Shterenlikht, A., Margetts, L., & Cebamanos, L. (2017). Fortran coarray/MPI Multi-scale CAFE for fracture in heterogeneous materials. In P. Iványi, B. H. V. Topping, & G. Várady (Eds.), *Proceedings of the Fifth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering* [40] (Civil-Comp Proceedings; Vol. 111). Civil-Comp Press. <https://doi.org/10.4203/ccp.111.40>

Peer reviewed version

Link to published version (if available):
[10.4203/ccp.111.40](https://doi.org/10.4203/ccp.111.40)

[Link to publication record in Explore Bristol Research](#)
PDF-document

This is the author accepted manuscript (AAM). The final published version (version of record) is available online via Civil-Comp Press at <http://www.ctresources.info/ccp/paper.html?id=9262>. Please refer to any applicable terms of use of the publisher.

University of Bristol - Explore Bristol Research

General rights

This document is made available in accordance with publisher policies. Please cite only the published version using the reference above. Full terms of use are available:
<http://www.bristol.ac.uk/red/research-policy/pure/user-guides/ebr-terms/>

Abstract

All real materials are heterogeneous, e.g. polycrystalline metal alloys, reinforced concrete, carbon fibre reinforced polymer (CFRP), wood, nuclear graphite or bone. Modelling of such materials involves concurrent simulation of multiple interacting and competing physical processes, acting at different length and time scales, e.g. dislocation flow, ply debonding or separation of atomic layers. In this work we demonstrate a multi-scale fracture framework where cellular automata (CA) represents material evolution, deformation and fracture at micro- or nano-scales and finite elements (FE) are used at structural scales. Fortran coarrays offer simple and intuitive data structures for 3D CA modelling of material microstructures. Fortran 2008 and 2015 coarrays are native Fortran means for SPMD style of parallel programming. CA is a structured grid and thus is well suited for implementation in coarrays. Design of a coarray cellular automata microstructure evolution library CGPACK is described. Simulations of solidification, recrystallisation and grain coarsening, and fracture can be performed at arbitrary length and time scales with CGPACK. We show how coarrays can be used together with an MPI FE library to create a two-way concurrent hierarchical and scalable multi-scale CAFE deformation and fracture framework. A highly portable MPI FE library ParaFEM was used in this work. Both CGPACK and ParaFEM are developed and distributed under BSD license, allowing free use, modification and redistribution in academia and for profit. The CAFE framework is based on mapping coarrays to MPI data structures. There are identical numbers of MPI ranks and coarray images in the framework. Data stored in coarray variables on each coarray image is mapped onto data stored on each MPI process, based on fact that the material and the structure occupy the same physical space. Continuum mechanics quantities, e.g. stress and strain tensors are passed from FE integration points to the CA, where they are distributed over cells (localisation) based on existing damage and microstructure heterogeneity. After each fracture propagation increment at the CA scale, the microstructural damage is encoded in a scalar damage variable (homogenisation) which is passed back to

the FE integration point. The stiffness (Young's modulus) of the integration point is scaled by the damage variable, so that the damage variable of 1 means no damage, and the damage variable of 0 means no remaining load bearing capacity. We show that independently CGPACK and ParaFEM programs can scale up well into tens of thousands of cores on Cray systems. Strong scaling of a hybrid ParaFEM/CGPACK MPI/coarray multi-scale framework was measured on Cray a simulation of a fracture of a steel round bar under tension. That program scales up to 7,000 cores. The UK national HPC system, ARCHER, Cray XC30, was used in this work. We conclude with a comparative analysis of synchronisation requirements in MPI and coarray programs. Coarrays is an example of a partitioned global address space (PGAS) programming model, with single sided communication, where a remote read or write operation can be performed by a single process/image, with no cooperation or even knowledge of the process whose data is being accessed. The single sided communication strategy is very different from two-way MPI send/receive communication. It poses specific challenges for synchronisation, data integrity and performance in coarray programs. We discuss these challenges and propose several possible solutions for future work.

Keywords: fracture, heterogeneous materials, multi-scale, Fortran coarrays, MPI, scaling, profiling

1 Introduction

Deformation and fracture problems of solid mechanics often involve multiple competing physical processes occurring at different time and length scales. Examples of such processes are phase transformation (10ps, 10nm), dislocation nucleation and propagation (10ps, 50nm), twin formation (1ns, 1nm), interaction of dislocations (100ns, 100nm), secondary microcrack nucleation in the process zone (10ns, 100 μ m), adiabatic shear (10 μ s, 100 μ m). However, engineering scale crack growth and component failure typically occur at much higher time and length scales (10^{-3} to 10^1 s, 10^{-3} to 10^1 m). A variety of multi-scale modelling approaches aimed to link different time and length scales together into a coherent model to deliver engineering scale predictions have been proposed to treat such problems, e.g. combined atomistic and continuum mechanics [1], molecular dynamics and continuum mechanics [2], discrete dislocation and continuum plasticity [3], etc. The cellular automata (CA) method has been used together with finite elements (FE), in a multi-scale cellular automata finite element (CAFE) framework for problems involving material microstructure, such as solidification [4], recrystallisation [5] or fracture of polycrystals [6–9]. FE is used to solve the continuum mechanics problem (coarse scale) to calculate the macroscopic quantities, such as the strain, stress or temperature gradients, while the microstructure (fine scale) is updated with the CA method. At each iteration of the CAFE model continuum mechanics quantities are passed from the coarse FE scale to the fine CA scale (localisation) and damage variables are passed from the CA scale back to the FE scale (homogenisation) [10]. Thus CAFE is a two-way hierarchical concurrent multi-scale

framework [11].

