Coarrays
From laptops to Supercomputers

Harvey Richardson
Cray UK Ltd

+ other contributors

BCS FSG Meeting Oct 1, 2015

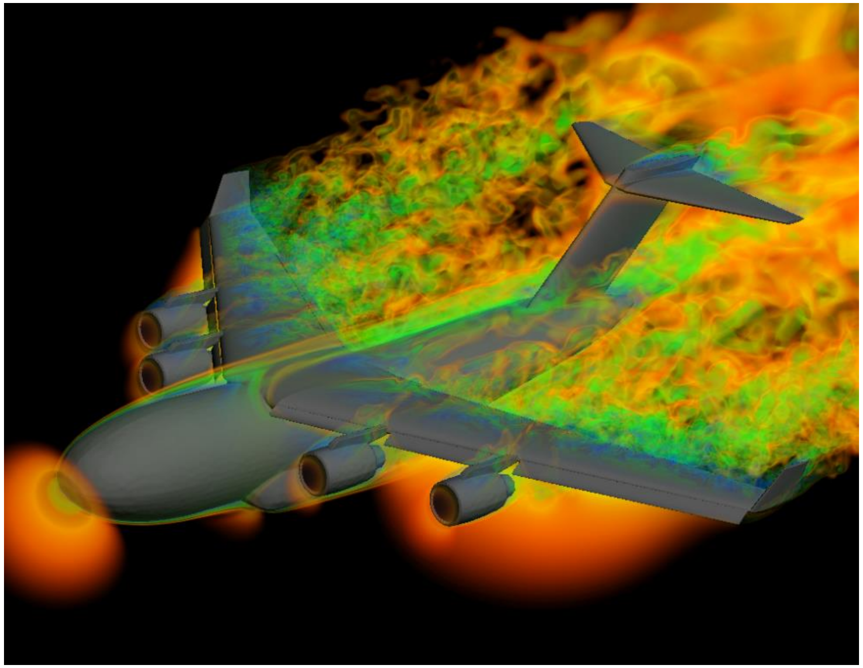
Agenda

- **Parallel programming models and High Performance Computing, where do coarrays fit?**
- **Coarray Implementations**
  - Open Source
  - Intel
  - Cray
- **Examples of coarray usage at scale:**
  - Gyrokinetic Fusion Code
  - Microstructure Simulation
  - Weather Forecasting code, ECMWF

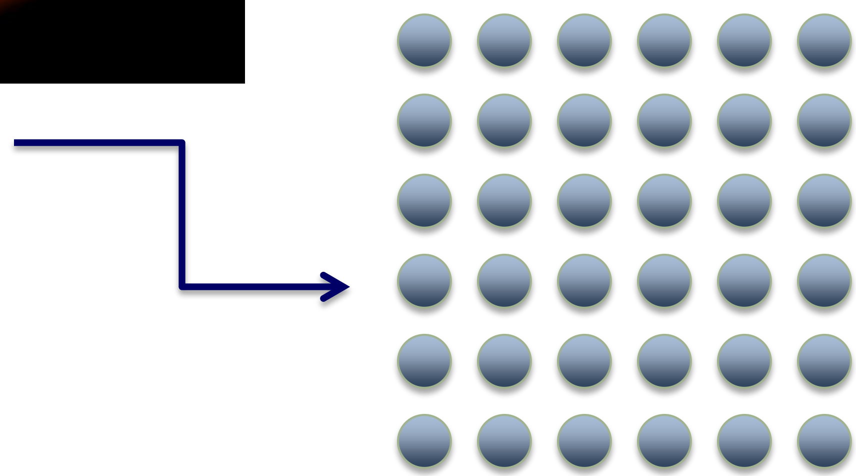
Parallel Programming Models

- More hardware has been a good way to run faster
- We started by marking up code for vectorization for special vector hardware
- Then SMPs came along (more than one cpu)
- For a directive-based approach OpenMP became pervasive as a way to mark up code that could utilize multiple threads (or cpus)
- Going beyond one process (or node) was a new challenge
- Message-passing was used for distributed architectures. Today this means MPI
- A Hybrid approach uses both at the same time
- PGAS approaches differentiate between local and remote access. One example is Fortran coarrays which is a language feature (as opposed to an API).

