Scaling hybrid coarray/MPI miniapps on Archer

L. Cebamanos[1], A. Shterenlikht[2], J.D. Arregui-Mena[3], L. Margetts[3]

[1]Edinburgh Parallel Computing Centre (EPCC)
The University of Edinburgh, King's Buildings, Edinburgh EH9 3FD, UK
Email: l.cebamanos@epcc.ed.ac.uk

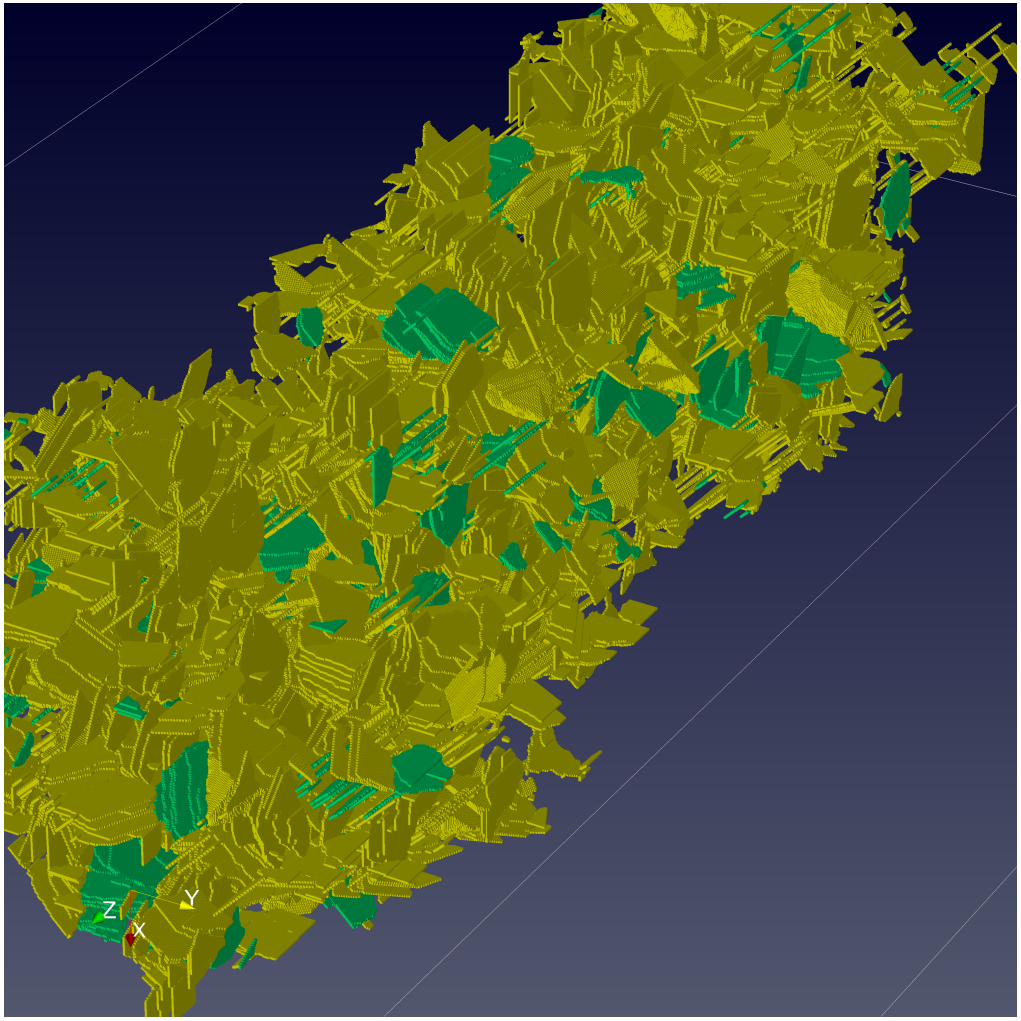
[2]Department of Mechanical Engineering
The University of Bristol, Bristol BS8 1TR, UK, Email: mexas@bris.ac.uk

[3]School of Mechanical, Aero and Civil Engineering
The University of Manchester, Manchester M13 9PL, UK
Emails: jose.arregui-mena@manchester.ac.uk, Lee.Margetts@manchester .ac.uk

CUG2016, London 8-12-MAY-2016

CGPACK - cellular automata microstructure simulation library: `https://sourceforge.net/projects/cgpack`

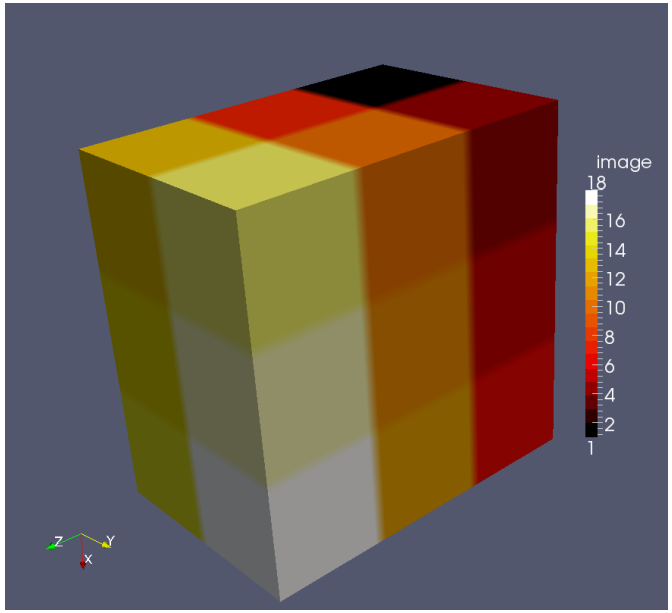
- Solidification, recrystallisation, and fracture of polycrystalline microstructures.
- Fortran 2008 coarrays + TS 18508 [1] extensions.
- HECToR, ARCHER, Intel, OpenCoarrays/GCC systems.
- BSD license

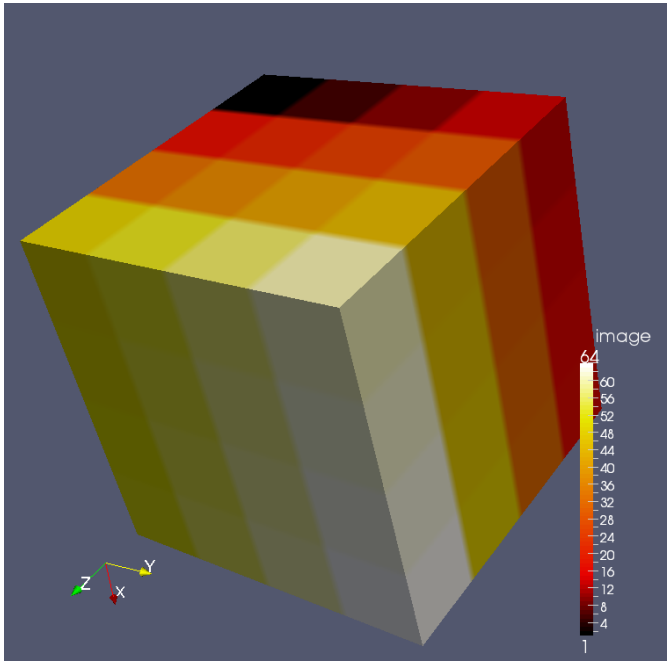


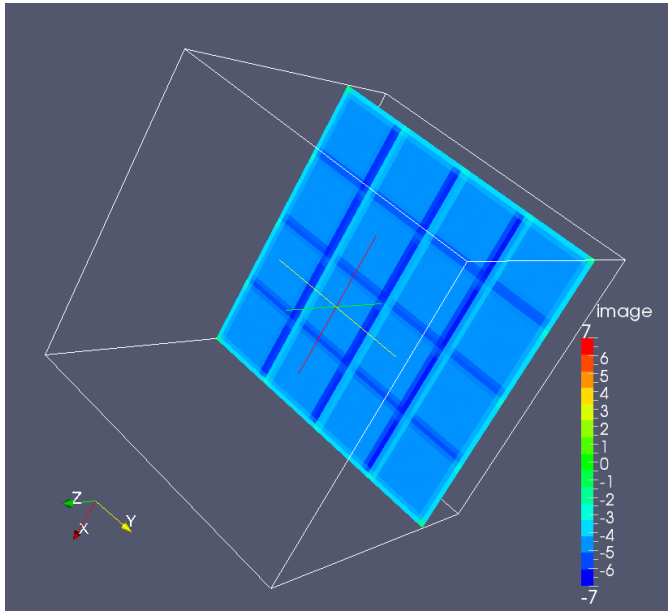
{100} and {110} micro-cracks in individual crystals merge into a macro-crack.