In the 3D CAFE method the Cartesian coordinate system is used and space is partitioned into identical cubic cells. Cells have physically meaningful states, e.g. liquid phase, intact crystal with a certain rotation tensor, crack front, crack flank, cleavage plane of a particular type, etc. The state of each cell at the next iteration is determined by the state of that cell, the states of its immediate neighbourhood cells, (e.g. the 26-cell Moore’s neighbourhood) and some continuum FE field variables (e.g. stress, strain or temperature), all taken at the current iteration. The CA method is very simple. CA has an explicitly local domain of influence, with no global equilibrium requirements, which opens opportunities for parallelisation. Each cell can be updated independently - in parallel. A much higher CA resolution can be achieved compared to the FE method, for the same computational cost.

In the following sections we describe how the CAFE model was implemented using a CA library implemented in Fortran 2008 coarrays and a FE library implemented in MPI. We show examples of microstructures simulated using the CA approach and describe the information flow between CA (coarrays) and FE (MPI). We present strong scaling results for multi-scale models of progressive brittle crack propagation in steel components. We conclude with a discussion of optimisation possibilities of a hybrid coarray/MPI code, and challenges posed by choosing an optimal synchronisation strategy for a coarray library.

2 Fortran coarrays

A very brief introduction to Fortran coarrays is given in this section. For more details refer to [12, 13]. Fortran coarrays have been used as an extension to the standard for over 20 years, mostly on Cray systems. Their appeal to users increased substantially when they were standardised in 2010 [13]. The next Fortran standard, informally called Fortran 2015, will significantly extend coarrays facilities with the addition of collectives, teams, events and facilities for dealing with hardware or software failures.

Coarrays are a native Fortran means for single program multiple data (SPMD) type programming. A coarray is an extension of a Fortran array data object in SPMD context. Square bracket syntax is used to define or refer to a coarray object:

```
integer :: i , ic [*] , k(10,10) , kc(10,10)[*]
real , allocatable :: r(:,:,:) , rc(:,:,:) [:,:,:]
```

where all variables with `[ ]` are coarray variables. At run time a certain number of identical copies of the executable (called *images*) are created by the operating system, which are executing asynchronously. Each image has read/write access to coarray variables on all other images:

```
ic[5] = i ! the invoking image copies its value of i
          ! to variable ic on image 5 (remote write)
```

```

! allocate coarray variable rc on all images
allocate( rc(3,3,3)[5,5,*] )

! the invoking image copies the whole of array rc from
! image with coindex set [1,2,3] to its own copy of
! array r (remote read)
r(:, :, :) = rc(:, :, :)[1,2,3]

```

The standard defines *execution segments* in a Fortran coarray program, which are separated by *image control* statements, such as SYNC ALL or SYNC IMAGES. SYNC ALL statement acts as a global barrier, similar to MPI_barrier.

Coarrays can coexist with other parallel technologies, such as MPI or OpenMP, although to date there are only a few examples of such hybrid codes. The European Centre for Medium-range Weather Forecasts (ECMWF) has used coarrays in combination with MPI and OpenMP in their codes [14]. Coarrays have been used together with OpenMP in plasma codes [15].

3 Cellular Automata modelling of microstructure

A polycrystalline grain microstructure is created by a simple solidification process. All cells are initially considered to be of liquid state, A certain number of randomly chosen cells represent grain nuclei. These are assigned states representing a randomly chosen orientation tensor. At each iteration of the solidification process a liquid cell can acquire the state of one of the 26 randomly chosen neighbours. This process is continuing until there are no liquid cells left in the model. Both fixed and self-similar boundary conditions can be used [6, 16, 17].

A 3D CA space with cubic cells of discrete states maps perfectly onto a 3D integer array coarray, hence coarrays are a natural implementation choice for CA models. In contrast the FE part of the CAFE model, which implements the coarse scale continuum solid mechanics calculations, typically has irregular boundaries. Most often MPI is used to implement parallel Lagrangian FE solvers.

CGPACK is a scalable CA library written in Fortran 2008 with extensive use of coarrays, `cgpacck.sf.net`. Work on CGPACK started in 2013 on HECToR, Cray XE6, then the UK national HPC system [18]. CGPACK has since been ported to Intel and OpenCoarray/GCC platforms.