Cooperating Processes Models

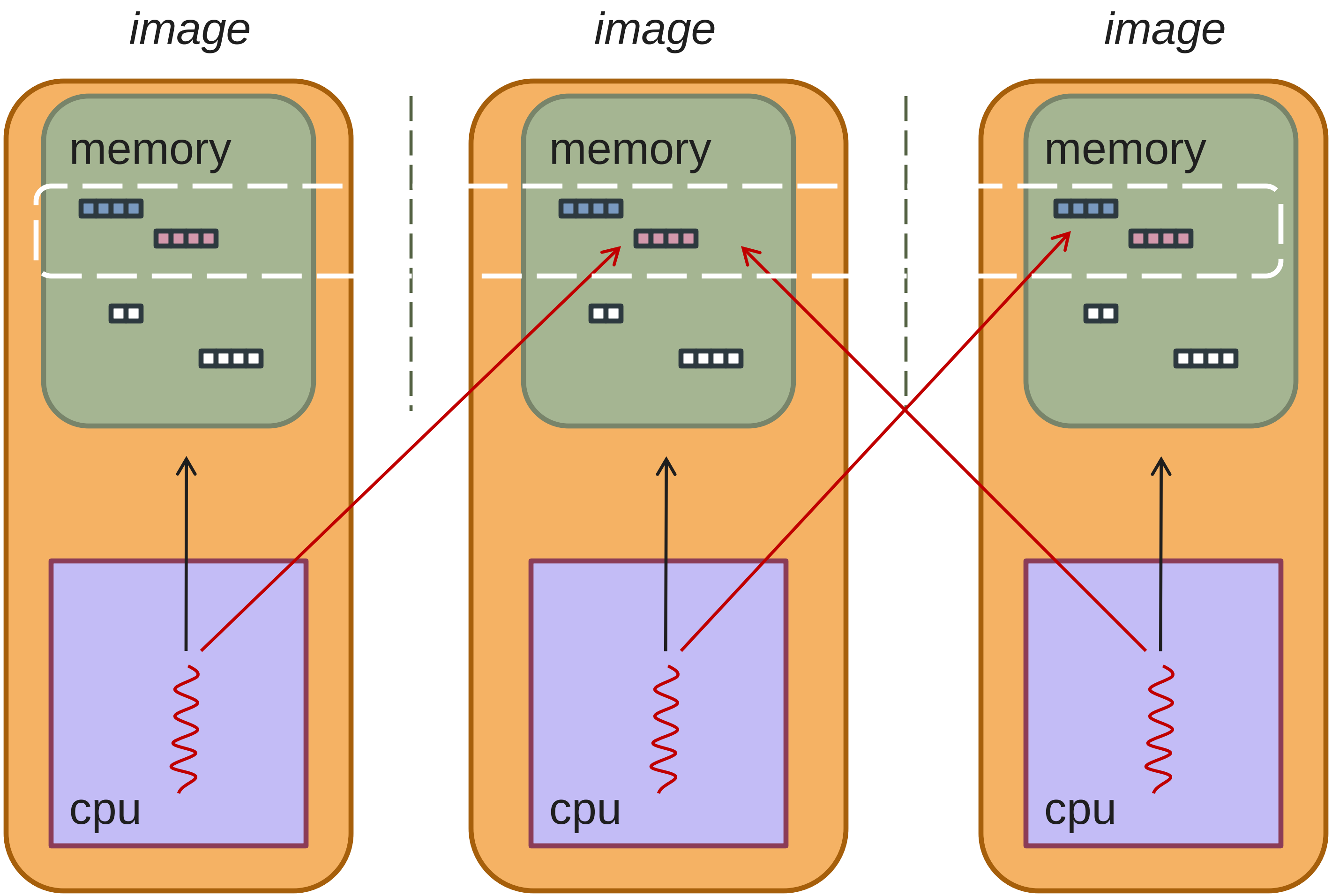


PROBLEM

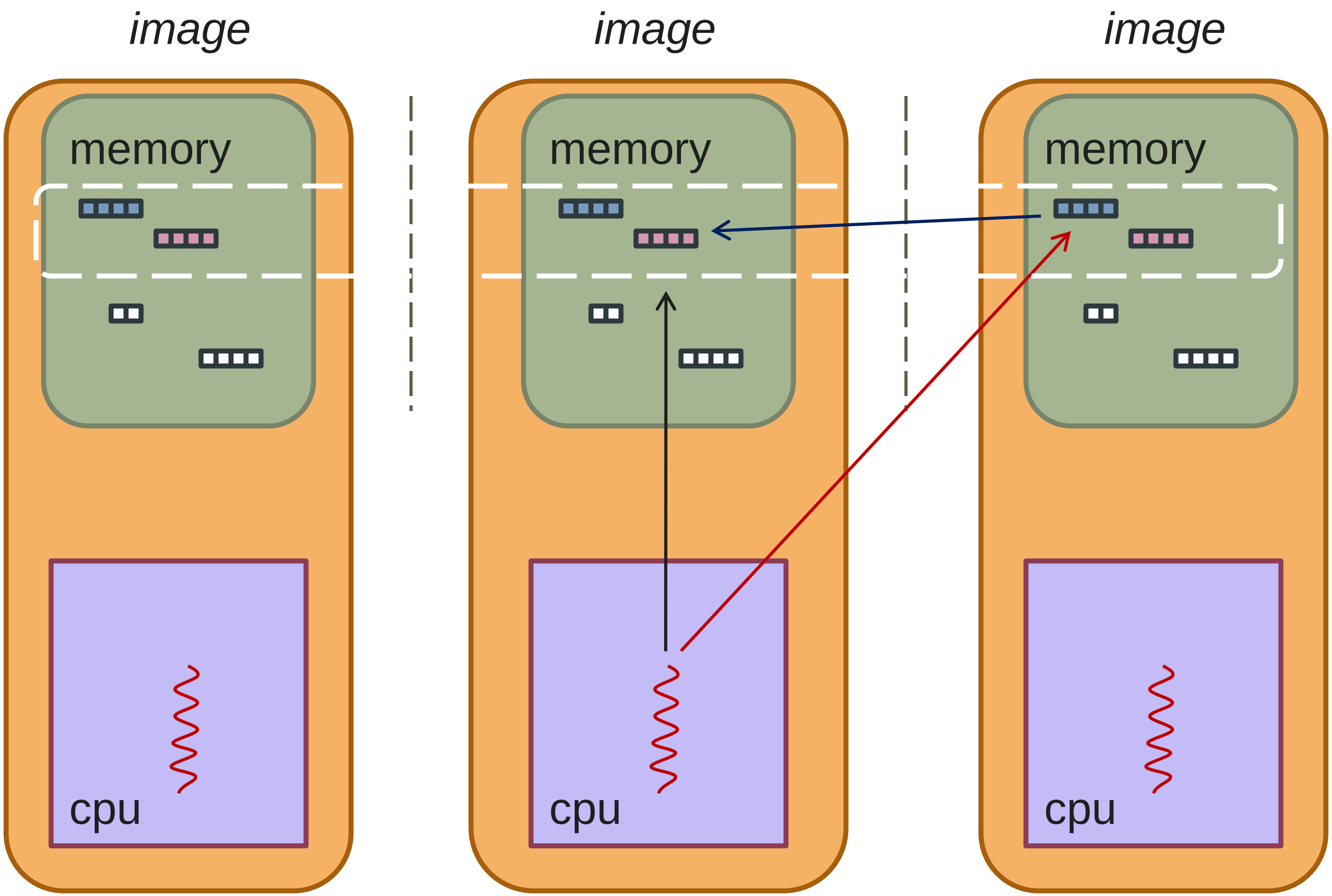


processes

Fortran coarray model



Fortran coarray model




```
a = b[3]
```

Implementations: An old slide from 2010

- **History of coarrays dates back to Cray implementations**
- **Expect support from vendors as part of Fortran 2008**
- **G95 had multi-image support in 2010**
  - was promising at the time
  - has not been updated for some time
- **gfortran**
  - Introduced single-image support at version 4.6
- **Intel: multi-process coarray support in Intel Composer XE 2011**
  (based on Fortran 2008 draft)
- **Runtimes are SMP, GASNet and compiler/vendor runtimes**
  - GASNet has support for multiple environments (IB, Myrinet, MPI, UDP and Cray/IBM systems) so could be an option for new implementations

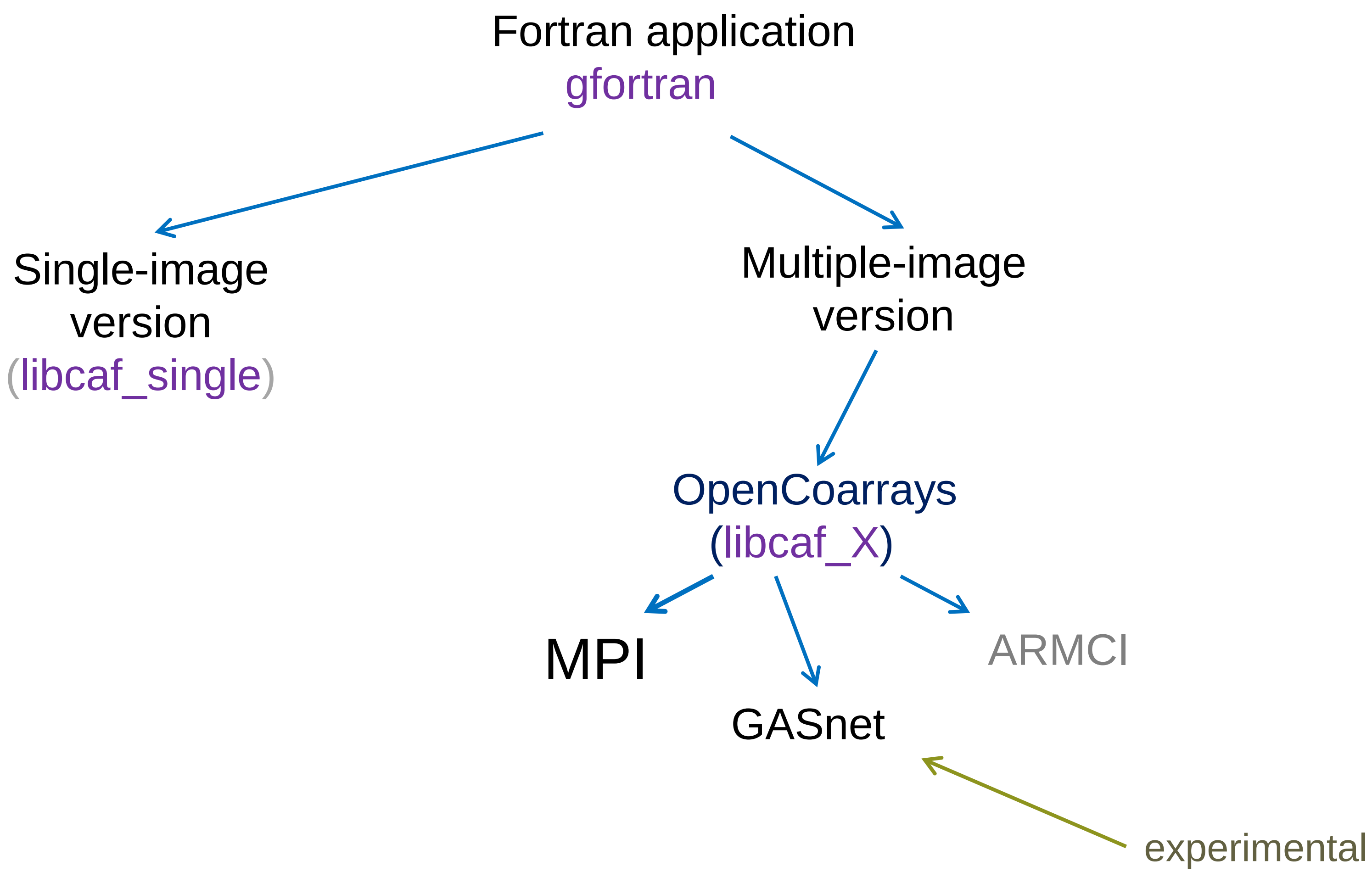
Open Source software stack for coarrays

- gfortran (from v5) supports Fortran 2008 coarrays along with broadcast/reduction collectives and atomics in TS18508
- Still some issues
- Single-image support (-fcoarray=single)
- Multi-image support via library (OpenCoarrays)

OpenCoarrays

- Provides a runtime to support coarrays
- Use –fcoarray=lib
- Implemented using:
  - MPI
  - GASnet
  - ARMCI (from Global Arrays)

OSS Stack for coarrays



So how can I use this myself?

Download binaries?