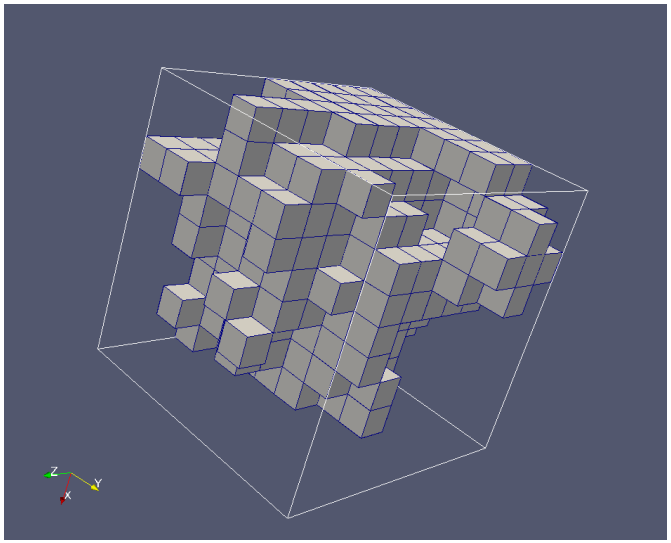
CGPACK design

- CA space coarray - 4D array, 3D corank - structured grid [2, 3, 4].
- Integer cell states
- Fixed or self-similar boundaries
- Traditional halo exchange





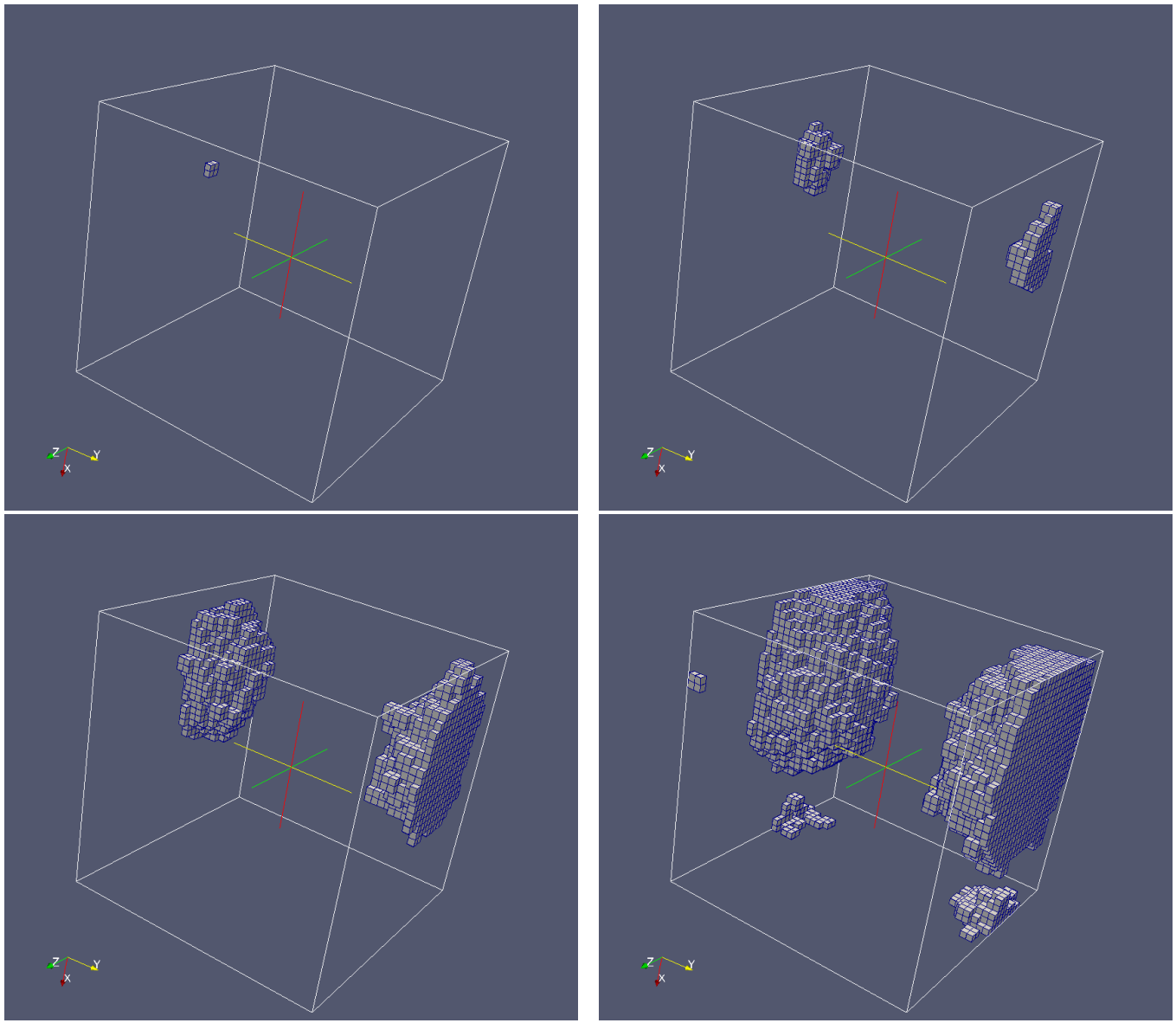




CGPACK space coarray:

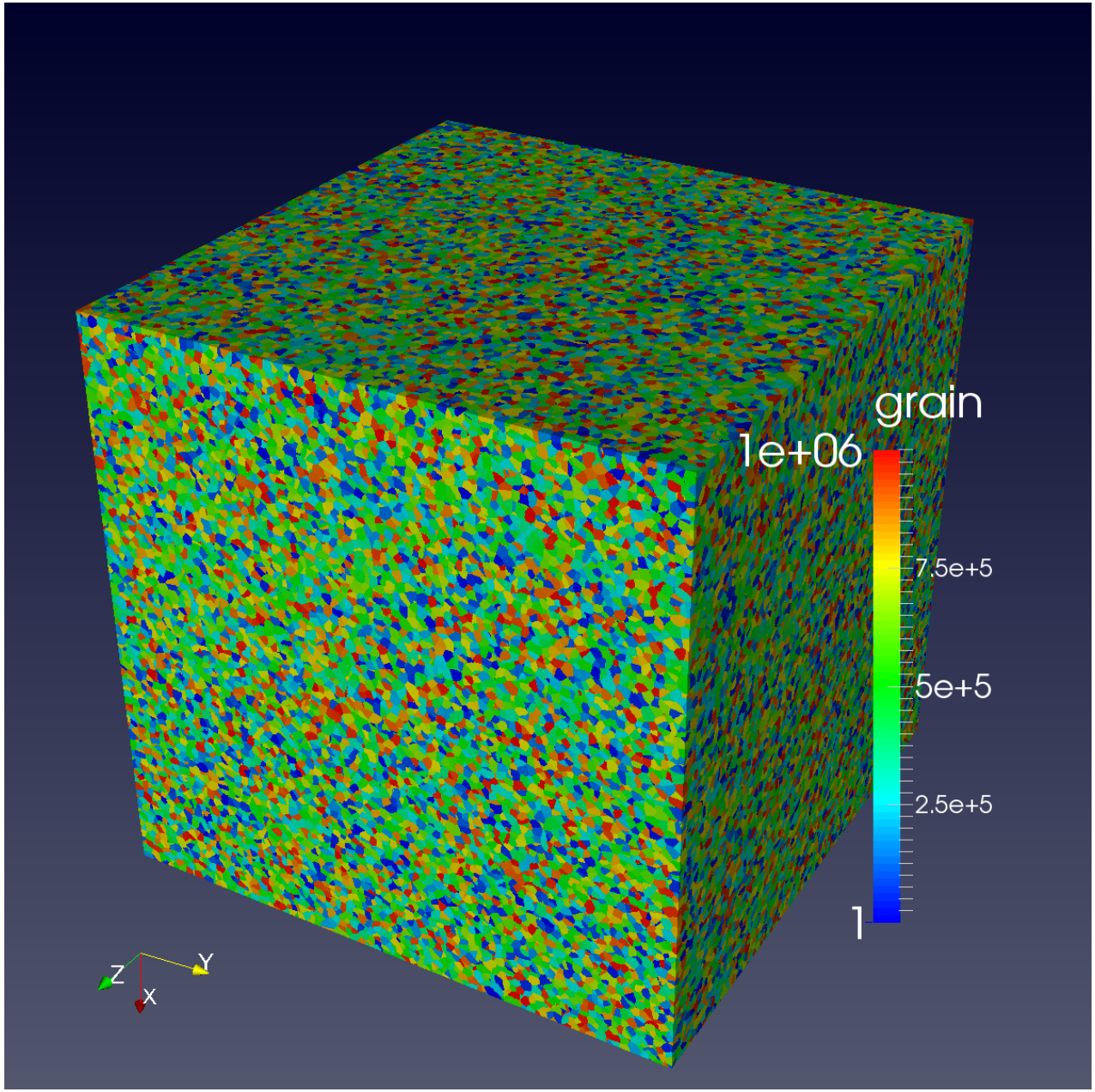
`integer, allocatable :: space(:,:,:,:)[:,:,:]`

- Discrete space, discrete time
- Mesh independent results require $\geq 10^5$ CA cells per crystal on average [5].
- Crystal (grain) is a cluster of cells of the same value.



CGPACK IO - unresolved

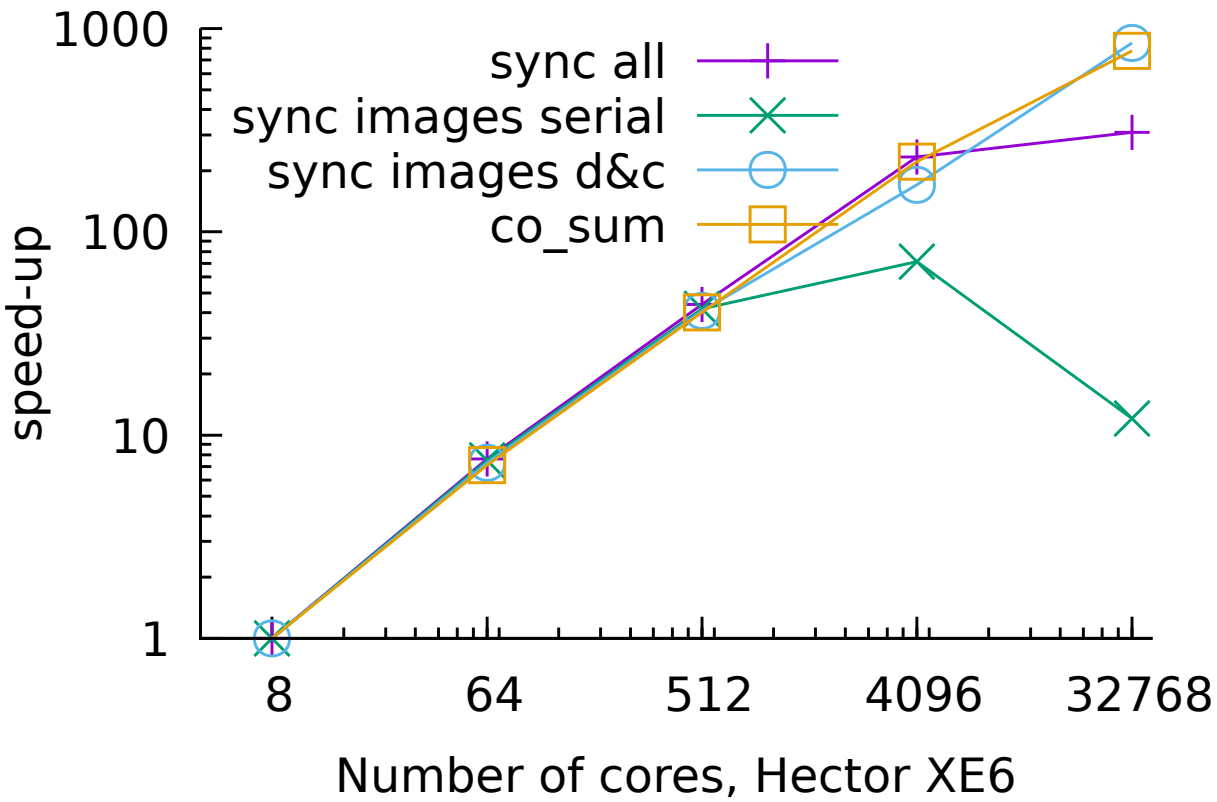
- MPI/IO speeds up to 2.3GB/s on HECToR (Cray XE6) [6].
- MPI/IO can reach 14GB/s on ARCHER (Cray XC30) [7].
- NetCDF (not yet implemented) - higher level of abstraction, sits on top of MPI/IO. [8].