An example of an equiaxed microstructure simulated with CGPACK is shown in Figure 1 where the colour denotes the orientation of each grain (single crystal). This dataset had 10^6 grains simulated with 10^{11} CA cells, This model represents approximately a $18 \times 18 \times 18$ mm volume of steel with the mean grain size, $\bar{d}$ of 0.2mm.

Figure 1: Schematics of the CA space (left) and an example of the simulated equiaxed microstructure (right)

4 The coarray/MPI CAFE framework

4.1 The FE part - ParaFEM

ParaFEM is a highly scalable and portable MPI FE library written in Fortran 2003, parafem.org.uk [19]. It is the latest extension of the sequential FE libraries originally written by Professor Ian Smith and first published in the 1980s [20]. Interestingly, at that time, they were distributed as open source on tape by NAG Ltd. The software comprises modules, subroutines, functions and around 70 example mini-apps [21]. The mini-apps are typically 2-4 pages long and are used to solve a variety of common engineering problems. The mini-app philosophy enables customisation by engineers, a feature that has enabled the work presented herein to be carried out with a reasonable amount of software development effort.

The parallelisation strategy adopted in ParaFEM involves working element-by-element at each stage of the finite element process, including building element stiffness matrices, solving the system of equations and recovering stress values (post-processing). No global matrix is ever assembled and so domain decomposition is avoided. Each MPI process is allocated an equal number of finite elements, balancing both computational load and memory usage. Parallel element-by-element versions of different iterative solvers are used for different problem types. These work in essentially the same way as their sequential counterparts [22], with the only difference being the need to pass messages between MPI processes when operating on distributed data structures.

The approach has been successful in solving a variety of problem types, from non-linear material behaviour [23] to coupled systems involving multiphysics, such as Biot consolidation and magneto-hydrodynamics [24]. The software has led to scientific advances in a range of disciplines such as Nuclear Engineering [25, 26], Biomechan-

ics [27, 28], Geomechanics [29] and Palaeontology [30].

Both ParaFEM and CGPACK are being actively developed, including contributions from the UK Software Sustainability Institute, `software.ac.uk` [31], and grants from the embedded CSE programme of ARCHER, the current UK national HPC system. Both ParaFEM and CGPACK libraries are distributed under BSD license.

4.2 Size of the CA coarray

CGPACK module `cgca_m2phys` deals with physical units and sizing of the main CA coarray. The 3D CA space is used to represent a rectilinear volume of material microstructure, of physical dimensions $l_1 \times l_2 \times l_3$, see Figure 1. The CA space is implemented as a 4D integer allocatable array `coarray`, with a 3D coindex set. The first 3 array dimensions identify a particular CA cell. The fourth array dimension is used to store multiple types of microstructural information, e.g. grains or damage [32]:

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

The exact dimensions and codimensions of the coarray `space` are chosen at run-time, based on the available number of images, N . First the codimensions are chosen, c_1, c_2, c_3 , such that $c_1 \times c_2 \times c_3 = N$. Arbitrarily we set $c_1 \geq c_2 \geq c_3$. The codimensions are chosen to minimise $c_1 - c_3$, i.e. to make the coarray grid as ‘cubic’ as possible. This is advantageous because it minimises the total number of halo cells, and thus the amount of remote data transfer. The quality of partitioning the microstructure into a 3D array of images is assessed by $q = 1 - (c_1 - c_3)/(N - 1)$, so that $q = 1$ means $c_1 = c_3$, i.e. the lowest possible number of halo cells while $q = 0$ means that $c_1 = N, c_2 = c_3 = 1$, indicating that the number of halo cells is maximised.

Prior work showed that mesh independent CA results are achieved when each crystal (grain) is represented by at least 10^5 cells on average [17]. Then, given the desired microstructure mean grain size, $\bar{d}$, the first 3 dimensions of `space` are calculated.

As an example consider a simulation of a $12 \times 12 \times 20$ mm volume of polycrystalline microstructure with $\bar{d} = 2$ mm on 192 images. Array `space` with 2 types of microstructural information is then allocated as:

```
allocate ( space (35 , 70 , 77 , 2) [ 8 , 4 , * ] )
```

where $c_3 = 6$. This allows for simulating 360 grains with $q = 0.98$, with the linear resolution of 23.2 cells per mm. The total size of the CA model is $280 \times 280 \times 462 \approx 36$ million cells. In general it is not possible to represent physical space with the exact given dimensions, with the same linear resolution along each coordinate axis, as a discrete CA space. In this example, the volume of microstructure that is actually simulated is $12.06 \times 12.06 \times 19.91$ mm.