- I could not get that to work

Build it all yourself

- Build the sources on Linux
- Your distro (or the repository you use) may not provide elements that are new enough
- In my case (for Ubuntu 14.04 VM)
  I had to **build**/**install**:
  m4, **g++**, **gcc** (c,c++,gfortran), **MPICH**, **Cmake**, **OpenCoarrays**
- See backup slides for more details

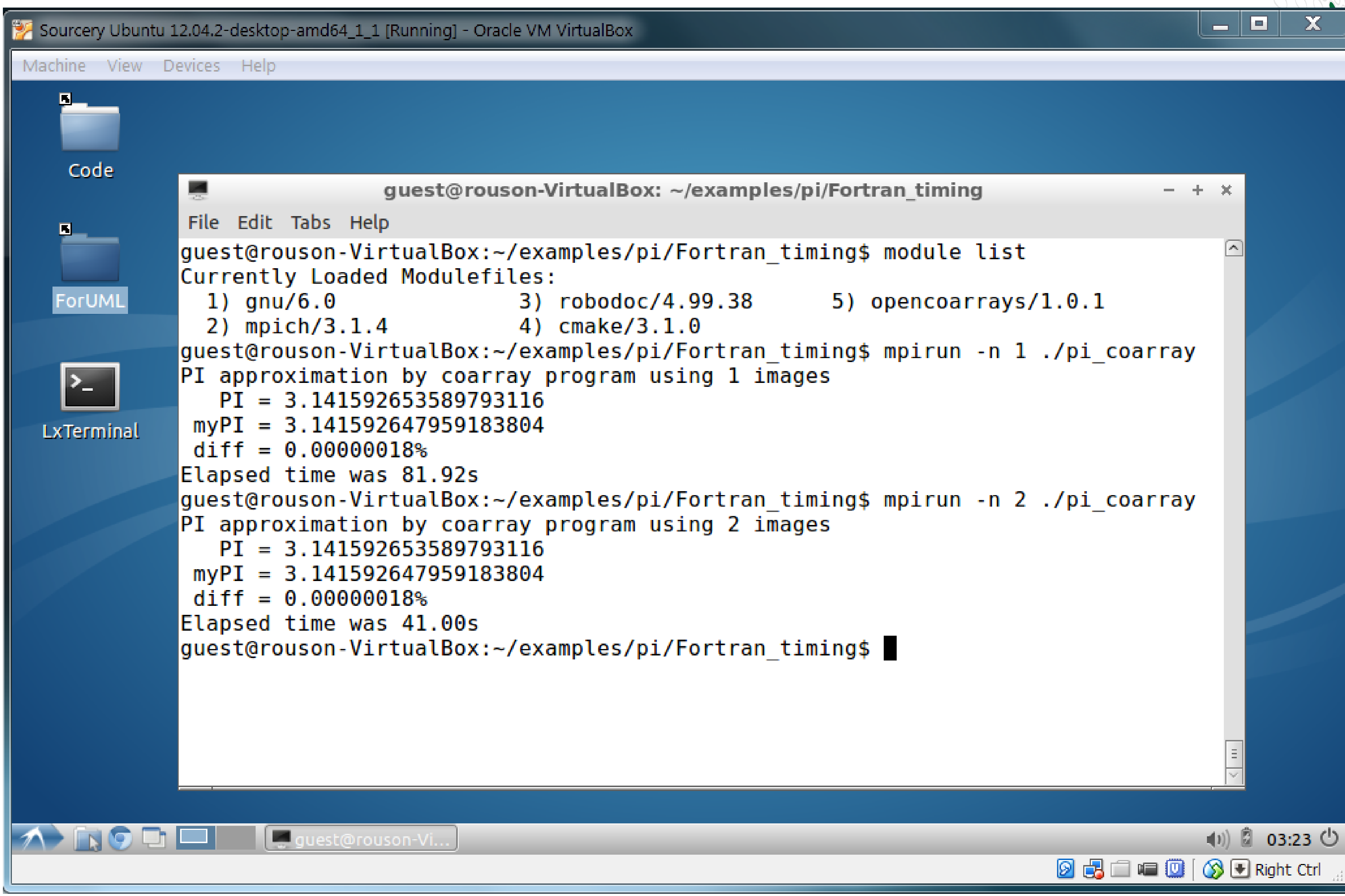
Use the Sourcery Institute VM

- Get from http://www.sourceryinstitute.org/
- This is a (4.3GB) pre-built VM appliance for VirtualBox

OpenCoarrays inside Ubuntu 14.04 VM

```
harvey@u14-04: ~/examples/pi/Fortran_timing
File Edit View Search Terminal Help
harvey@u14-04:~/examples/pi/Fortran_timing$ make -f Makefile.Gnu pi_coarray
caf -o pi_coarray pi_coarray.f90 params.o -O2 -I ../timers/Fortran ../timers/For
tran/timer.a
harvey@u14-04:~/examples/pi/Fortran_timing$ mpirun -n 1 ./pi_coarray
PI approximation by coarray program using 1 images
   PI = 3.141592653589793116
 myPI = 3.141592647959183804
 diff = 0.00000018%
Elapsed time was 80.92s
harvey@u14-04:~/examples/pi/Fortran_timing$ mpirun -n 2 ./pi_coarray
PI approximation by coarray program using 2 images
   PI = 3.141592653589793116
 myPI = 3.141592647959183804
 diff = 0.00000018%
Elapsed time was 40.74s
harvey@u14-04:~/examples/pi/Fortran_timing$ mpirun -n 4 ./pi_coarray
PI approximation by coarray program using 4 images
   PI = 3.141592653589793116
 myPI = 3.141592647959183804
 diff = 0.00000018%
Elapsed time was 21.02s
harvey@u14-04:~/examples/pi/Fortran_timing$ _
```

Sourcery Institute VM



Intel® Fortran Compiler

- Coarray support since Intel Composer XE 2011 (v12.0)
- Functionally complete in v13.0 a year later
- This used a distributed runtime based on Intel MPI
- Single-node support built-in (also MPI)
- Distributed (cluster) runtime needs the Cluster version of the current Parallel Studio product (licensing requirement)

- Intel are currently prioritising additional features

Intel Composer XE in Ubuntu VM

Applications Places System

harvey@u10-vbox: ~/fortran

File Edit View Search Terminal Help

```
harvey@u10-vbox:~/fortran$ ifort -coarray -O3 -o pdensity pdensity.f90
harvey@u10-vbox:~/fortran$ export FOR_COARRAY_NUM_IMAGES=1
harvey@u10-vbox:~/fortran$ ./pdensity
 Calculating prime density on           1 images
      539778 primes in     8000000 numbers
 density is  6.75%
 asymptotic theory gives  6.71%
Done in 15.61 seconds
harvey@u10-vbox:~/fortran$ export FOR_COARRAY_NUM_IMAGES=2
harvey@u10-vbox:~/fortran$ ./pdensity
 Calculating prime density on           2 images
      539778 primes in     8000000 numbers
 density is  6.75%
 asymptotic theory gives  6.71%
Done in 10.31 seconds
harvey@u10-vbox:~/fortran$ _
```

Cray Compilation Environment (CCE) Fortran

- **Cray has supported coarrays and UPC on various architectures for nearly two decades (from T3E)**
- **Full PGAS support on the Cray XT/XE/XC**
- **CCE Fortran Compiler**
  - ANSI/ISO Fortran 2008 compliant (since CCE 8.1 in 2012)
  - OpenMP 4.0, OpenACC 2.0
  - Coarray support integrated into the compiler
  - CCE 8.3/8.4 support TS29113 (interoperability) + collectives and atomics from new parallel features (spelling to be changed soon)
- **Fully integrated with the Cray software stack**
  - Same compiler drivers, job launch tools, libraries
  - Integrated with Craypat – Cray performance tools
  - Can mix MPI and coarrays

History of Cray PGAS runtimes

- **Cray X1/X2**
  - Hardware supports communication by direct load/store
  - Very efficient with low overhead
- **Cray XT**
  - PGAS (UPC,CAF) layered on GASNet/portals (so messaging)
  - Not that efficient
- **Cray XE/XC**
  - PGAS layered on DMAPP portable layer over Gemini/Aries network hardware
  - Aries supports RDMA, atomic operations and has hardware support for barrier/reduction operations
  - Intermediate efficiency between XT and X1/X2

The hardware/software stack has performance implications!