10^6 grains, 10^{11} cells - 400GB dataset, $>$ 4 hours on 1000 ARCHER nodes (24k cores).

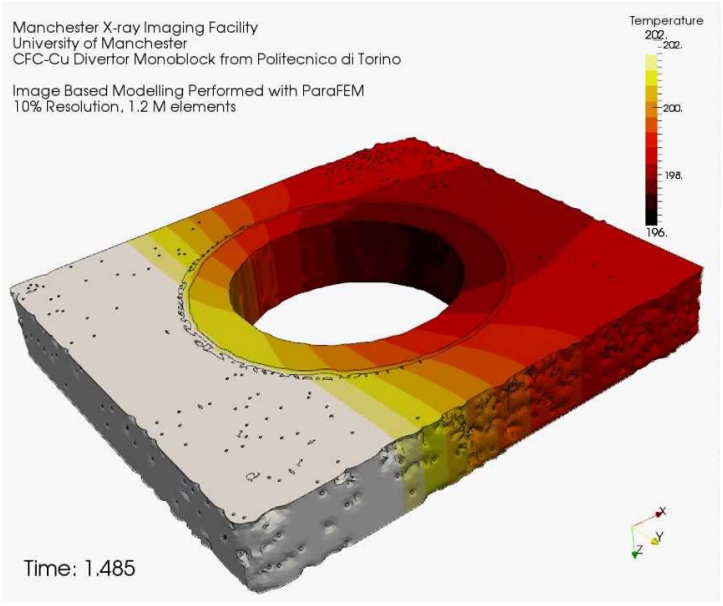
CGPACK scaling

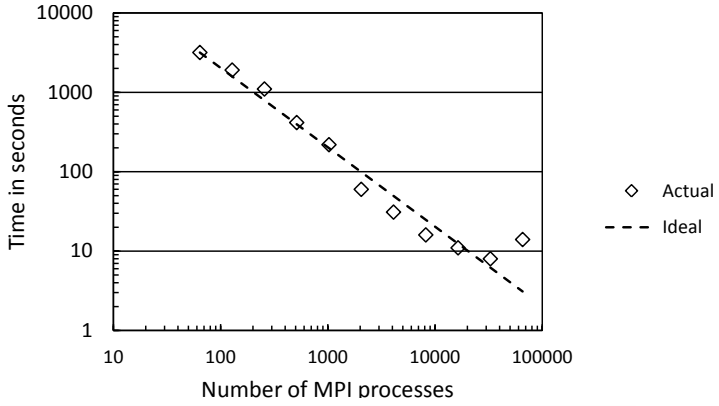
- Up to 32k cores on HECToR and ARCHER for solidification problems.
- Scaling varies for different programs built with CGPACK, depending on which routines are called, in what order and requirements for synchronisation.

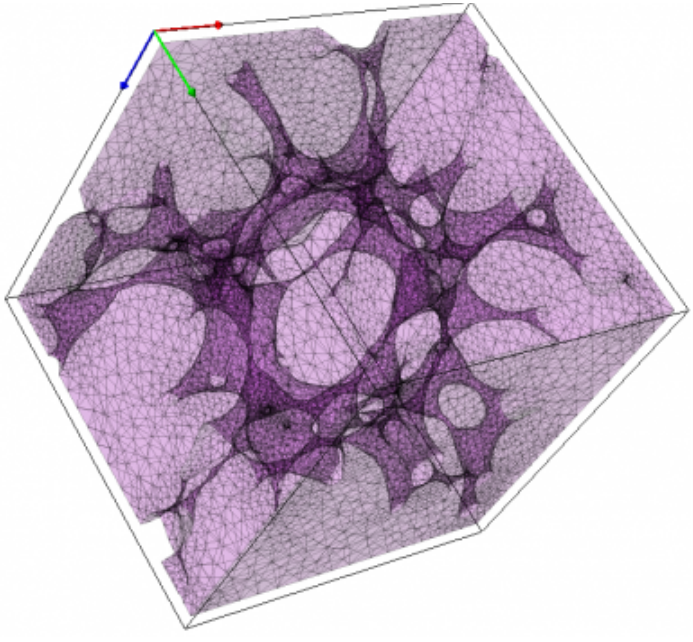


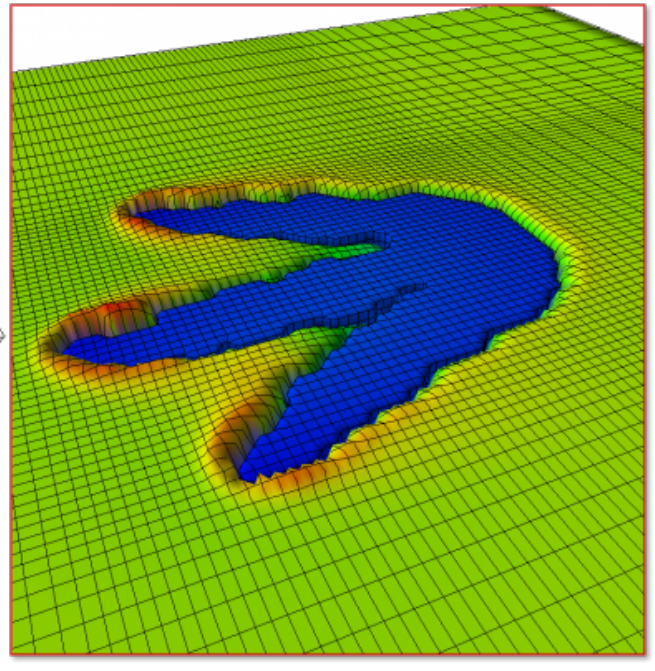
ParaFEM - scalable general purpose finite element library

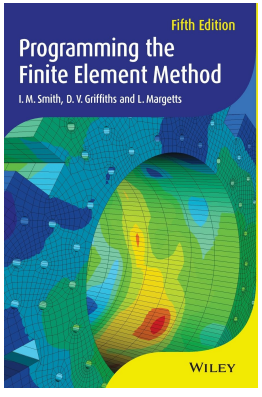
- `http://parafem.org.uk`
- Fortran 90 MPI
- Highly portable, many users [9]
- Excellent scaling
- BSD license