Figure 2: Schematics of the the FE domain

Figure 3: Schematics of a multi-scale CAFE model composed of the FE domain superimposed with the CA material space (left) and a possible partition of the multi-scale model on 4 PEs (right)

4.3 Establishing the CA to FE mapping

CGPACK module `cgca_m3pfem` contains data structures and subroutines which establish a mapping between the CA space and the FE mesh. A schematic example of an irregular FE domain is shown in Figure 2. Sometimes, the CA space will be fully inside the FE model, but in general, the CA space can be of arbitrary size and orientation with respect to the FE domain, depending on what deformation and/or fracture phenomena are to be studied with it, as shown in Figure 3. Some FEs will occupy the same physical space as some CA cells. These FEs and cells form a two-way macro/micro multi-scale CAFE model. However, as indicated in Figure 3, in general, there will be cells occupying physical space outside of the body. Such cells do not participate in a multi-scale CAFE analysis.

The coarray/MPI CAFE framework is built with an assumption that at runtime there is always an identical number of MPI processes and coarray images, and that the first MPI process and the first image exist on the first processing element (PE), and so on.

A schematic partition of the CAFE model on 4 PEs is shown in Figure 4. The boxes show on which PE the corresponding parts of the model are stored. For example, "image 1" and "MPI 1" parts of the model are stored on PE 1. However, these FEs do not share physical space with these CA cells. Instead cells on image 1 share physical

Figure 4: Schematics of communications between the MPI (FE) and the coarray (CA) parts of the coarray/MPI (CAFE) hybrid model on 4 PEs

space with FEs on PE 3, labelled "MPI 3". This is important because information transfer is required only between CA and FE which occupy the same physical space. In this example the MPI part of the model stored on PE 3 will have to communicate with the coarrays stored on PEs 1 and 3.

Communications between the MPI (FE) and the coarray (CA) parts of the coarray/MPI (CAFE) hybrid model are shown schematically with arrows in Fig. 4. The imbalance in the communication pattern is clear. The FE part of the model stored on PE 4 will not communicate with CA at all. However, the FE part of the model stored on PE 1 will need to communicate with CA coarrays stored on PEs 2 and 4.

The mapping of FE to CA is established via a private allocatable array of derived type:

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

based on coordinates of FE centroids calculated by each MPI process (`lcentr` stands for local, i.e. non-coarray array of centroids). These coordinates are stored in a coarray of derived type with allocatable array component:

```
type rca
  real, allocatable :: r(:, :, :)
end type rca
```

```
type( rca ) :: centroid_tmp[*]
```

which is allocated as

```
allocate( centroid_tmp%r(3, nels_pp) )
```

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

There are two different routines which establish `lcentr` on each image from `centroid_tmp`. Subroutine `cgca_pfem_cenc` implements an all-to-all communication pattern, i.e. each image reads `centroid_tmp` from every image. Subroutine `cgca_pfem_map` uses temporary arrays and coarray collectives `CO_SUM` and `CO_MAX`. At the time of writing coarray collectives are supported by the Cray and GCC/OpenCoarrays compilers. The two routines differ in their use of remote communications. However, both routines implement the same algorithm for establishing `lcentr` - if the centroid of an FE on any image is within the CA coarray on this image, then this FE is added to `lcentr` on this image.

Figure 5: `lcentr` arrays on two images P and Q.

Fig. 5 schematically shows `lcentr` arrays established on two images P and Q. In this example finite element `n`, stored on image Q, has centroid coordinates `r`, which identify a physical location within the CA coarray on image P. So this element is stored in the `lcentr` array on image P. Finite element `m`, also stored on image Q, has centroid coordinates `u`, which identify a physical location within the CA coarray also on image Q. So this element is stored in the `lcentr` array on image Q. FEs with centroids outside of the CA space are not entered in `lcentr`. `lcentr` plays a key role in information transfer between the FE and the CA parts of the multi-scale CAFE model.

Figure 6: Schematics of the patch test with the quarter model highlighted (left) and the deformed FE mesh with the contour plot of vertical displacement (right)

4.4 CAFE modelling of brittle fracture in polycrystals

Diverse CAFE fracture models can be constructed from the CGPACK and the ParaFEM libraries. The simplest case, presented here, uses a combination of linear isotropic elastic FE with cleavage (fully brittle transgranular fracture mode) CA. Cleavage is the dominant low temperature fracture mode in body centre cubic (bcc) crystals, such as iron. Each time or strain increment of the FE solver the stress tensor, $\mathbf{t}$, is passed to the CA, where it is resolved into normal stresses on $\{100\}$ and $\{110\}$ crystal planes - t_{100}, t_{110} [6, 10]. The localisation (or scatter) algorithm distributes the FE quantities over CA cells based on existing damage in the microstructure, while preserving the FE energy [11].

The cleavage model includes 2 parameters - a fracture stress, σ_F , linked to the free surface energy, γ , and a characteristic length, L . If $t_{100} \geq \sigma_F$ or $t_{110} \geq \sigma_F$, then a CA cleavage crack is extended by L per unit of time. Crack morphology is reduced to a single damage variable, d , by the homogenisation (or gather) algorithm, and the Young's modulus of each FE integration point is reduced according to d , where $d = 1$ means no damage, and $d = 0$ means that the integration point has no load bearing capacity. To avoid numerical instability the FE stiffness is not reduced to below 10^{-3} of the original value (corresponding to $d = 10^{-3}$).