Examples of coarray usage at scale…

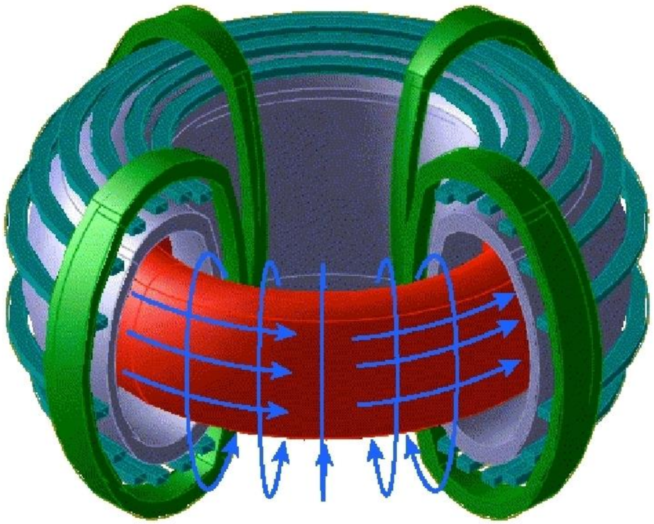
- Gyrokinetic Fusion Code
- Microstructure Simulation
- Weather Forecasting code, ECMWF

Gyrokinetic Fusion Code

- Tokamak fusion code for transport of charged particles
- Involved were: Robert Preissl, Nathan Wichmann, Bill Long, John Shalf, Stephane Ethier, Alice Koniges from Lawrence Berkeley National Lab (LBNL), Cray Inc and Princeton Plasma Physics Laboratory (PPPL)
- Optimized Hybrid MPI/OpenMP kernels replace with PGAS (coarray)/OpenMP

Gyrokinetic Fusion Code

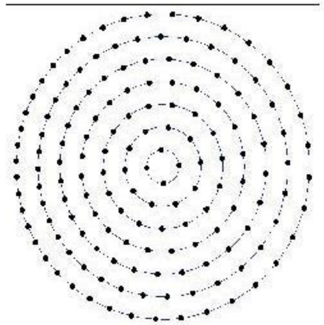
Tokamak geometry



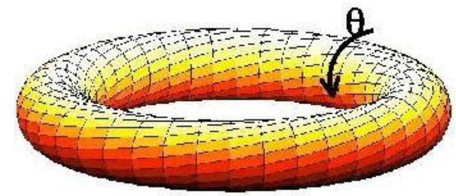
- Particle in Cell (PIC) approach to simulate motion of confined particles
- Motion caused by electromagnetic force on particle

Gyrokinetic Fusion Code

GTC full torus mesh



Points on poloidal plane



Computational domain with poloidal plane and field line following grid points

- Many particles stay in a cell for small time step but some don’t
- Timestep chosen to limit travel to 4 cells away
- Departing particles stored in a buffer and when this is full the data is sent to the neighboring cell’s incoming buffer
- Force fields recomputed once particles are redistributed
- Coarrays used to avoid coordinating the receive of the data: >40% improvement at 27,160 processes/images
- SC11 paper

Cleavage propagation across crystal boundaries

Anton Shterenlikht,
Department of Mechanical Engineering,
University of Bristol

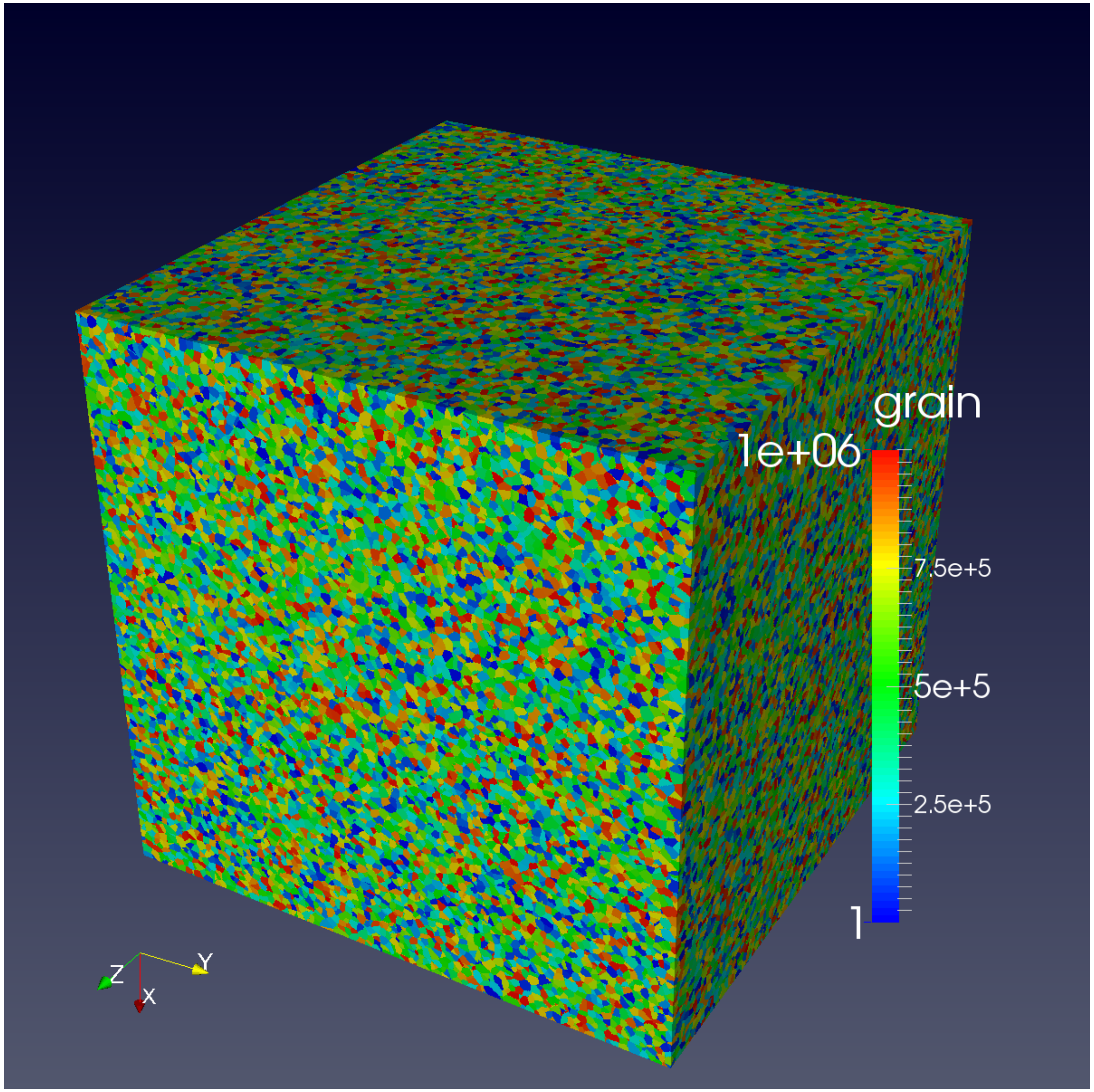
- Cellular Grains PACKage (CGPACK):
  Fortran 2008 code for microstructure simulation

  Main model coarray:

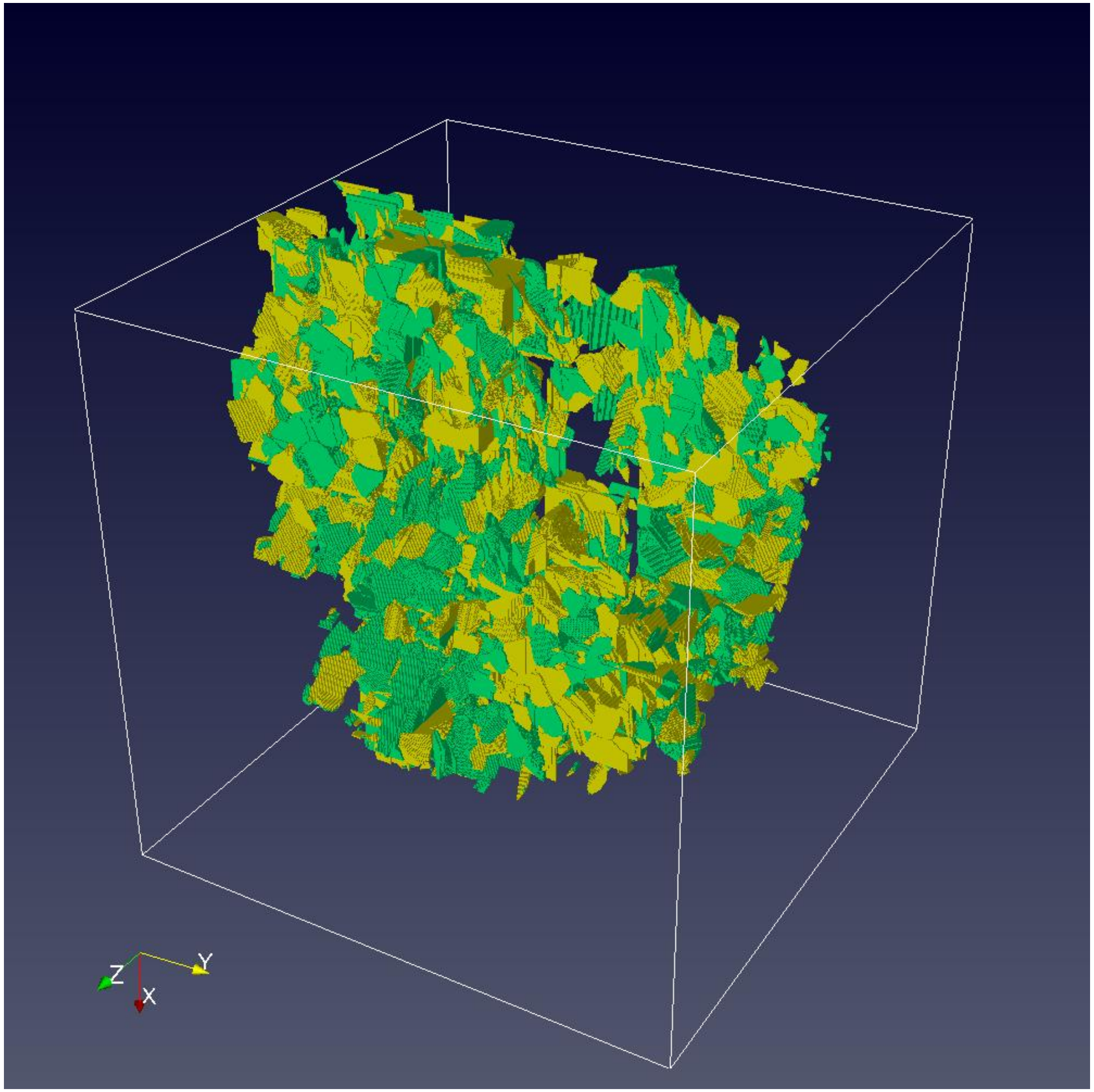
  ```
  integer, allocatable :: space(:,:,:,:) [:,:,:]
  ```