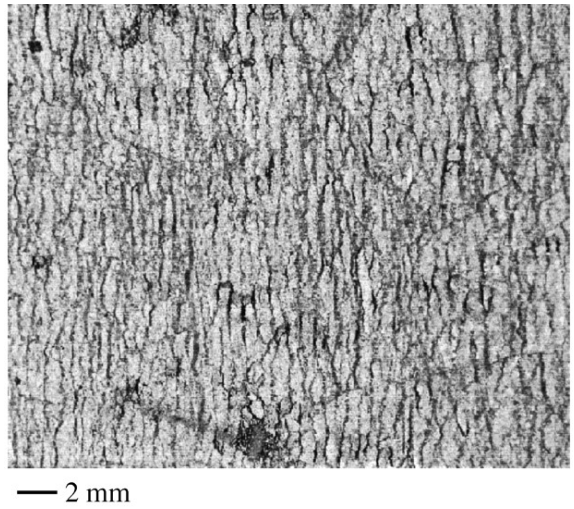


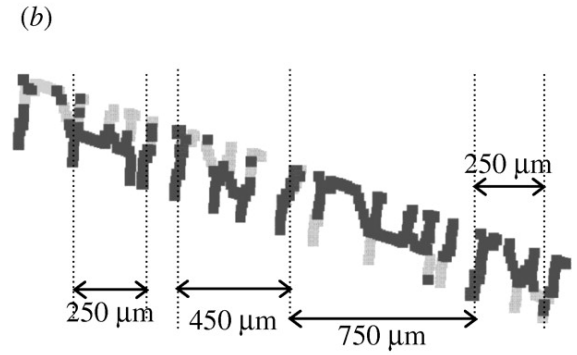


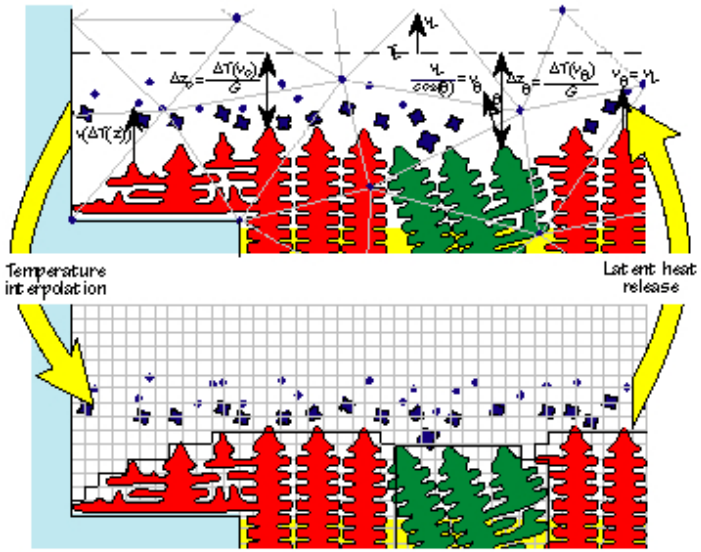


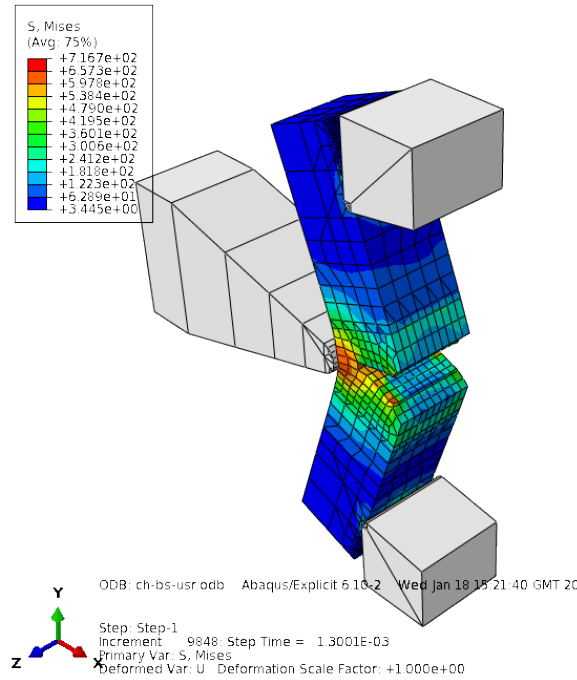
Cellular Automata Finite Element (CAFE)

- Used for solidification [10], recrystallisation [11] and fracture [12, 13].
- FE - continuum mechanics - stress, strain, etc.
- CA - crystals, crystal boundaries, cleavage, grain boundary fracture
- FE $\rightarrow$ CA - stress, strain
- CA $\rightarrow$ FE - damage variables

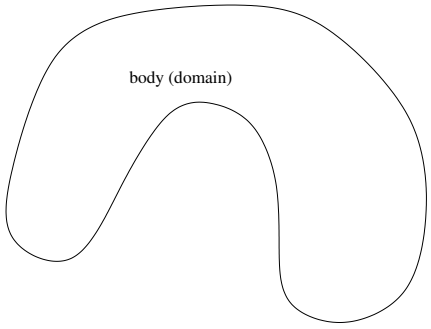






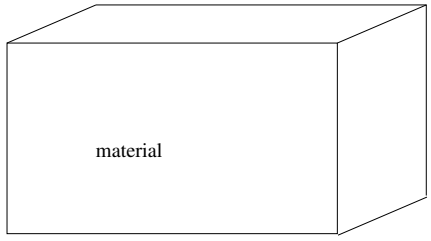


CAFE design: structured CA grid + unstructured FE grid



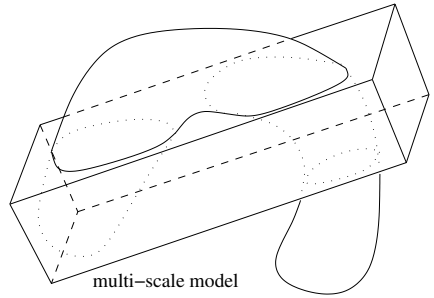
FE

+

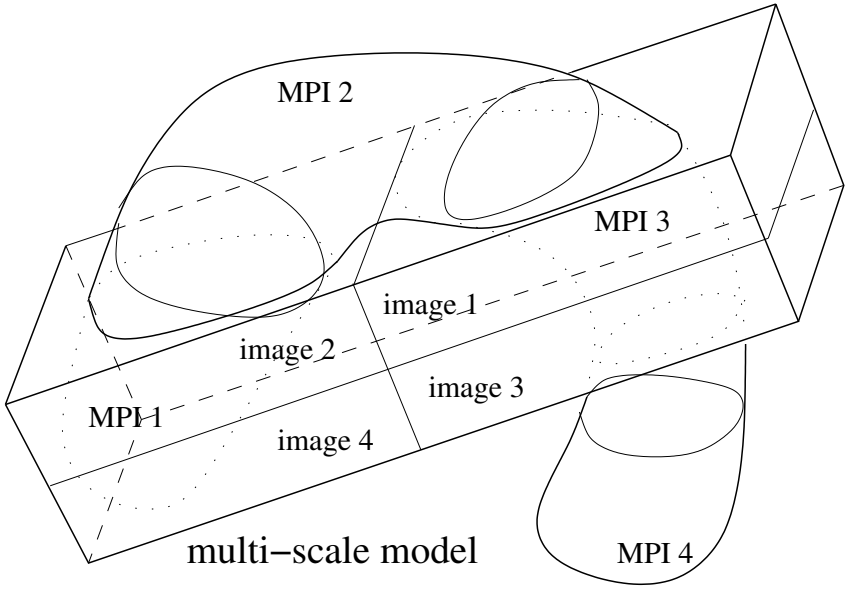


CA

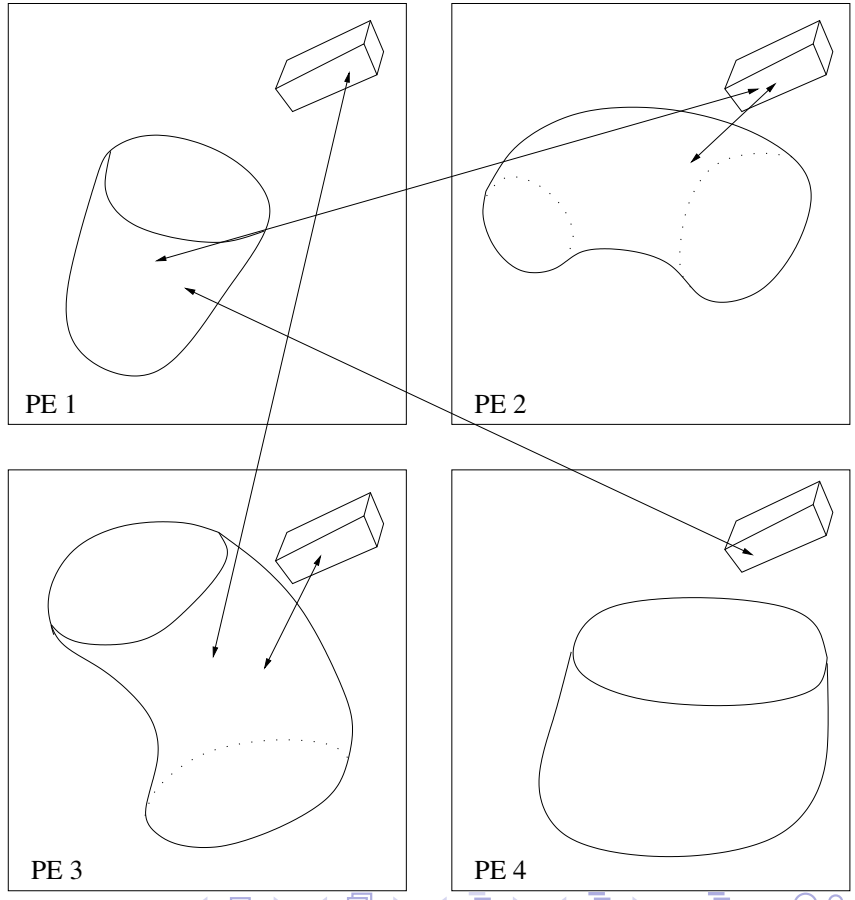
=



CAFE



Example with 4 PE (4 MPI processes, 4 coarray images). Arrows are FE $\leftrightarrow$ CA comms.



FE → CA mapping via a private allocatable array of derived type:

```
type mcen
  integer :: image, elnum
  real :: centr(3)
end type mcen
type( mcen ), allocatable :: lcentr(:)
```

based on coordinates of FE centroids calculated by each MPI process and stored in centroid_tmp coarray:

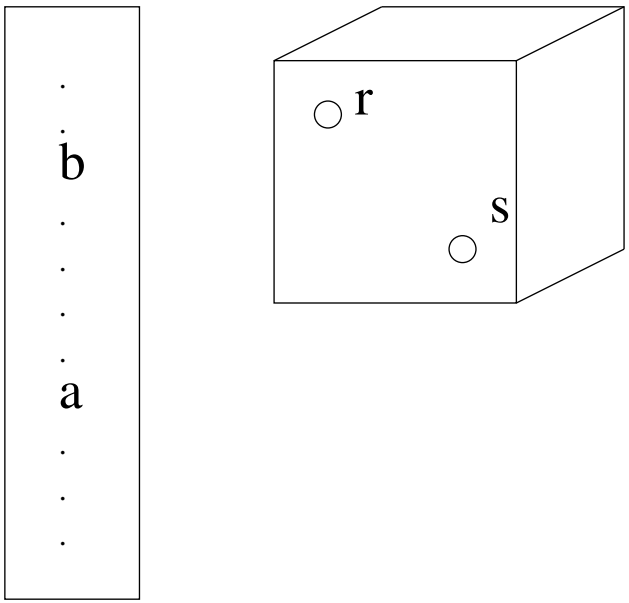
```
type rca
  real, allocatable :: r(:,:)
end type rca
type( rca ) :: centroid_tmp[*]
 :
allocate( centroid_tmp%r(3, nels_pp) )
```

where nels_pp is the number of FE stored on this PE.

lcentr arrays on images P and Q

PE, image, MPI process P

elements CA

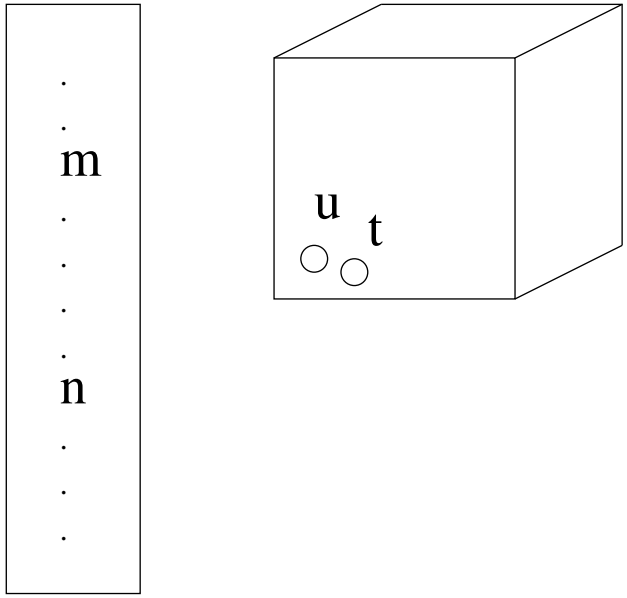


lcentr

image	. . . Q . . . P . . .
elnum	. . . n . . . b . . .
centr	. . . r . . . s . . .