4.4.1 A patch test

Schematics of a 3D patch test and a quarter FE model are shown in Figure 6.

Prediction of scatter is one of the strongest features of a CAFE approach. Each run of the CAFE model a new random microstructure is simulated which leads to the possibility of a stochastic structural integrity analysis. Figure 7 shows two random realisations of a polycrystalline microstructure leading to two unique crack propagation histories, and hence to unique macro-crack topologies, shown in Figure 8, and associated work of fracture.

Figure 7: Two random CA realisations of microstructure, showing grain boundaries (purple) and crack traces (green).

Figure 8: Emerging macro-cracks through two random CA microstructures shown in Figure 7. Green cracks are on $\{110\}$ planes. Yellow cracks are on $\{100\}$ planes

4.4.2 A rod under tension

CAFE cleavage simulation in a rod under tension is shown in Figures 9 and 10. The FE model is a 140mm long mild steel cylinder of 10mm diameter and 100mm gauge length. One end of the cylinder is constrained and an axial force is applied to the other end. The FE elastic properties are the Young's modulus of 200GPa and the Poisson's ratio of 0.3. Details of the CA material block were given at the end of Sec. 4.2. The CA block is positioned centrally on the cylinder, see Figure 9.

Figure 9 shows the polycrystalline microstructure layer of the `space` coarray. The colour of each grain (single crystal) encodes its rotation tensor. It also shows the grain boundaries in the fracture layer of the `space` coarray.

Figure 10 shows the macro-crack emerging from linking cracks on preferential cleavage planes in individual crystals. There are 4 cell fracture states in this model:

Figure 9: CAFE modelling of a steel cylinder under tension showing the CA microstructure (left) and the CA microstructure grain boundaries, with inactive cells removed (right). The FE cylinder mesh is semi-transparent for clarity

-1, -2, -3 and -4. -1 (yellow) denotes crack flanks on 100 planes. -3 (light blue) denotes crack flanks on 110 planes. Both yellow and light blue regions are clearly visible in Fig. 10. -2 (dark blue) denotes crack edges on 100 planes. -4 (cyan) denotes crack edges on 110 planes. Also shown in Figure 10 is the FE mesh at the end of the simulation, when the macroscopic cleavage crack has propagated across nearly the whole of the cross section. The contour plot of the axial displacement is superimposed over the mesh. Note a high displacement gradient across the crack.

5 Fortran coarray/MPI CAFE performance on HPC

Individually both ParaFEM and CGPACK libraries showed the potential to scale well into tens of thousands of cores, as seen in Figs. 11 and 12. Note that since both ParaFEM and CGPACK are libraries, scaling analysis makes sense only in context of specific programs built with these libraries. A representative scaling of a CAFE multi-scale fracture simulation with 1M FE and 800M CA cells on Cray XC30, is shown in Fig. 12. The scaling limit is about 7,000 cores (300 Cray XC30 nodes).

5.1 CAFE IO

A typical volume of microstructure in a CAFE approach might include 10^6 grains or 10^{11} cells. With 4-byte integers to store cell states, each layer of `space` coarray

Figure 10: Micro-cracks merging into a macro-crack in a steel cylinder under tension (left) and the distorted FE mesh at the end of the CAFE simulation, with the axial displacement contours (right)

will take $\approx 373\text{GB}$, i.e. 745GB for both fracture and microstructure datasets. Multi-step CAFE analyses, e.g. progressive fracture propagation through microstructure, demand that `space` coarray is written to disk at regular intervals. It is clear that efficient coarray IO is required for good scaling.

The Fortran standard does not include parallel IO. However, approaches to achieving high IO performance in MPI programs can be readily applied to coarrays [33].

A single writer/single file serial model is easiest to implement, but has the lowest performance, about 100MB/s on the Cray XE6. In contrast, a multiple writers/single file parallel model has the highest performance using MPI/IO. With some tuning of the Lustre file system, in particular `lfs` stripe size and stripe count settings, rates of 2.3GB/s have been achieved. In all cases the `CA space` array coarray is written out as a binary dataset with no metadata. Knowledge of the array extents and of linear spatial resolution is required for post-processing.

NetCDF and HDF5 IO writers are beneficial to direct MPI/IO because the metadata is written either together with the data, as in the NetCDF case, or is encoded in a simple XDMF wrapper, as in the HDF5 case. Both NetCDF and HDF5 have been implemented in CGPACK. However, as Figure 13 shows, at present maximum NetCDF IO rates are only about 1.2GB/s [34], which is significantly lower than direct use of MPI/IO. Work is under way to implement MPI/IO in ParaFEM.