- Can be coupled with continuum mechanics (for example ParaFEM) to understand influence of macroscopic effects like temperature and stress on microstructure evolution
- Use Case: brittle fracture

Grain Structure



Fracture: Polycrystalline Iron

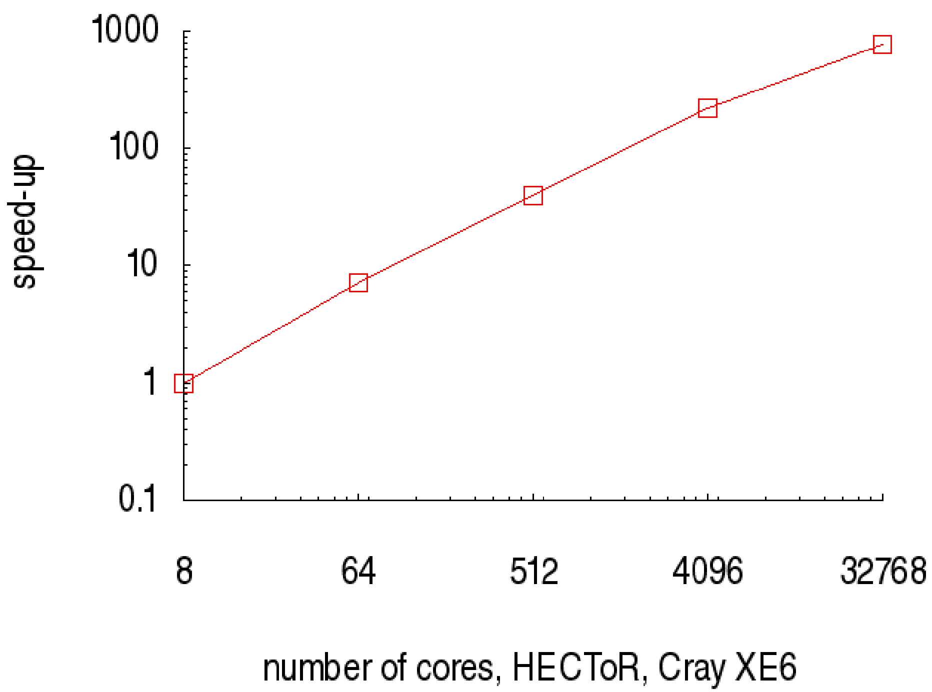


Only fractures shown

Yellow:
{100} cracks,
Green:
{110} cracks

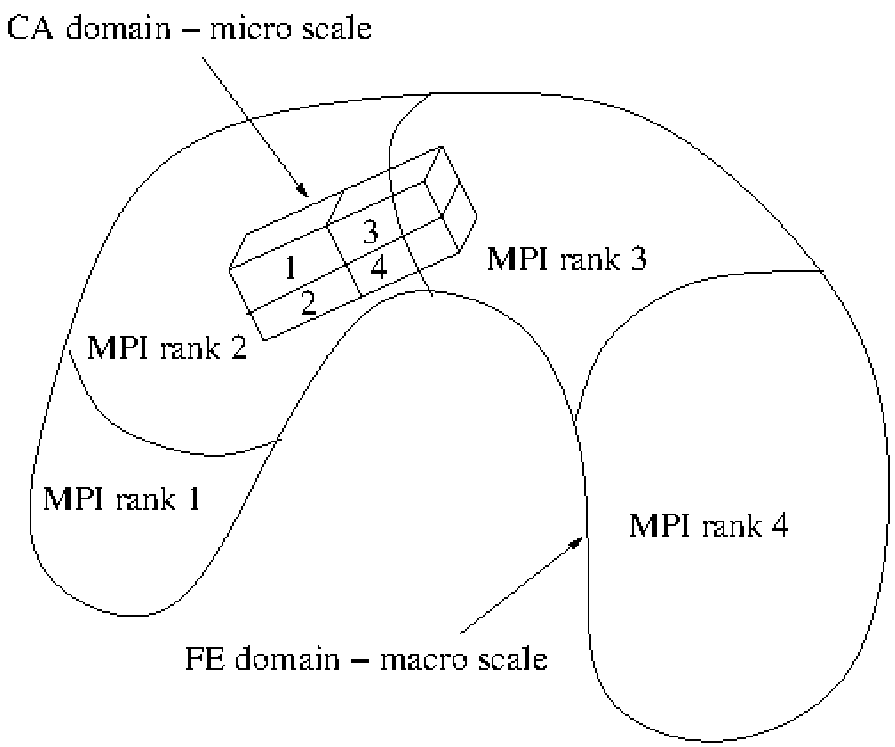
CGPACK

- **Parallelism is required to get to the large memory sizes and performance required**
- **A coarray implementation was relatively simple**
  - Divide domain in boxes
  - Communication between boxes (halo-swap) using coarray operations
- **Microstructure generation code can be very scalable:**



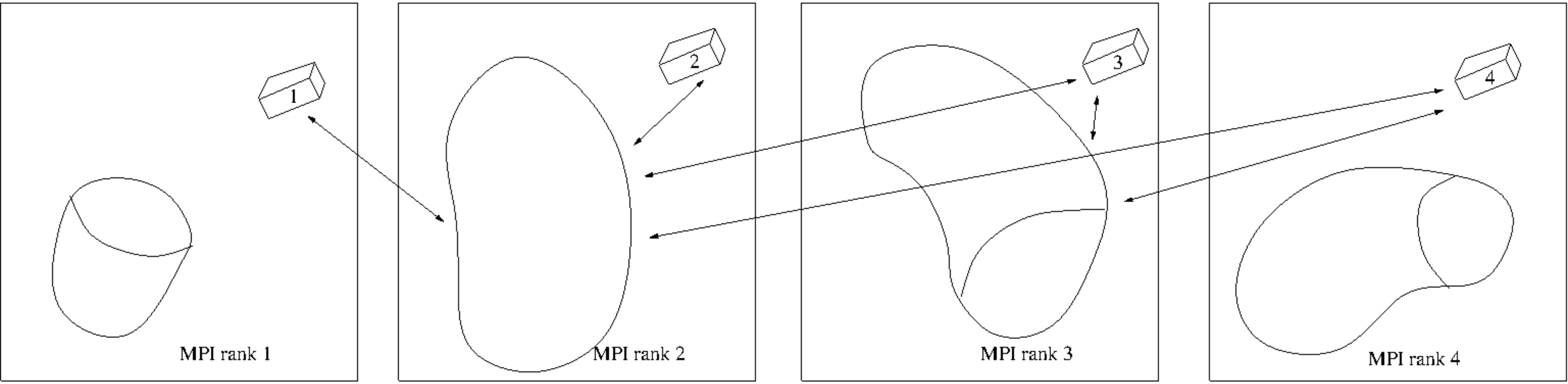
Multiscale simulations

- On some platforms you can interface MPI and coarrays: to connect a Finite Element engineering code to CGPACK
- The communication patterns (MPI process to coarray image) are complex but not a problem:



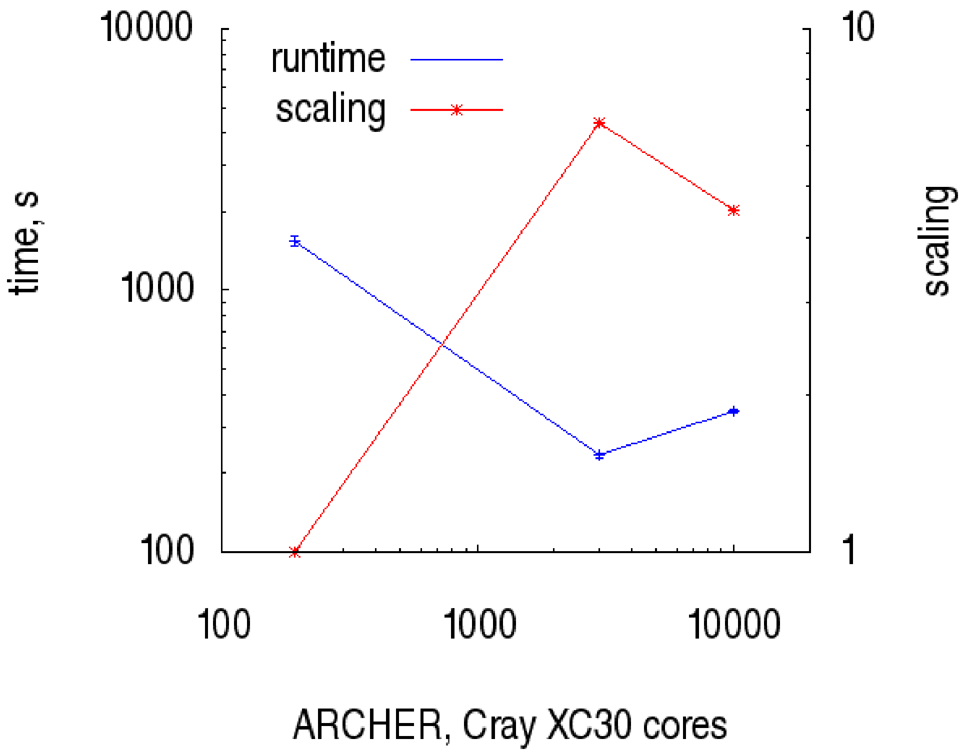
Multiscale simulation communication pattern

- MPI ranks and coarray images need to communicate
- So for example from previous slide



Current status

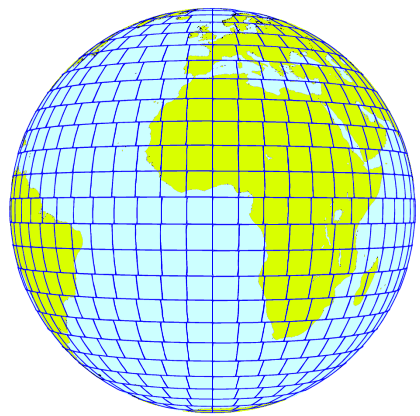
- Some problems are going to need work to get scaling

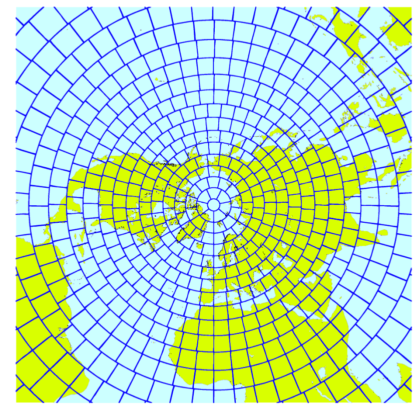


- ARCHER eCSE project was funded to investigate scaling and performance

ECMWF Integrated Forecasting system (IFS)

George Mozdzynski,
European Centre for Medium-Range Weather Forecasts



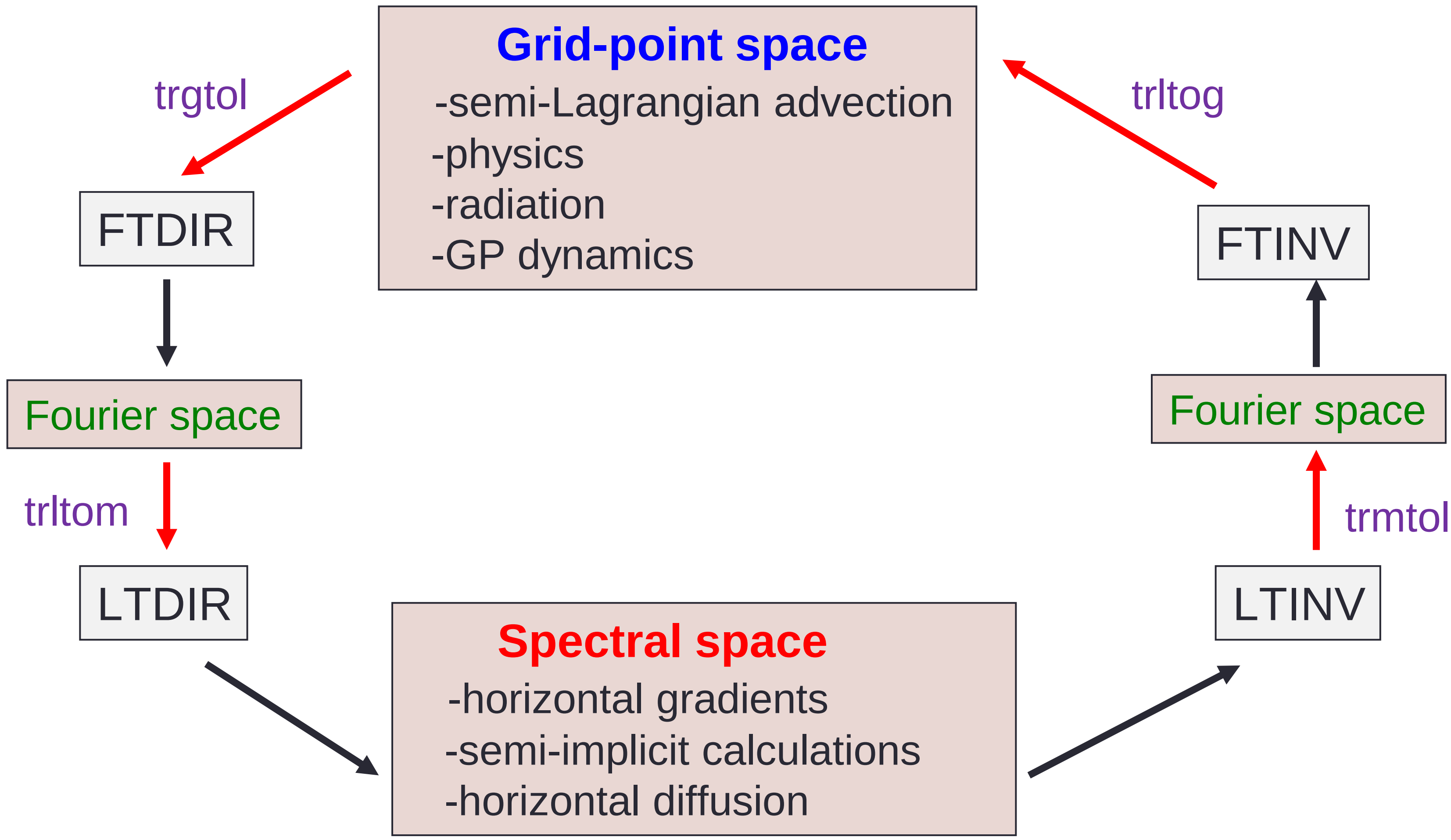


- **IFS is ECMWF's production forecasting code**
- **Work was undertaken as part of the EU *Collaborative Research into Exascale Systemware, Tools and Applications* (CRESTA) project**

ECMWF Integrated Forecasting system (IFS)

- **Need to move to higher resolution**
- **Communication costs will dominate computation**
- **Significant work undertaken to optimize transforms and transposes:**
  - Overlap Legendre transforms and associated transpositions
  - Overlap Fourier transforms and associated transpositions
- **Monolithic MPI Alltoallv implementation replaced by coarray puts directly from some OpenMP loops (data sent as soon as available)**
- **Use of Fortran coarrays was natural choice for ECMWF**
- **Very promising scalability improvement for large testcases**

Schematic description of spectral transform method in the IFS model (single time-step)



LTINV Recoding: From MPI to coarrays

```
!$OMP PARALLEL DO SCHEDULE(DYNAMIC,1) PRIVATE(JM,IM)
DO JM=1,D%NUMP
  IM = D%MYMS(JM)
  CALL
LTINV(IM,JM,KF_OUT_LT,KF_UV,KF_SCALARS,KF_SCDERS,ILEI2,IDIM1,&
   & PSPVOR,PSPDIV,PSPSCALAR ,&
   & PSPSC3A,PSPSC3B,PSPSC2 , &
   & KFLDPTRUV,KFLDPTRSC,FSPGL_PROC)
ENDDO
!$OMP END PARALLEL DO
DO J=1,NPRTRW
  ILENS(J) = D%NLTSFTB(J)*IFIELD
  IOFFS(J) = D%NSTAGT0B(J)*IFIELD
  ILENR(J) = D%NLTSGTB(J)*IFIELD
  IOFFR(J) = D%NSTAGT0B(D%MSTABF(J))*IFIELD
ENDDO
CALL MPL_ALLTOALLV(PSENDBUF=FOUBUF_IN,KSENDCOUNTS=ILENS,&
 & PRECVBUF=FOUBUF,KRECVCOUNTS=ILENR,&
 & KSENDDISPL=IOFFS,KRECVDISPL=IOFFR,&
 & KCOMM=MPL_ALL_MS_COMM,CDSTRING='TRMTOL:')
```