PE, image, MPI process Q

elements CA



lcentr

image	. . . Q . . . P . . .
elnum	. . . m . . . a . . .
centr	. . . u . . . t . . .

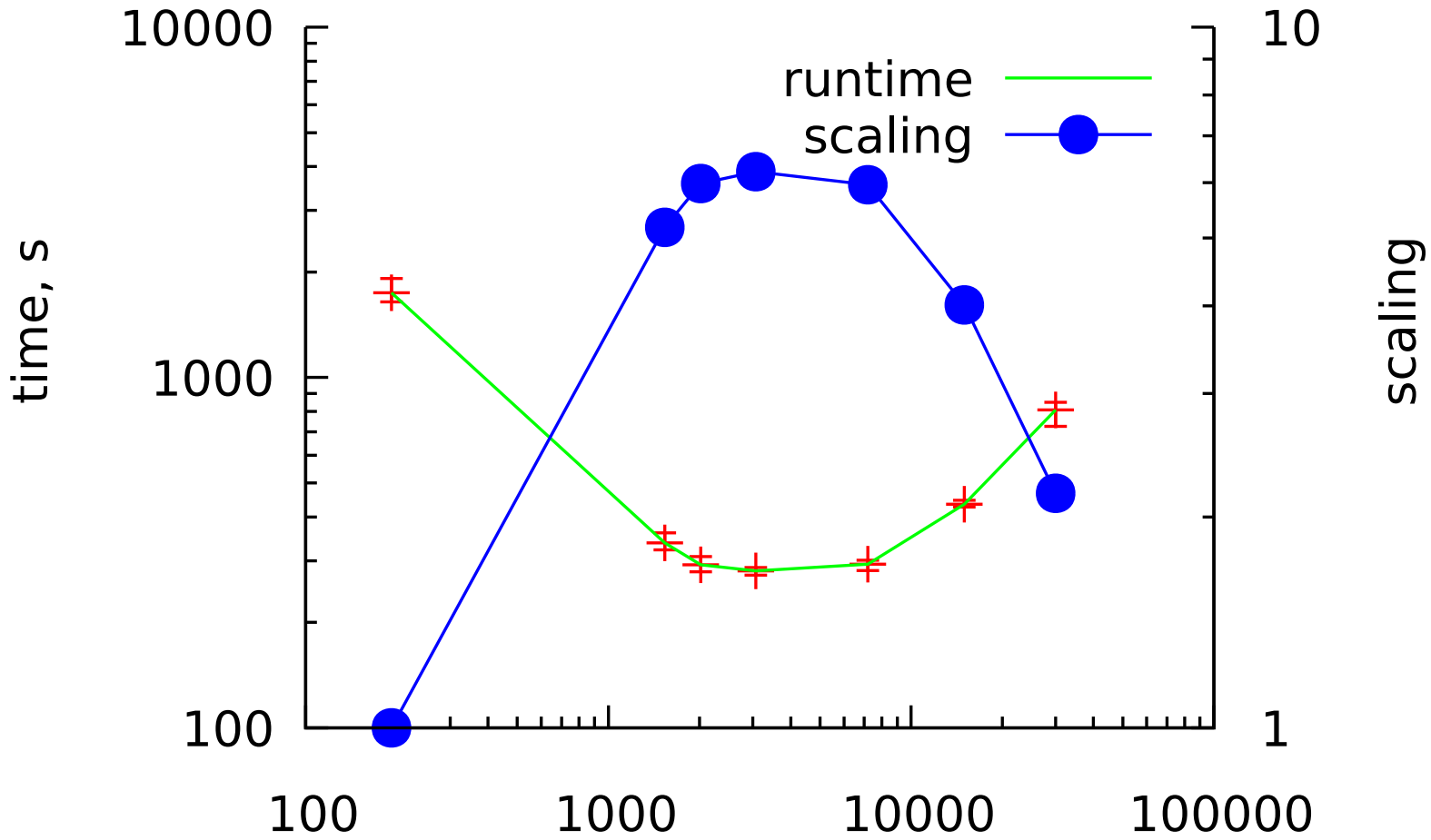
All-to-all vs nearest neighbour for lcentr

- cgca_pfem_cenc - all-to-all routine.
- cgca_pfem_map - nearest neighbour - temporary arrays and coarray collectives CO_SUM and CO_MAX, described in TS 18508 [1] and will be included in the next revision of the Fortran standard, Fortran 2015. At the time of writing coarray collectives are available on Cray systems as extension to the standard [14]. The two routines differ in their use of remote communications.

cgca_pfem_map

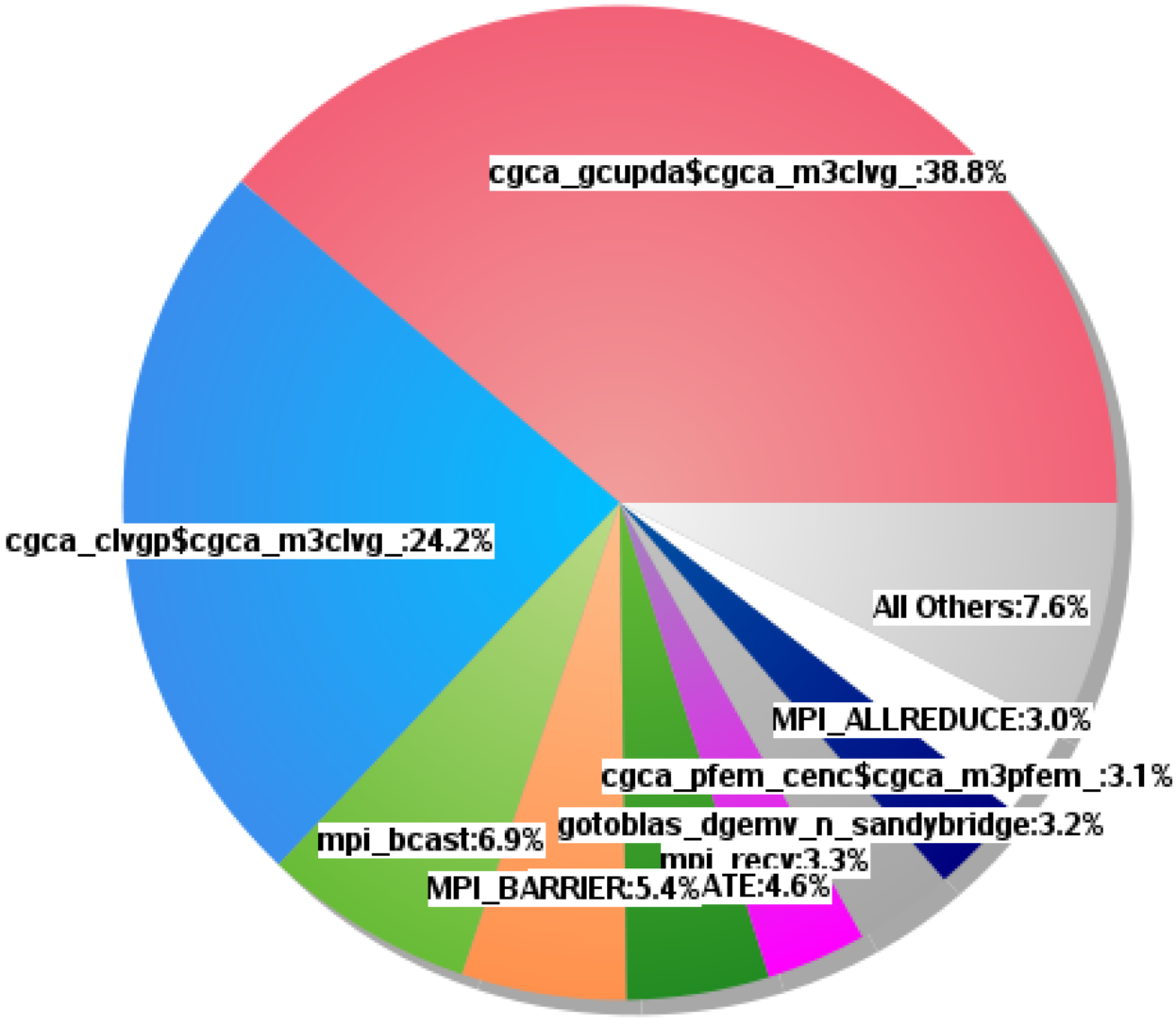
```
integer :: maxfe, start, pend, ctmpsize
real, allocatable :: tmp(:,:)
 ! Calc. the max num. of FE stored on this img
maxfe = size( centroid_tmp%r, dim=2 )
ctmpsize = maxfe
call co_max( source = maxfe )}
allocate(tmp( maxfe*num_images(),5),source=0.0)
 ! Each image writes to a unique portion of tmp
start = (this_image() - 1)*maxfe + 1
pend = start + ctmpsize - 1
tmp(start:pend,1)=real(this_image(),kind=4)
 ! Write element number *as real*
tmp( start : pend, 2 ) = &
  real( (/ (j, j = 1, ctmpsize) /), kind=4 )
 ! Write centroid coord
tmp( start : pend, 3:5 ) = &
  transpose( centroid_tmp%r(:,:) )
call co_sum( source = tmp )
```

Initial CAFE scaling



ParaFEM/CGPACK MPI/coarray miniapp scaling on ARCHER XC30 for a 3D problem with 1M FE and 800M CA cells.