Figure 11: Strong scaling of ParaFEM (MPI) for a 3D transient flow explicit analysis on Cray XE6. Reproduced from [21]

Figure 12: Strong scaling of a CGPACK (coarrays) solidification model (left) and of a ParaFEM/CGPACK (MPI/coarrays) CAFE fracture model

5.2 Synchronising a coarray library

The CGPACK library consists of a number of modules and submodules, with serial and parallel subroutines. A variety of programs can be built, using as many or as few CGPACK library routines as required. The design of the library makes only very basic assumptions on the order of calls to CGPACK routines in a program, e.g. fracture routines must be called after routines establishing microstructure. An error condition is flagged if the order of these routines is reversed. In most other cases the logic of the order of the invocation of CGPACK routines is left to the user.

Figure 13: CGPACK IO rates showing the influence of the Lustre file system stripe size and stripe count settings

An immediate consequence of such design is that inter image synchronisation becomes a hard decision. The Fortran standard imposes very strict segment ordering rules to ensure data integrity and to prevent deadlocks [13].

Synchronisation requirements differ for each individual CGPACK routine. For example, a halo exchange algorithm logically maps best onto SYNC IMAGES image control statement. Assuming a 3D grid of images, $[:, :, :]$, see Sec. 4.2, each image has to synchronise only with its 26 neighbouring images, i.e. image with the coindex set $[a, b, c]$ has to synchronise with images from $[a-1, b-1, c-1]$ to $[a+1, b+1, c+1]$. However, the library writer has no way to predict what routines will precede or succeed the halo exchange routine. In practice this often means that the only safe image control statement is SYNC ALL, a global barrier. A fragment of a typical coarray CAFE program might look like this:

```

call cgca_nr( space ) ! sync all inside
call cgca_rt( grt )    ! sync all inside
call cgca_sld( space ) ! sync all inside
call cgca_igb( space )
sync all
call cgca_hxi( space )
sync all
call cgca_gbs( space )
sync all
call cgca_hxi( space )
sync all
sync all

```



```

call cgca_gcu( space ) ! local routine
                                ! no sync needed

```

Note that some CGPACK routines include image control statements in the beginning and/or the end, e.g. `cgca_sld`, the solidification routine and `cgca_nr`, the nucleation routine. It does not make sense to start `cgca_sld` on any image until `cgca_nr` has finished on all images. In such cases the responsibility for arranging sufficient synchronisation has been taken away from the end user. However, in other cases, the user is likely to deploy `SYNC ALL` to be safe, as shown above.

While excessive use of `SYNC ALL` might lead to over synchronisation, and hence to poor scaling, our prior profiling analysis on Cray XC30 concluded that the current ParaFEM/CGPACK (MPI/coarrays) scaling limit of 7,000 cores, see Figure 12, is not related to this [32].

ParaFEM synchronisation properties are very different, because most of its routines use 2-way message passing MPI calls. In this regard it is very fortunate that in a ParaFEM/CGPACK CAFE program the calls to each library do not alternate often, - there is typically a large chunk of code made of ParaFEM calls, then `SYNC ALL`, then a large chunk of code made of CGPACK calls, etc. A fragment of a CAFE fracture program is shown below.

```

call cgca_pfem_salloc( nels_pp , nip , nst )
sync all
!end CGPACK part
!start ParaFEM part
CALL rearrange( rest )
elements_0: DO iel=1, nels_pp
  CALL find_g3( g_num_pp(:, iel), &
                g_g_pp(:, iel), rest )
END DO elements_0

```

5.3 Opportunities for thread parallelisation

Many CA routines contain triple nested loops over all cells on an image. An example below is taken from `cgca_clvgn`, the cleavage propagation routine. Each iteration of the main loop all cells in the CA on an image are processed.

```

main: do iter = 1,N
  do x3 = lbr(3), ubr(3)
  do x2 = lbr(2), ubr(2)
  do x1 = lbr(1), ubr(1)
    live: if ...
      ! scan only through undamaged cells
      call cgca_clvgn( clvgflag )
      if ( clvgflag ) call sub( space )
    end if live
  end do x1
  end do x2
  end do x3

```

```

end do
end do
end do
call co_sum( clvgglob )
sync all
call cgca_hxi( space )
sync all
call cgca_dacf( space )

```

Such nested loops might present good opportunities for thread parallelisation with either OpenMP or OpenACC (e.g. on GPUs or Xeon Phi), although the use of underpopulated nodes might be required. Fortran 2008 new intrinsic `DO CONCURRENT` should also be explored, although at present its performance portability is inferior to OpenMP. Recently, ParaFEM has been ported to Xeon Phi [35]. In order to make best use of the Xeon Phi architecture, the code needed some rewriting to use a mixed OpenMP/MPI parallelisation strategy. On standard x86 multicore processors, the addition of OpenMP provides no benefit. However, on the Xeon Phi, OpenMP using 4 threads per core provides an additional 4-fold speed-up in run times. Porting of CGPACK to Xeon Phi is planned for the future.

