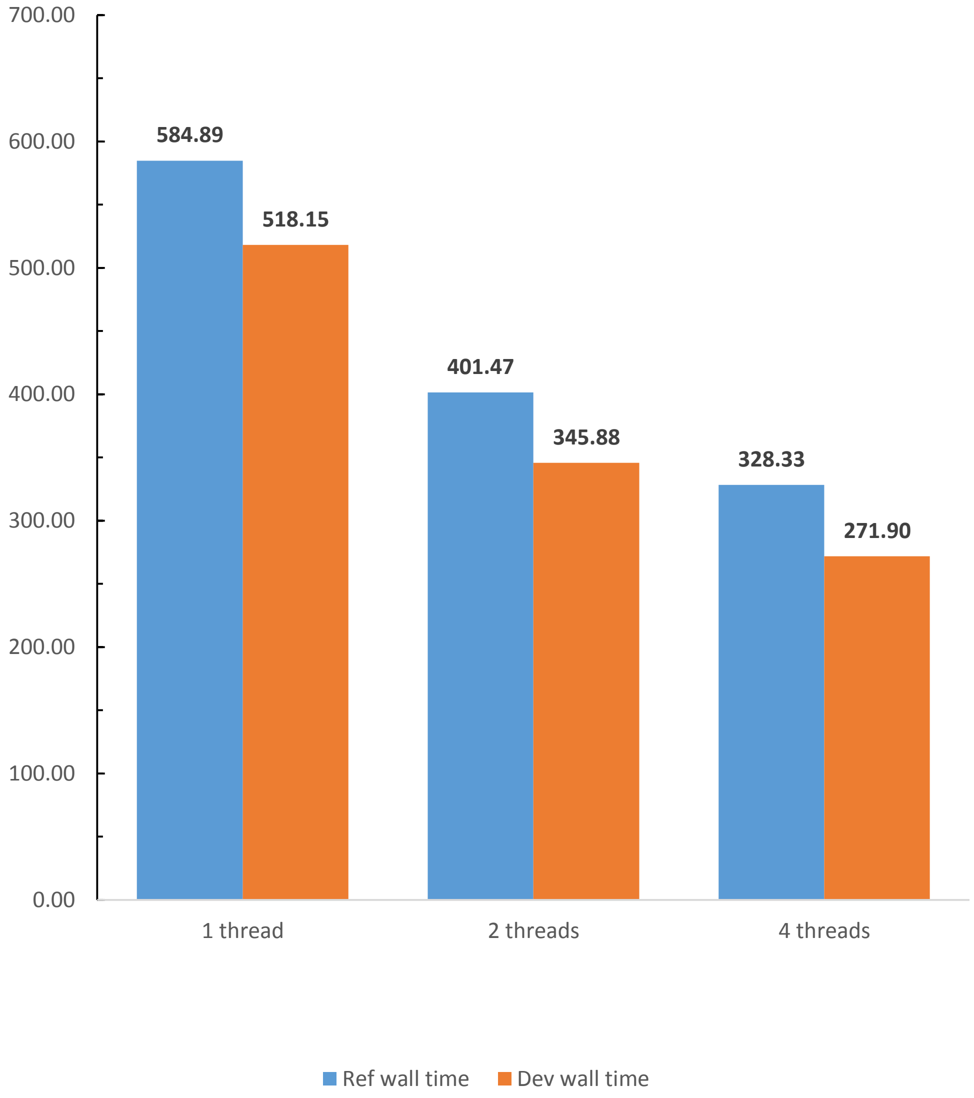
STATIC 8 Thread case is only using 6 Threads – hence 1/3 time

DYNAMIC 8 Threads uses all threads – recovers to 1/4 time



Overall improvement due to segmented data and OpenMP

- The improvement for the single-threaded run is due to the cache-blocking, sending segments of atmosphere to the aerosol processes
- Increasing the number of threads reduces the wall time for both the Reference and Development versions as the UM has regions of OpenMP separate to UKCA
- All this work done with 128 MPI task configuration and fully populated nodes
- The gain in performance due to this work is seen to be
  - 12% through cache blocking seen in the first column pair
  - 14% with cache blocking and 2 OpenMP threads
  - 17% with cache blocking and 4 OpenMP threads
  - Ref. Scaling: 2T = 1.46; 4T = 1.78
  - Dev. Scaling: 2T=1.50; 4T=1.91



status

- Currently month 16 status
  - Early development work within vn8.6 completed
  - Restructured code to send smaller amount of data to GLOMAP
  - Improve highest workload function by 69%
  - Reduce the time spent in GLOMAP from 22% to 14% (30% improvement)
  - Overall runtime reduction by 10%
- Ongoing plan
  - Add the segmentation method to vn10.5 ; ready for UKESM
  - Activate OpenMP around GLOMAP within a vn10.4 branch
  - Start work on a case that has active chemistry

Summary

UNIVERSITY OF LEEDS

- Implementing a segmented atmosphere acts like cache-blocking
  - UKCA components account for 30% of the **reference** single-threaded run
  - This development reduces that to 24% of the **development** run time
  - Corresponding to an overall improvement of 12% in the run-time
- Subsequent implementation of OpenMP provides improved speed of execution
  - Higher thread counts for **reference** code reveal UKCA as a higher fraction of the run-time
  - For 4 threads it is 40%, the OpenMP enabled UKCA **development** has reduced this to 20%
- Clear that some UKCA is still serial
  - This will be addressed in the near future
- This case is aerosol only with limited chemistry
  - The methodology will be applied to chemistry processes as well.