Initial profiling



Profiling function distribution for ParaFEM/CGPACK MPI/coarray miniapp with all-to-all routine cgca_gcupda at 7200 cores.

Initial profiling

```
100.0% | 20,520.4 |    -- |    -- |Total
|-----------------------------------------------------------------
|  71.4% | 14,649.9 |    -- |    -- |USER
||----------------------------------------------------------------
||  38.7% |  7,950.6 | 913.4 | 10.3% |cgca_gcupda$cgca_m3clvg_
||  24.1% |  4,951.2 | 940.8 | 16.0% |cgca_clvgp$cgca_m3clvg_
||   3.1% |    638.0 |  70.0 |  9.9% |cgca_pfem_cenc$cgca_m3pfem_
||   1.8% |    367.5 | 578.5 | 61.2% |cgca_hxi$cgca_m2hx_
||   1.7% |    346.0 | 196.0 | 36.2% |cgca_clvgn$cgca_m3clvg_
||================================================================
|  19.8% |  4,061.4 |    -- |    -- |MPI
||----------------------------------------------------------------
||   6.9% |  1,413.5 | 356.5 | 20.1% |mpi_bcast
||   5.4% |  1,098.3 | 419.7 | 27.7% |MPI_BARRIER
||   3.3% |    670.0 | 322.0 | 32.5% |mpi_recv
||   3.0% |    615.3 |  61.7 |  9.1% |MPI_ALLREDUCE
||================================================================
|   8.8% |  1,797.2 |    -- |    -- |ETC
||----------------------------------------------------------------
||   4.6% |    950.5 |   5.5 |  0.6% |__DEALLOCATE
||   3.2% |    654.2 | 110.8 | 14.5% |gotoblas_dgemv_n_sandybridge
|================================================================
```

Raw profiling data for ParaFEM/CGPACK MPI/coarray miniapp with all-to-all routine cgca_gcupda at 7200 cores.

cgca_gcupda - all-to-all

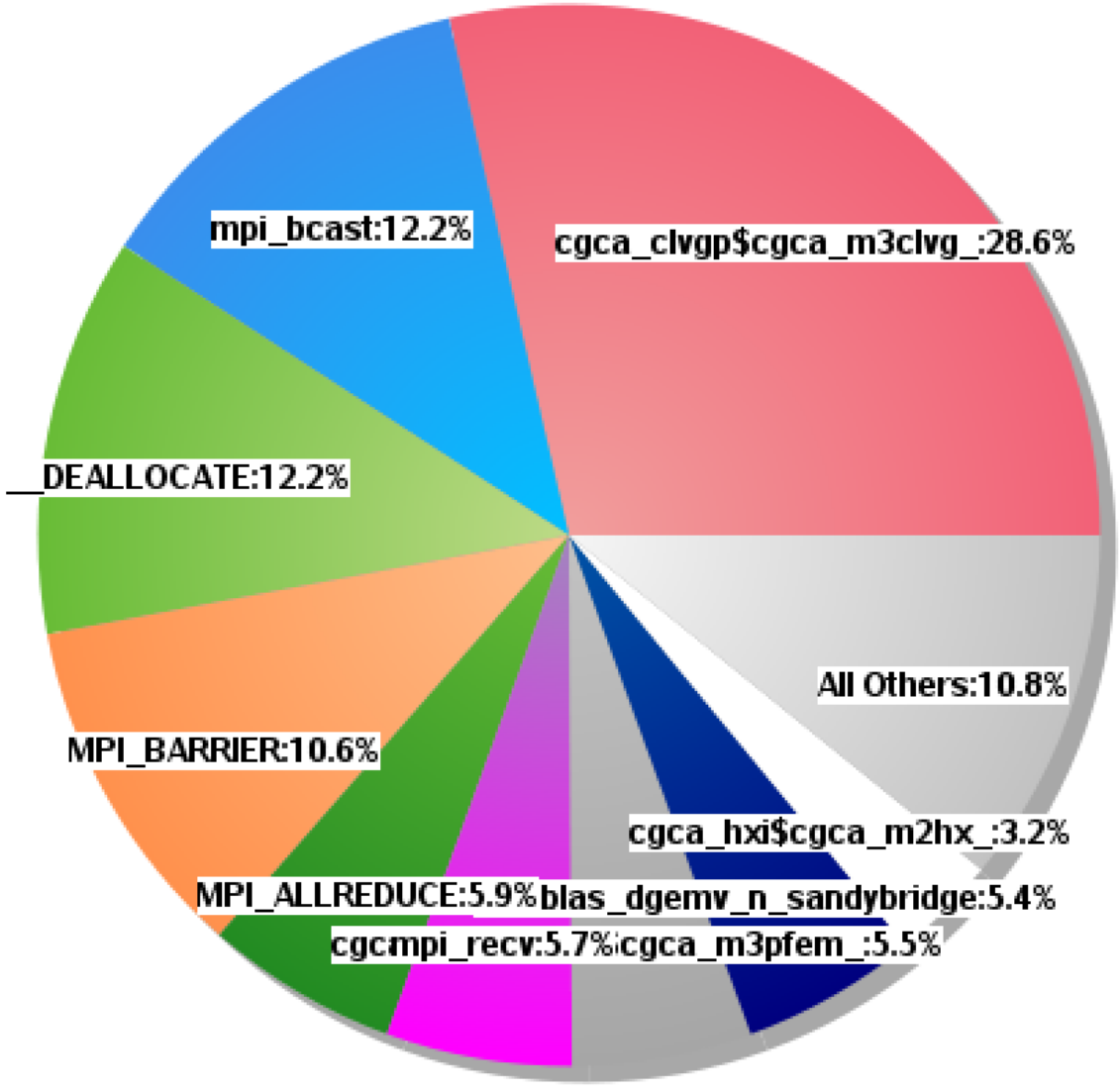
```
integer :: gcupd(100,3)[*], rndint, j, &
           img, gcupd_local(100,3)
real :: rnd
:
call random_number( rnd )
rndint = int( rnd*num_images() ) + 1
do j = rndint, rndint + num_images() - 1
 img = j
 if (img .gt. num_images()) &
       img = img - num_images()
  if (img .eq. this_image()) cycle
  :
  gcupd_local(:,:) = gcupd(:,:)[img]
  :
end do
```

cgca_gcupdn - nearest neighbour

```
do i = -1 , 1
do j = -1 , 1
do k = -1 , 1
 ! Get the coindex set of the neighbour
 ncod = mycod + (/ i , j , k /)
 :
 gcupd_local(:,:) = &
     gcupd(:,:)[ncod(1),ncod(2),ncod(3)]
 :
end do
end do
end do
```

Note: the nearest neighbour must be called *multiple times* to propagate changes from every image to all other images.

Profiling cgca_gcupdn



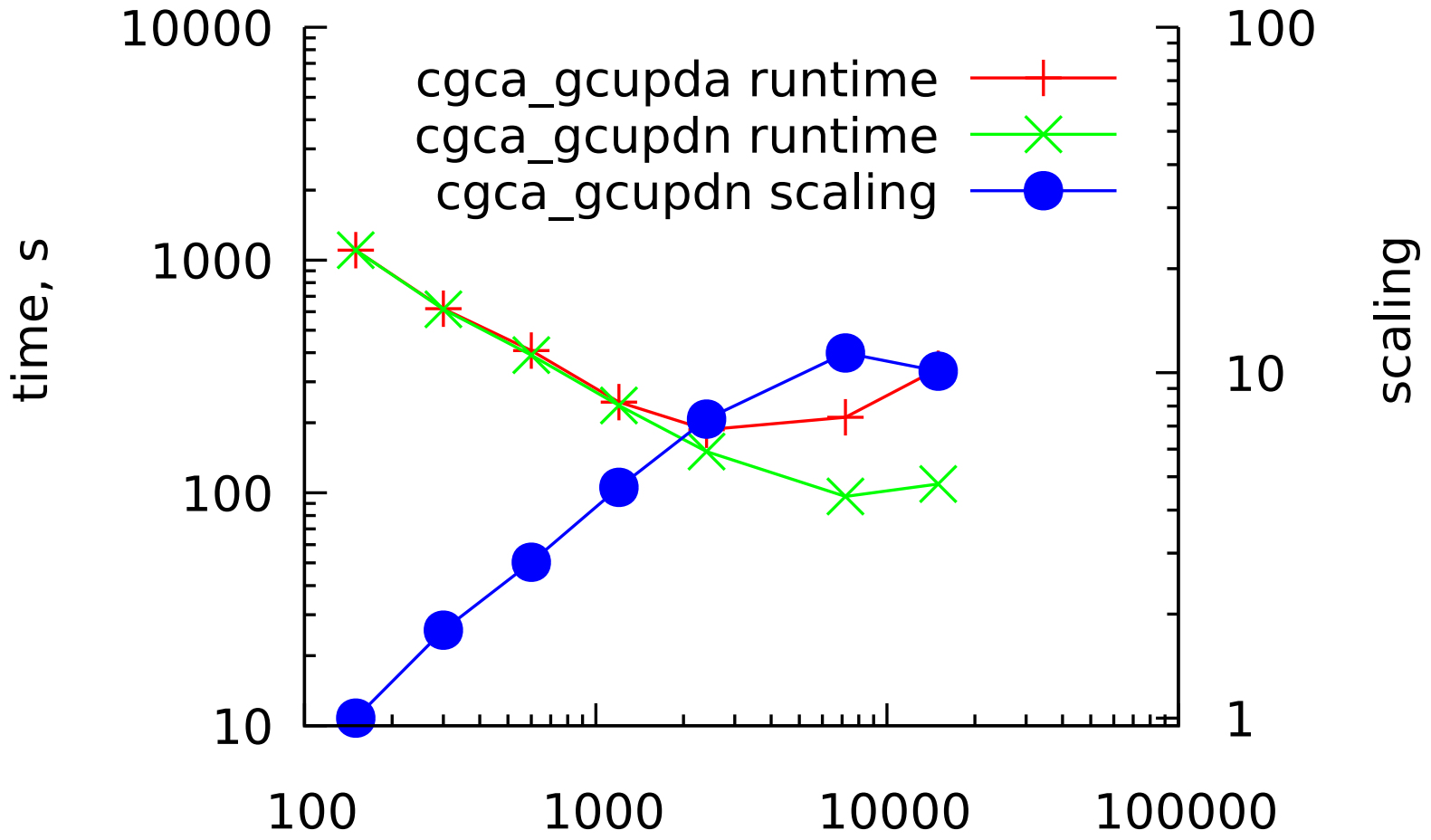
Profiling function distribution for ParaFEM/CGPACK MPI/coarray miniapp with the neareast neighbour routine cgca_gcupdn at 7200 cores.