6 Conclusions

Coarrays are a new exciting feature of standard Fortran. It was shown in this work that coarrays are a natural data representation model for 3D cellular automata framework. The use of an integer allocatable array coarray with 4 dimensions and 3 codimensions in CGPACK for a 3D CA polycrystalline microstructure simulation was successful. It was shown that solidification and fracture of a polycrystalline microstructure can be modelled efficiently on HPC systems. Solidification models can scale to 32k cores on Cray systems. A multi-scale continuum/microstructure model was constructed by linking together cellular automata coarray library CGPACK with MPI finite element library ParaFEM. Considerable attention has been given to establishing a robust FE to CA mapping data structures and procedures, resulting in a concurrent hierarchical two-way multi-scale CAFE model. Coarrays of derived type with allocatable components were found to be very useful for maintaining dynamic data structures which link the MPI and the coarray parts of the framework. A cleavage fracture of a cylindrical ferritic steel specimen was shown as a simple CAFE application, which scaled up to 7k cores on Cray XC30. A diverse range of other CAFE programs can be created by using ParaFEM with CGPACK. This work proves that interfacing MPI and coarrays is easily achievable. This opens many possibilities for applications in other areas of science and engineering. In addition, because both ParaFEM and CGPACK are distributed under BSD license, the two libraries can be used by researchers from other fields, e.g. biomechanics for study of bone growth, fracture and regeneration.

Acknowledgements

This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (<http://www.archer.ac.uk>). This work was carried out using the computational facilities of the Advanced Computing Research Centre, The University of Bristol, UK (<http://www.bris.ac.uk/acrc>). We would like to acknowledge the assistance of the Software Sustainability Institute, UK (<https://www.software.ac.uk>).

References

- [1] W.A. Curtin, R.E. Miller, “Atomistic/continuum coupling in computational materials science”, *Model. Simul. Mater. Sci. Eng.*, 11: R33–R68, 2003.
- [2] M. Xu, T. Belytschko, “Conservation properties of the bridging domain method for coupled molecular/ continuum dynamics”, *Int. J. Numer. Meth. Eng.*, 76: 278–294, 2008.
- [3] M. Wallin, W.A. Curtin, M. Ristinmaa, A. Needleman, “Multi-scale plasticity modeling: Coupled discrete dislocation and continuum crystal plasticity”, *J. Mech. Phys. Solids*, 56: 3167–3180, 2008.
- [4] G. Guillemot, C.A. Gandin, M. Bellet, “Interaction between single grain solidification and macro segregation: Application of a cellular automaton - Finite element model”, *J. Crystal Growth*, 303: 58–68, 2007.
- [5] C. Zheng, D. Raabe, “Interaction between recrystallization and phase transformation during intercritical annealing in a cold-rolled dual-phase steel: A cellular automaton model”, *Acta Materialia*, 61: 5504–5517, 2013.
- [6] A. Shterenlikht, L. Margetts, “Three-dimensional cellular automata modelling of cleavage propagation across crystal boundaries in polycrystalline micro- structures”, *Proc. Roy. Soc. A*, 471: 20150039, 2015.
- [7] A. Shterenlikht, I.C. Howard, “The CAFE model of fracture – application to a TMCR steel”, *Fatigue Fract. Eng. Mater. Struct.*, 29: 770–787, 2006.
- [8] S. Das, A. Shterenlikht, I.C. Howard, E.J. Palmiere, “A general method for coupling microstructural response with structural performance”, *Proc. Roy. Soc. A*, 462: 2085–2096, 2006.
- [9] S.J. Wu, C.L. Davis, A. Shterenlikht, I.C. Howard, “Modeling the ductile-brittle transition behavior in Thermomechanically Controlled Rolled Steels”, *Met. Mater. Trans. A*, 36: 989–997, 2005.

