Fortran coarray/MPI Multi-Scale CAFE for Fracture in Heterogeneous Materials

A. Shterenlikht\textsuperscript{1}, L. Margetts\textsuperscript{2}, L. Cebamanos\textsuperscript{3}

\textsuperscript{1}Mech Eng Dept, The University of Bristol, Bristol BS8 1TR, UK, mexas@bris.ac.uk

\textsuperscript{2}School of Mechanical, Aero and Civil Engineering, The University of Manchester, Manchester M13 9PL, UK, Lee.Margetts@manchester.ac.uk

\textsuperscript{3}Edinburgh Parallel Computing Centre (EPCC), The University of Edinburgh, King’s Buildings, Edinburgh EH9 3FD, UK, l.cebamanos@epcc.ed.ac.uk

PARENG 2017, Pécs, Hungary 30th-31st May, 2017
Fracture in heterogeneous materials

- polycrystal cleavage
- reinforced concrete
- metal matrix
- bone
- graphite

- All real materials are heterogeneous
- Multiple fracture and damage processes happen at different time and length scales → need multi-scale framework
Fracture: $\text{CA} + \text{FE} = \text{CAFE}$ multi-scale model

- **Structured grids** - cellular automata (CA), unstructured grids - finite elements (FE)
- **CA** (microstructure) + **FE** (continuum mechanics) = CAFE
- Transgranular cleavage - fracture stress or strain criteria
- **FE** → **CA** (localisation) - stress, strain fields
- **CA** → **FE** (homogenisation) - damage variables
Fortran coarrays for CA

- Fortran native SPMD parallel programming feature
- Cray, Intel, OpenCoarrays/GCC support
- CGPACK - cellular automata microstructure simulation library: [cgpack.sf.net](http://cgpack.sf.net). See also [1, 2, 3, 4].
- Easy halo exchange
- CA space coarray - 4D array, 3 codimensions:
  
  ```fortran
  integer, allocatable :: space (:, :, :, :) [::, ::, ::]
  ```

- Ideal for structured grids:

![18 imgs; 64 imgs](image1.png)
Coarray IO - no native Fortran parallel IO

- MPI/IO up to 2.3GB/s on Cray XE6
- MPI/IO up to 8GB/s on Cray XC30 (can reach 14GB/s [5])
- NetCDF 4.3, HDF5 1.8.14 - only up to 1.2GB/s on Cray XC30.
- Lfs stripe count, size, number of images, file size, Cray hugepages...
- 0.5 - 1TB datasets
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 [6]
- Excellent scaling
- BSD license
CAFE design: structured CA grid + unstructured FE grid

Example with 4 PE (4 MPI processes, 4 coarray images). Arrows are FE ↔ CA comms.
lcentr arrays on images P and Q

PE, image, MPI process P

lcentr

image
elnum
centr

... Q ... P ...
... n ... b ...
... r ... s ...

PE, image, MPI process Q

lcentr

image
elnum
centr

... Q ... P ...
... m ... a ...
... u ... t ...
Fracture modelling

- Diverse CAFE fracture models can be constructed from CGPACK + ParaFEM libraries.
- Simple case: isotropic linear elastic FE \((E, \nu)\) + cleavage (fully brittle transgranular fracture mode) CA.
- FE stress tensor \(t\) passed to CA, resolved on normal stresses on \(\{100\}\) and \(\{110\}\) crystal planes - \(t_{100}, t_{110}\) [2, 7].
- 2 parameters - fracture stress, \(\sigma_F\), linked to the free surface energy, \(\gamma\), and a characteristic length, \(L\).
- If \(t_{100} \geq \sigma_F\) or \(t_{110} \geq \sigma_F\) then a CA crack extends by \(L\) per unit of time.
- Crack morphology is reduced to a single damage variable, \(d\). \(d = 1\) initially (no damage). \(d = 0\) - integration point has failed, no load bearing capacity.
Cleavage in a steel cylinder under tension
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.
CA - coarray (over)synchronisation?

```fortran
  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
  call cgca_gcu( space ) ! local routine no sync
```