Profiling cgca_gcupdn

```
100.0% | 12,199.5 |    -- |    -- |Total
|-----------------------------------------------------------
|  44.8% |  5,459.7 |    -- |    -- |USER
||----------------------------------------------------------
||  28.6% |  3,484.0 | 582.0 | 14.3% |cgca_clvgp$cgca_m3clvg_
||   5.5% |    666.1 |  93.9 | 12.4% |cgca_pfem_cenc$cgca_m3pfem_
||   3.2% |    393.1 | 752.9 | 65.7% |cgca_hxi$cgca_m2hx_
||   2.8% |    346.0 | 176.0 | 33.7% |cgca_clvgn$cgca_m3clvg_
||   1.4% |    165.2 |  37.8 | 18.6% |cgca_sld$cgca_m3sld_
||   1.0% |    126.0 |  82.0 | 39.4% |xx14_
||==========================================================
|  36.7% |  4,472.1 |    -- |    -- |MPI
||----------------------------------------------------------
||  12.2% |  1,484.4 | 380.6 | 20.4% |mpi_bcast
||  10.6% |  1,287.9 | 389.1 | 23.2% |MPI_BARRIER
||   5.9% |    714.9 |  90.1 | 11.2% |MPI_ALLREDUCE
||   5.7% |    689.4 | 338.6 | 32.9% |mpi_recv
||   1.5% |    179.1 | 417.9 | 70.0% |MPI_REDUCE
||==========================================================
|  18.5% |  2,256.1 |    -- |    -- |ETC
||----------------------------------------------------------
||  12.1% |  1,480.9 |   4.1 |  0.3% |__DEALLOCATE
||   5.4% |    653.8 |  95.2 | 12.7% |gotoblas_dgemv_n_sandybridge
|==========================================================
```

Raw profiling data for ParaFEM/CGPACK MPI/coarray miniapp with the neareast neighbour routine cgca_gcupdn at 7200 cores.

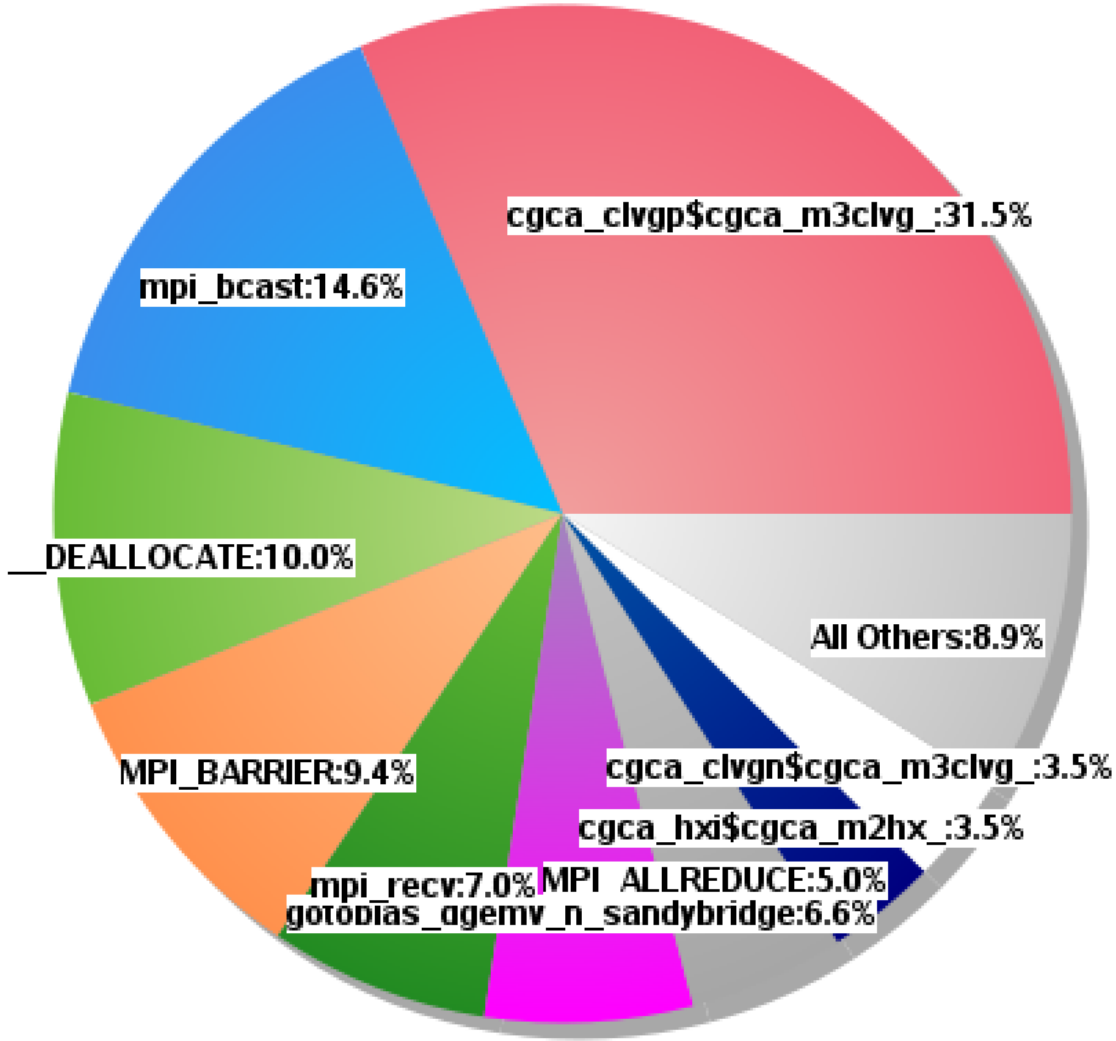
Scaling improvement with cgca_gcupdn over cgca_gcupda



Runtimes and scaling for ParaFEM/CGPACK MPI/coarray miniapp with the nearest neighbour, cgca_gcupdn, and all-to-all, cgca_gcupda, algorithms.

Scaling limit increased from 2k to 7k cores.

Profiling with cgca_pfem_map



Profiling function distribution for ParaFEM/CGPACK MPI/coarray miniapp with cgca_gcupdn and cgca_pfem_map at 7200 cores.