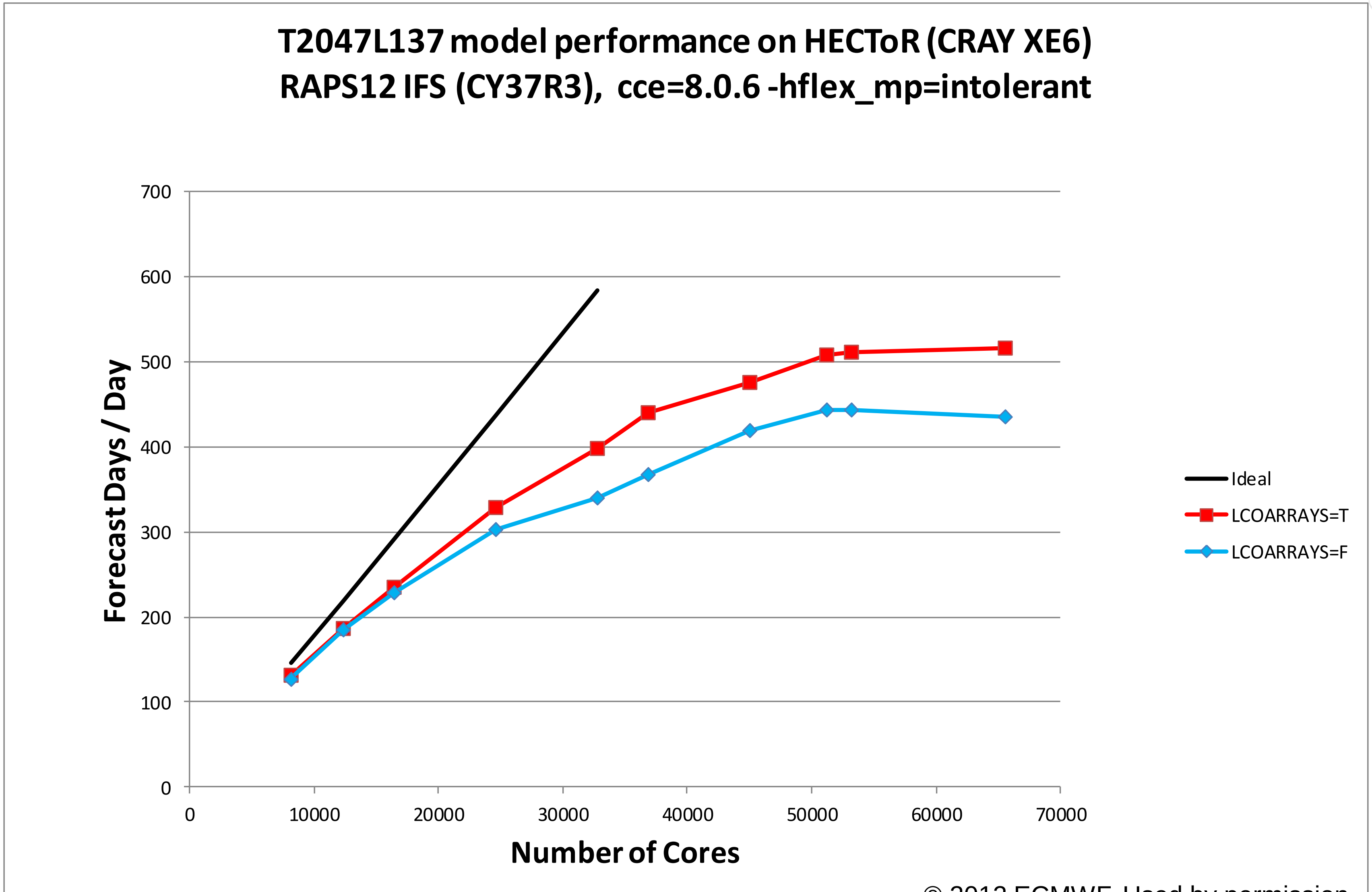
COMPUTE
COMMUNICATION

LTINV Recoding: From MPI to coarrays

```
!$OMP PARALLEL DO SCHEDULE(DYNAMIC,1)
  PRIVATE(JM,IM,JW,IPE,ILEN,ILENS,IOFFS,IOFFR)
DO JM=1,D%NUMP
  IM = D%MYMS(JM)
  CALL LTINV(IM,JM,KF_OUT_LT,KF_UV,KF_SCALARS,KF_SCDERS,ILEI2,IDIM1,&
    & PSPVOR,PSPDIV,PSPSCALAR ,&
    & PSPSC3A,PSPSC3B,PSPSC2 , &
    & KFLDPTRUV,KFLDPTRSC,FSPGL_PROC)
  DO JW=1,NPRTRW
    CALL SET2PE(IPE,0,0,JW,MYSETV)
    ILEN = D%NLEN_M(JW,1,JM)*IFIELD
    IF( ILEN > 0 )THEN
      IOFFS = (D%NSTAGT0B(JW)+D%NOFF_M(JW,1,JM))*IFIELD
      IOFFR = (D%NSTAGT0BW(JW,MYSETW)+D%NOFF_M(JW,1,JM))*IFIELD
      FOUBUF_C(IOFFR+1:IOFFR+ILEN)[IPE]=FOUBUF_IN(IOFFS+1:IOFFS+ILEN)
    ENDIF
    ILENS = D%NLEN_M(JW,2,JM)*IFIELD
    IF( ILENS > 0 )THEN
      IOFFS = (D%NSTAGT0B(JW)+D%NOFF_M(JW,2,JM))*IFIELD
      IOFFR = (D%NSTAGT0BW(JW,MYSETW)+D%NOFF_M(JW,2,JM))*IFIELD
      FOUBUF_C(IOFFR+1:IOFFR+ILENS)[IPE]=FOUBUF_IN(IOFFS+1:IOFFS+ILENS)
    ENDIF
  ENDDO
ENDDO
!$OMP END PARALLEL DO
SYNC IMAGES(D%NMYSETW)
FOUBUF(1:IBLEN)=FOUBUF_C(1:IBLEN)[MYPROC]
```

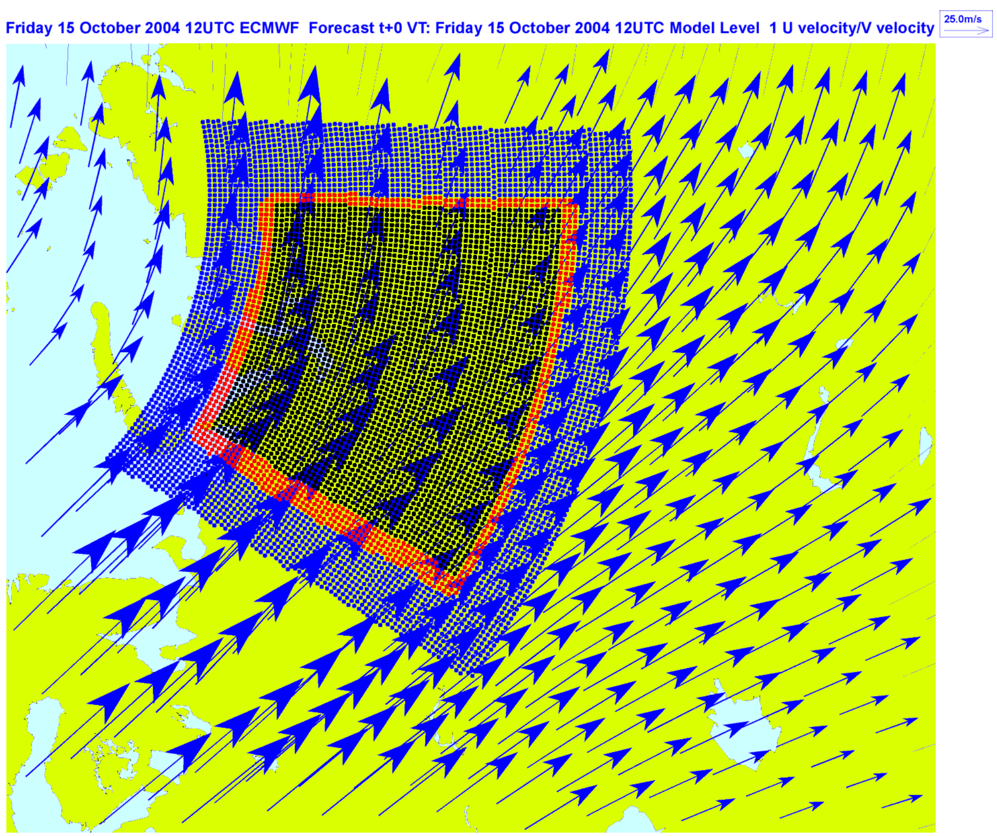
COMPUTE
COMMUNICATION

IFS scaling with coarrays



Coarrays for halo-swap

- Later work targeted communication halos for semi-lagrangian transport
- Grid-point trajectories cross MPI task boundaries