- All images sync with their 26 neighbours.
- Some routines have sync inside.
- Other sync responsibility is left to end user.
use, intrinsic iso_fortran_env, only: event_type

type(event_type) :: var[:,:,:,:]
integer, allocatable :: space(:,:,:,:,:)[:,:,:,:,:]
integer :: errstat, myrank(3)

allocate var, space
myrank = this_image(space)

! do some work, then notify neighbours

event post(
    &
    var[myrank(1)-1, myrank(2), myrank(3)], &
    stat=errstat)

! 25 more posts
:event wait(var, until_count=26, stat=errstat)

! when all 26 neighbours posted, continue work:
Future: thread level parallelism: OpenMP, DO CONCURRENT

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 ...
      call cgca_clvgn( clvgflag )
      if ( clvgflag ) call sub( space )
    end if live
  end do
end do
end do
end do

call co_sum( clvgglob )
sync all

call cgca_hxi( space )
sync all

call cgca_dacf( space )
Conclusions

- Fortran coarrays are an ideal match for cellular automata
- Hybrid coarray+MPI multi-scale fracture framework is feasible
- Scaling up to 7k cores currently, work ongoing
- Profiling/tracing tools: CrayPAT, TAU, Score-P, Scalasca - coarray support is improving
- Coarray synchronisation - major issue: data integrity & performance
Acknowledgements

- This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (http://www.archer.ac.uk), [archer.ac.uk](http://www.archer.ac.uk).
- This work was carried out using the computational facilities of the Advanced Computing Research Centre, University of Bristol - https://www.acrc.bris.ac.uk, [www.acrc.bris.ac.uk](http://www.acrc.bris.ac.uk).
Cellular automata (CA) basics

- discrete space, discrete time, discrete states - fully digital framework, **structured grids**
- finite or infinite space
- finite space: fixed or self-similar boundaries
- cell neighbourhood, e.g. von Neumann’s:

![Von Neumann's neighbourhood](image)

or Moore’s:

![Moore's neighbourhood](image)

- iterative process
- state of a cells at next iteration is a function of the state of this cells and of the states of its neighbourhood cells at the current iteration
Primitive 3D solidification - probabilistic CA

- States: liquid = 0, crystals > 0.
- Cell state uniquely encodes crystal orientation tensor, i.e. a look-up table.
- Each iteration a liquid cell acquires a state of a randomly chosen neighbour (3D Moore’s neighbourhood - 26 cells).

```
0 0 0 0 1 1 1 1
0 0 0 1 1 1 1 1
0 0 1 1 1 1 1 1
0 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
```

```
i
```

```
```

```
i + 1
```

```
0 0 1 1 1 1 1 1
0 0 1 1 1 1 1 1
0 0 1 1 1 1 1 1
0 0 1 1 1 1 1 1
0 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1
0 1 1 1 1 1 1 1
```
Primitive probabilistic 3D solidification - results

For more results ➤ CGPACK
Recrystallisation

- The grain boundary velocity \( \dot{x} = nmp \), \( n \) - the normal to the grain boundary segment, \( m \) - mobility, \( p \) - the driving force.

- If \( \dot{x} \Delta t \geq c \), where \( \Delta t \) - time increment, \( c \) - cell size, then a cell joins the growing grain.

- Mobility strongly depends on temperature:
  \( m \approx \alpha \exp(-\beta/T) \), \( \alpha, \beta \) - some parameters, \( T \) - temperature.

Dierk Raabe site
Other CA examples

- Sand pile formation
- Ising magnetisation
- Land use
- Diffusion
- Fire

More CAFE examples

- Used for solidification [9], recrystallisation [10] and fracture [11, 12].

- FE - continuum mechanics - stress, strain, etc.

- CA - crystals, crystal boundaries, cleavage, grain boundary fracture

- FE → CA - stress, strain

- CA → FE - damage variables
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>
<thead>
<tr>
<th>Table 1: Profile by Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samp%</td>
</tr>
<tr>
<td>100.0%</td>
</tr>
<tr>
<td>43.6%</td>
</tr>
</tbody>
</table>

| 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 |
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.