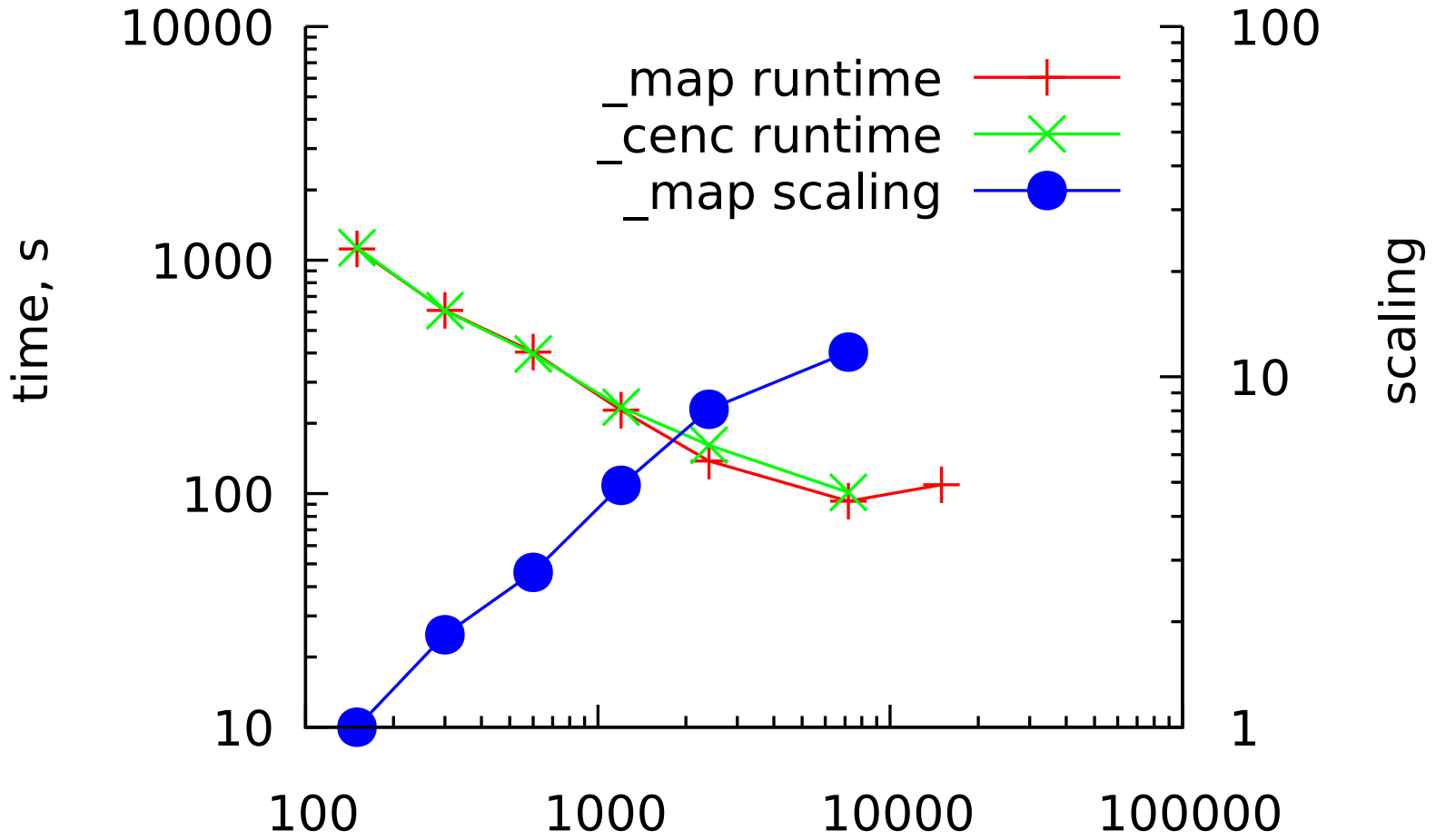
Profiling with cgca_pfem_map

Table 1: Profile by Function

Samp%	Samp	Imb. Samp	Imb. Samp%	Group Function PE=HIDE Thread=HIDE
100.0%	9,903.4	--	--	Total
43.6%	4,321.6	--	--	USER
31.4%	3,110.7	589.3	15.9%	cgca_clvgp$cgca_m3clvg_
3.5%	346.0	513.0	59.7%	cgca_hxi$cgca_m2hx_
3.5%	342.0	175.0	33.8%	cgca_clvgn$cgca_m3clvg_
1.2%	116.3	4.7	3.9%	cgca_pfem_map$cgca_m3pfem_
1.1%	106.8	1,537.2	93.5%	cgca_clvgsd$cgca_m3clvg_
1.0%	99.9	24.1	19.5%	cgca_sld$cgca_m3sld_
38.4%	3,803.6	--	--	MPI
14.6%	1,446.6	350.4	19.5%	mpi_bcast
9.4%	932.4	473.6	33.7%	MPI_BARRIER
7.0%	689.5	371.5	35.0%	mpi_recv
4.9%	489.3	76.7	13.6%	MPI_ALLREDUCE
1.5%	145.4	314.6	68.4%	MPI_REDUCE
17.8%	1,766.8	--	--	ETC
9.9%	983.9	8.1	0.8%	__DEALLOCATE
6.6%	652.3	93.7	12.6%	gotoblas_dgemv_n_sandybridge

Raw profiling data for ParaFEM/CG-PACK MPI/coarray miniapp with cgca_gcupdn and cgca_pfem_map at 7200 cores.

Profiling with cgca_pfem_map



Runtimes and scaling for ParaFEM/CGPACK MPI/coarray miniapp with cgca_pfem_map and cgca_pfem_cenc. cgca_pfem_map or cgca_pfem_cenc are called only once during the execution of the miniapp. Hence only a minor improvement is obtained, only from about 1000 cores.

Issues with CrayPAT

```
|  71.4% | 14,649.9 |    -- |    -- |USER
||-----------------------------------------------------------------
||  38.7% |  7,950.6 | 913.4 | 10.3% |cgca_gcupda$cgca_m3clvg_
||  24.1% |  4,951.2 | 940.8 | 16.0% |cgca_clvgp$cgca_m3clvg_
||   3.1% |    638.0 |  70.0 |  9.9% |cgca_pfem_cenc$cgca_m3pfem_
||   1.8% |    367.5 | 578.5 | 61.2% |cgca_hxi$cgca_m2hx_
||   1.7% |    346.0 | 196.0 | 36.2% |cgca_clvgn$cgca_m3clvg_
||=================================================================
```

cgca_gcupda is top in sampling results, but is absent from tracing. It is called the same number of times as cgca_hxi.

```
|  29.7% |  99.743118 |       -- |    -- |  5,226,813.1 |USER
||-----------------------------------------------------------------------------
||  17.4% |  58.326659 | 36.082315 | 38.2% |          5.0 |cgca_clvgp$cgca_m3clvg_
||   5.6% |  18.876152 |  5.062089 | 21.1% |          1.0 |cgca_pfem_cenc$cgca_m3pfem_
||   3.3% |  11.145318 | 15.328335 | 57.9% |          1.0 |xx14_
||   1.7% |   5.705317 |  8.788733 | 60.6% |  5,224,771.1 |cgca_clvgn$cgca_m3clvg_
||   1.7% |   5.689672 |  1.910819 | 25.1% |      2,035.0 |cgca_hxi$cgca_m2hx_
||=============================================================================
```

Issues with CrayPAT

All profiling was done with single thread.

```
CrayPat/X:  Version 6.2.2 Revision 13378 (xf 13240)  11/20/14 14:32:58
Number of PEs (MPI ranks):   480

Numbers of PEs per Node:      24  PEs on each of  20  Nodes

Numbers of Threads per PE:     3

Number of Cores per Socket:   12
Execution start time:  Thu Mar  3 13:40:17 2016
System name and speed:  tdsmom 2701 MHz
```

Incorrect number of threads indentified by CrayPAT in a tracing experiment of ParaFEM/CGPACK MPI/coarray miniapp with cgca_gcupda.

Future work - optimisation of coarray synchronisation

```
! ===>>> implicit sync all inside <<<===
call cgca_sld(cgca_space, .false. ,0,10,cgca_solid)
call cgca_igb( cgca_space )
call cgca_gbs( cgca_space )
call cgca_hxi( cgca_space )
sync all
call cgca_gcu( cgca_space )
sync all
```

- Some routines have sync inside.
- Other sync responsibility is left to the end user.
- Over-synchronisation?
- Enough sync is required by the standard. A standard conforming Fortran coarray program will not deadlock or suffer races.


ISO/IEC JTC1/SC22/WG5 N2074, *TS 18508 Additional Parallel Features in Fortran*, 2015.


A. Shterenlikht and L. Margetts, "Three-dimensional cellular automata modelling of cleavage propagation across crystal boundaries in polycrystalline microstructures," *Proc. Roy. Soc. A*, vol. 471, p. 20150039, 2015. [Online]. Available: http://eis.bris.ac.uk/~mexas/pub/2015prsa.pdf


A. Shterenlikht, L. Margetts, L. Cebamanos, and D. Henty, "Fortran 2008 coarrays," *ACM Fortran Forum*, vol. 34, pp. 10–30, 2015. [Online]. Available: http://eis.bris.ac.uk/~mexas/pub/2015acmff.pdf


A. Shterenlikht, "Fortran coarray library for 3D cellular automata microstructure simulation," in *Proc. 7th PGAS Conf., 3-4 October 2013, Edinburgh, Scotland, UK*, M. Weiland, A. Jackson, and N. Johnson, Eds. The University of Edinburgh, 2014, pp. 16–24. [Online]. Available: http://www.pgas2013.org.uk/sites/default/files/pgas2013proceedings.pdf


J. Phillips, A. Shterenlikht, and M. J. Pavier, "Cellular automata modelling of nano-crystalline instability," in *Proc. 20th UK ACME Conf. 27-28 March 2012, The University of Manchester, UK*, 2012. [Online]. Available: http://eis.bris.ac.uk/~mexas/pub/2012_ACME.pdf


A. Shterenlikht, "Fortran 2008 coarrays," Invited talk at Fortran specialist group meeting, BCS/IoP joint meeting, 2014. [Online]. Available: http://eis.bris.ac.uk/~mexas/pub/coar_bcs.pdf


D. Henty, A. Jackson, C. Moulinec, and V. Szeremi, " Performance of Parallel IO on ARCHER, version 1.1," ARCHER White Papers, 2015. [Online]. Available:

http://archer.ac.uk/documentation/white-papers/parallelIO/ARCHER_wp_parallelIO.pdf


T. Collins, "Using NETCDF with Fortran on ARCHER , version 1.1," ARCHER White Papers, 2016. [Online]. Available: http://archer.ac.uk/documentation/white-papers/fortanIO_netCDF/fortranIO_netCDF.pdf


I. M. Smith, D. V. Griffiths, and L. Margetts, *Programming the Finite Element Method*, 5th ed. Wiley, 2014.


C. A. Gandin and M. Rappaz, "A coupled finite element-cellular automaton model for the prediction of dendritic grain structures in solidification processes," *Acta Met. and Mat.*, vol. 42, no. 7, pp. 2233–2246, 1994.


C. Zheng and D. Raabe, "Interaction between recrystallization and phase transformation during intercritical annealing in a cold-rolled dual-phase steel: A cellular automaton model," *Acta Materialia*, vol. 61, pp. 5504–5517, 2013.


A. Shterenlikht and I. C. Howard, "The CAFE model of fracture – application to a TMCR steel," *Fatigue Fract. Eng. Mater. Struct.*, vol. 29, pp. 770–787, 2006.


S. Das, A. Shterenlikht, I. C. Howard, and E. J. Palmiere, "A general method for coupling microstructural response with structural performance," *Proc. Roy. Soc. A*, vol. 462, pp. 2085–2096, 2006.


ISO/IEC 1539-1:2010, *Fortran – Part 1: Base language, International Standard*, 2010.