- [10] A. Shterenlikht, S. Margetts, L. McDonald, N.K. Bourne, “Towards mechanism-based simulation of impact damage using exascale computing”, *AIP Conference Proceedings*, 1793: 080009, 2017.
- [11] V. Kouznetsova, W.A.M. Brekelmans, F.P.T. Baaijens, “An approach to micro-macro modeling of heterogeneous materials”, *Comp. Mech.*, 27: 37–48, 2001.
- [12] M. Metcalf, J. Reid, M. Cohen, *Modern Fortran Explained*, Oxford, 2011.
- [13] ISO/IEC 1539-1:2010, *Fortran – Part 1: Base language, International Standard*, 2010.
- [14] G. Mozdzyński, M. Hamrud, N. Wedi, “A Partitioned Global Address Space implementation of the European Centre for Medium Range Weather Forecasts Integrated Forecasting System”, *Int. J. High Perf. Comp. Appl.*, 29: 261–273, 2015.
- [15] R. Preissl, N. Wichmann, B. Long, J. Shalf, S. Ethier, A. Koniges, “Multi-threaded Address Space Communication Techniques for Gyrokinetic Fusion Applications on Ultra-Scale Platforms”, in *Supercomputing 2011, USA*, 2011.
- [16] A. Shterenlikht, L. Margetts, L. Cebamanos, D. Henty, “Fortran 2008 coarrays”, *ACM Fortran Forum*, 34: 10–30, 2015.
- [17] J. Phillips, A. Shterenlikht, M.J. Pavier, “Cellular automata modelling of nanocrystalline instability”, in *Proc. 20th UK ACME Conf., Manchester, UK*, 2012.
- [18] A. Shterenlikht, “Fortran coarray library for 3D cellular automata microstructure simulation”, in *Proc. 7th PGAS Conf., Edinburgh, UK*, pages 16–24, 2014.
- [19] L. Margetts, *Parallel Finite Element Analysis*, PhD thesis, University of Manchester, 2002.
- [20] I.M. Smith, *Programming the Finite Element Method*, Wiley, 1982.
- [21] I.M. Smith, D.V. Griffiths, L. Margetts, *Programming the Finite Element Method*, Wiley, 5ed, 2014.
- [22] I.M. Smith, L. Margetts, “The convergence variability of parallel iterative solvers”, *Eng. Computations*, 23: 154–165, 2006.
- [23] I.M. Smith, L. Margetts, “Portable parallel processing for nonlinear problems”, in *Proc. VII Int. Conf. Computational Plasticity, Barcelona, Spain*, 2003.
- [24] L. Margetts, I.M. Smith, J.M. Leng, “Parallel 3D finite element analysis of coupled problems”, in *Proc. III European Conf. Comp. Mechanics in Solids, Structures and Coupled Problems in Engineering, Lisbon, Portugal*, 2006.

- [25] L.M. Evans, L. Margetts, V. Casalegno, L.M. Lever, J. Bushell, T. Lowe, A. Wallwork, P. Young, A. Lindemann, M. Schmidt, P.M. Mummery, “Transient thermal finite element analysis of CFC-Cu ITER monoblock using X-ray tomography data”, *Fusion Eng. Des.*, 100: 100–111, 2015.
- [26] J.D. Arregui-Mena, L. Margetts, D.V. Griffiths, L. Lever, G. Hall, P.M. Mummery, “Spatial variability in the coefficient of thermal expansion induces pre-service stresses in computer models of virgin Gilsocarbon bricks”, *J. Nuclear Mater.*, 465: 793–804, 2015.
- [27] F. Levrero-Florencio, L. Margetts, E. Sales, S. Xie, K. Manda, P. Pankaj, “Evaluating the macroscopic yield behaviour of trabecular bone using a nonlinear homogenisation approach”, *J. Mech. Behavior Biomed. Mater.*, 61: 384–96, 2016.
- [28] S.D. Rawson, L. Margetts, J.K.F. Wong, S.H. Cartmell, “Sutured tendon repair; a multi-scale finite element model”, *Biomech. Model. Mechanobiol.*, 14: 123–133, 2015.
- [29] L. Margetts, I.M. Smith, L.M. Lever, D.V. Griffiths, “Parallel processing of excavation in soils with randomly generated material properties”, in *Proc. 8th European Conf. Numer. Methods in Geotechnical Engineering, Delft, Netherlands*, pages 265–270, 2014.
- [30] L. Margetts, J.M. Leng, I.M. Smith, P.L. Manning, “Parallel three dimensional analysis of dinosaur trackway formation”, in *Proc. 6th European Conf. Numer. Methods in Geotechnical Engineering, Graz, Austria*, pages 743–749, 2006.
- [31] S. Crouch, N.C. Hong, S. Hettrick, M. Jackson, A. Pawlik, S. Sufi, L. Carr, D.D. Roure, C. Goble, M. Parsons, “The Software Sustainability Institute: Changing Research Software Attitudes and Practices”, *Computing in Science & Engineering*, 15: 74–80, 2013.
- [32] L. Cebamanos, A. Shterenlikht, D. Arregui-Mena, L. Margetts, “Scaling hybrid coarray/MPI miniapps on Archer”, in *Cray User Group 2016 meeting (CUG2016), London, 8-12-MAY-2016*, 2016.
- [33] D. Henty, A. Jackson, C. Moulinec, V. Szeremi, “Performance of Parallel IO on ARCHER, version 1.1”, 2015, <http://archer.ac.uk>.
- [34] A. Shterenlikht, L. Margetts, L. Cebamanos, J.D. Arregui-Mena, “Multi-scale CAFE framework for simulating fracture in heterogeneous materials implemented in Fortran coarrays and MPI”, in *PGAS Application Workshop (PAW), Supercomputing 2016, USA*, 2016.
- [35] L. Margetts, J.D. Arregui Mena, T. Hewitt, L. Mason, “Parallel finite element analysis using the Intel Xeon Phi”, in *Proc. Emerging Technology Conf. (EMiT 2016), ISBN 978-0-9933426-3-9*, 2016.