IFS work status

- **This work was a research project**
- **Before considering for production code would need:**
  - More compilers supporting coarrays
  - Other features (teams) to be implemented

References

- "Multithreaded Address Space Communication Techniques for Gyrokinetic Fusion Applications on Ultra-Scale Platforms", *Robert Preissl, Nathan Wichmann, Bill Long, John Shalf, Stephane Ethier, Alice Koniges*, SC11 best paper finalist
- Shterenlikht, A, Margetts, L, Cebamanos, L & Henty, D 2015, 'Fortran 2008 coarrays'. *ACM SIGPLAN Fortran Forum*, vol 34., pp. 10-30
- Shterenlikht, A & Margetts, L 2015, 'Three-dimensional cellular automata modelling of cleavage propagation across crystal boundaries in polycrystalline microstructures'. *Proc R Soc*, vol 471.
- "A PGAS implementation by co-design of the ECMWF Integrated Forecasting system," George Mozdzynski,, 15th Workshop on High Performance Computing in Meteorology, 1-5 October 2012.

Questions?

Backup Slides

Extra information if you want to try

- VMs and VirtualBox
- Installing the OpenCoarrays stack from scratch
- Using the Sourcery Institute VM

Just be aware that you need to be familiar with software installation and have GBs of disk space to try these

Virtual Machines and Virtualbox

- Software Virtualisation is an easy way to experiment with software in isolation
- This is a particularly useful approach for Linux software on Microsoft Windows.
- VMWare and Oracle have products

Virtualbox

- The author's preference, low overhead and free
- Lets you install an OS inside software inside an OS
- Came via a Sun Microsystems acquisition, now owned by Oracle
- Open Source license with more restrictively licensed Extension Pack (USB support etc.) available for Personal Use

VirtualBox

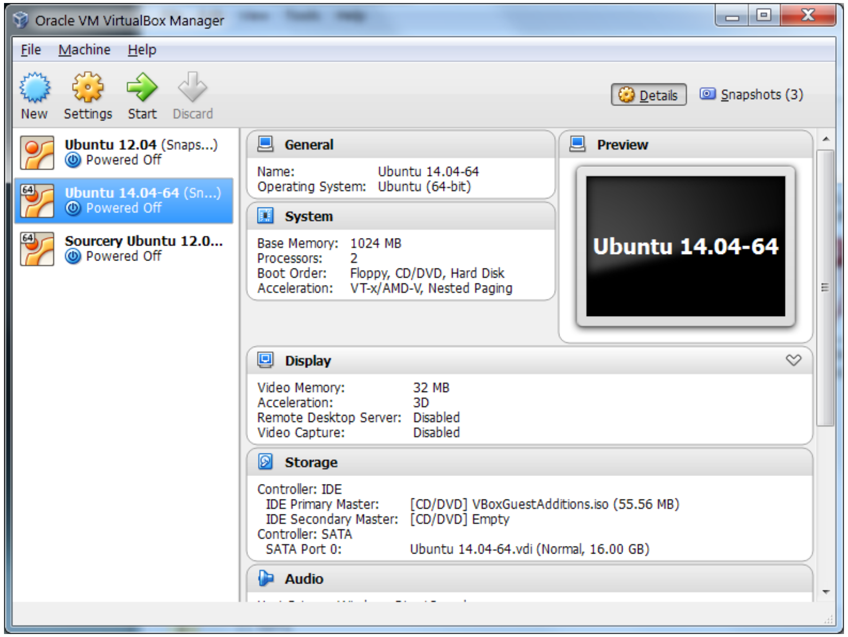
- Get from http://www.virtualbox.org
- Latest stable version is 4.3.30
- Versions for Windows, Mac OS and Linux

Installation

- Download installer, extension pack, user guide
- Run the Installer
- Choose what to install
  USB support (if you want to use your USB hardware)
  Networking: Host-Only (within your host)
  Bridged (VM visible outside your host)
- Some post-install configuration advised:
  Add Extension Pack (preferences) if you want features
  Set filesystem location for VMs (they can be LARGE)

Installing VMs: Example Ubuntu 14.04 LTS

- **Download an ISO image for the OS: ubuntu-14.04.3-desktop-amd64.iso**
- **Create new VM:**
  - Virtual drive: 16GB (dynamic)
  - RAM: 2GB
  - cpus: 2
- **Point CD device to the ISO (Storage,controller/IDE/+)**



- **Start VM and follow OS install instructions**
- **Post VM install: Use VM menu to add guest additions**

Ubuntu VM: Post-install configuration

This is what I do:

- Login and search for terminal so you can…
- Install graphical package manager and updates (if you did not do updates during install):

```
% sudo apt-get update
% sudo apt-get install synaptic
% sudo apt-get upgrade
```

- Install alternative login choices (if you hate Unity)

```
% sudo apt-get install gnome-session-fallback
```

- Install dynamic kernel module support (used to be required for Virtualbox but does not harm)

```
% sudo apt-get install dkms
```

- Reboot VM, Install guest additions from VM menu and agree to prompts
- Reboot VM again and resize VM window if you want

Installing OpenCoarray prerequisites in the VM

- Install m4 and g++ from command line or gui
  `% sudo apt-get install m4 g++`
- … these are required for the gcc/gfortran build

- Create a new location for your software stack
  % dir=/usr/local2
  % sudo mkdir /usr/local2
  % sudo chown $USER /usr/local2

- We will put our software in /usr/local2/build
  % mkdir /usr/local2/build

- In subsequent slides we use $dir instead of /usr/local2

Installing OpenCoarray prerequisites in the VM

- Now for gcc (expect 2 hours or so using one cpu)
- Download (into $dir/build)

```
% wget \
ftp://ftp.mirrorservice.org/sites/sourceware.org/pub/gcc/releases/gcc-5.2.0/gcc-5.2.0.tar.bz2
% gunzip -c gcc-5.2.0.tar.bz2 | tar xfv -
% cd $dir/build-gcc-5.2.0
```

- Now get prerequisite libs, configure and build:

```
% ./contrib/download_prerequisites
% mkdir -p objdir
% cd objdir
% ../configure --prefix=$dir \
      --enable-languages=c,c++,fortran \
      --disable-multilib
% make
% make install
```

Installing OpenCoarrays in the VM…

Now make sure you use your new software from now on:

```
% export PATH=$dir:$PATH
% export LD_LIBRARY_PATH=$dir/lib64
```

- … perhaps create a setup.rc file to do this

Cmake

- wget http://www.cmake.org/files/v3.3/cmake-3.3.2.tar.gz
- Uncompress/untar in build dir and cd to source directory

```
% ./configure --prefix=$dir
% make
% make install
```

MPICH

- http://www.mpich.org/static/downloads/3.1.4/mpich-3.1.4.tar.gz
- Unpack and install similarly to Cmake (remember --prefix)
- Apparently OpenMPI also fine but did not try it

Installing OpenCoarrays in the VM…

OpenCoarrays

- Download from website or wget \
  https://github.com/sourceryinstitute/opencoarrays/releases/download/1.0.1/opencoarrays-1.0.1.tar.gz
- `mkdir -p build`
- `cd build`
- `CC=$dir/bin/mpicc FC=$dir/bin/mpif90 cmake .. \`
  `-DCMAKE_INSTALL_PREFIX=$dir`
- `make`
- `make install`
- `ctest`

Using OpenCoarrays in the VM

Final Setup:

- remember to include new software (/usr/local2/bin) in path and libraries (/usr/local2/lib64) in LD_LIBRARY_PATH

Now you should be able to

- Compile a Fortran coarray program with the caf wrapper
- Run program using *N* images

```
% mpirun -n N ./a.out
```

Using the Sourcery Institute VM

- **You don't need to build all the software**
- **You still need to setup Virtualbox**
- **Then you can use a (4.3GB) prebuilt Virtual Machine**
- **Get from http://www.sourceryinstitute.org/**
- **Double-click the file and VirtualBox will configure a VM**
- **Probably want to change settings for:**
  - RAM (preset to 12GB!)
  - Number of cpus
  - USB support (you might not need any)