High performance computing for sustainable development

Scientific and engineering computation in practice

Charles Crosby
CHPC
April 2023

Introduction – HPC and the UN's sustainable development goals

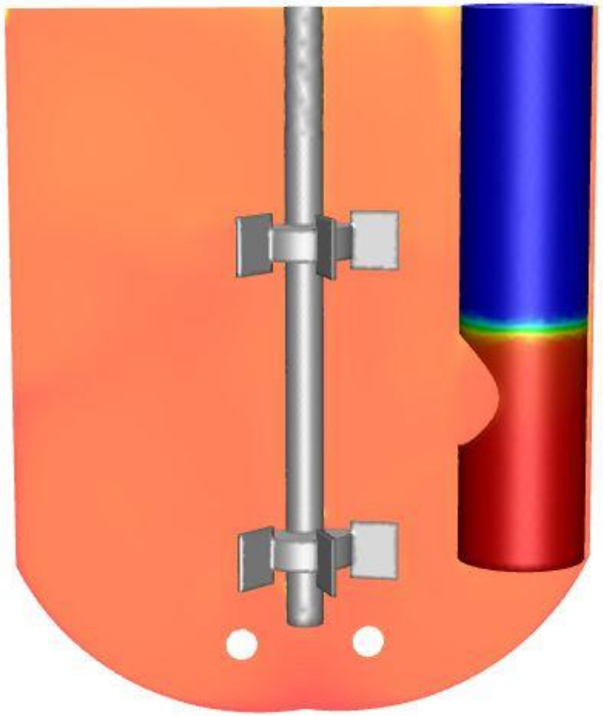
The goals that we can easily relate to HPC:

- 4 – Quality education
- 6 – Clean water & sanitation
- 7 – Affordable & clean energy
- 9 – Industry, innovation & infrastructure
- 11 – Sustainable cities & communities
- 13 – Climate action

Sustainable development goals

NATIONAL INTEGRATED CYBERINFRASTRUCTURE SYSTEM
CHPC

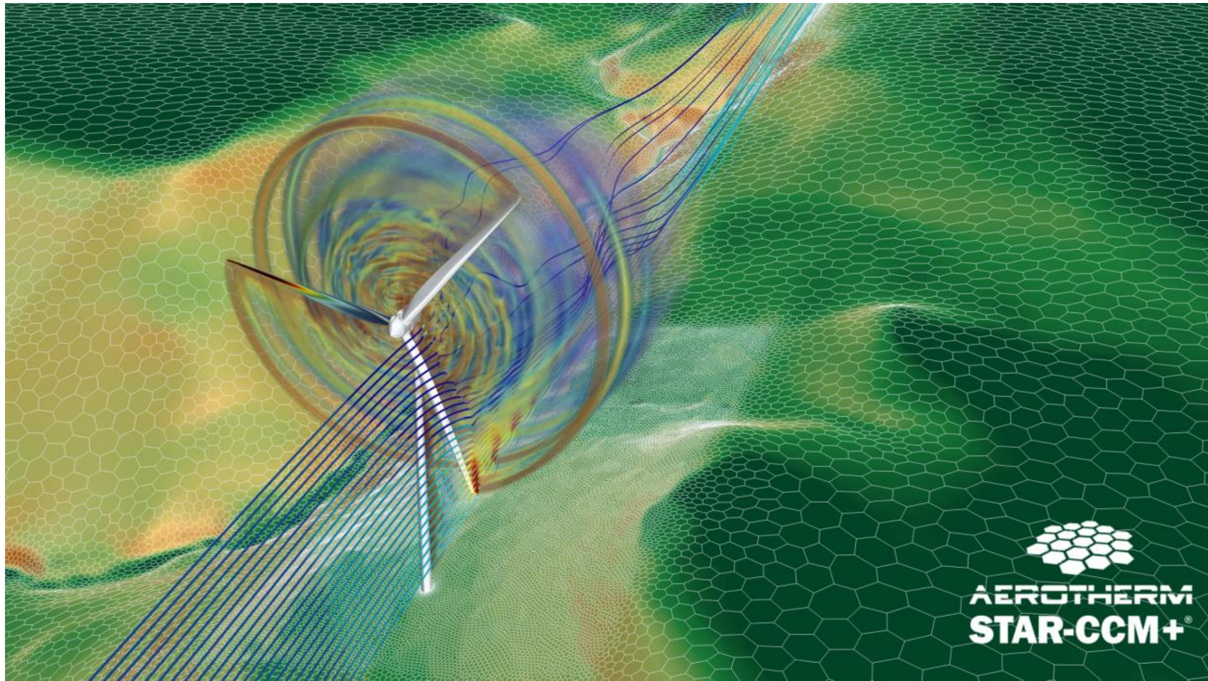
- 4 – Quality education
  - HPC requires appropriately educated people, especially with the prevalence of AI
- 6 – Clean water & sanitation
  - A field where we see quite a lot of HPC-empowered work
    - Water treatment
    - Dams, distribution systems
    - A field that requires HPC-empowered engineering



A national initiative of the Department of Science and Innovation and implemented by the CSIR

Sustainable development goals

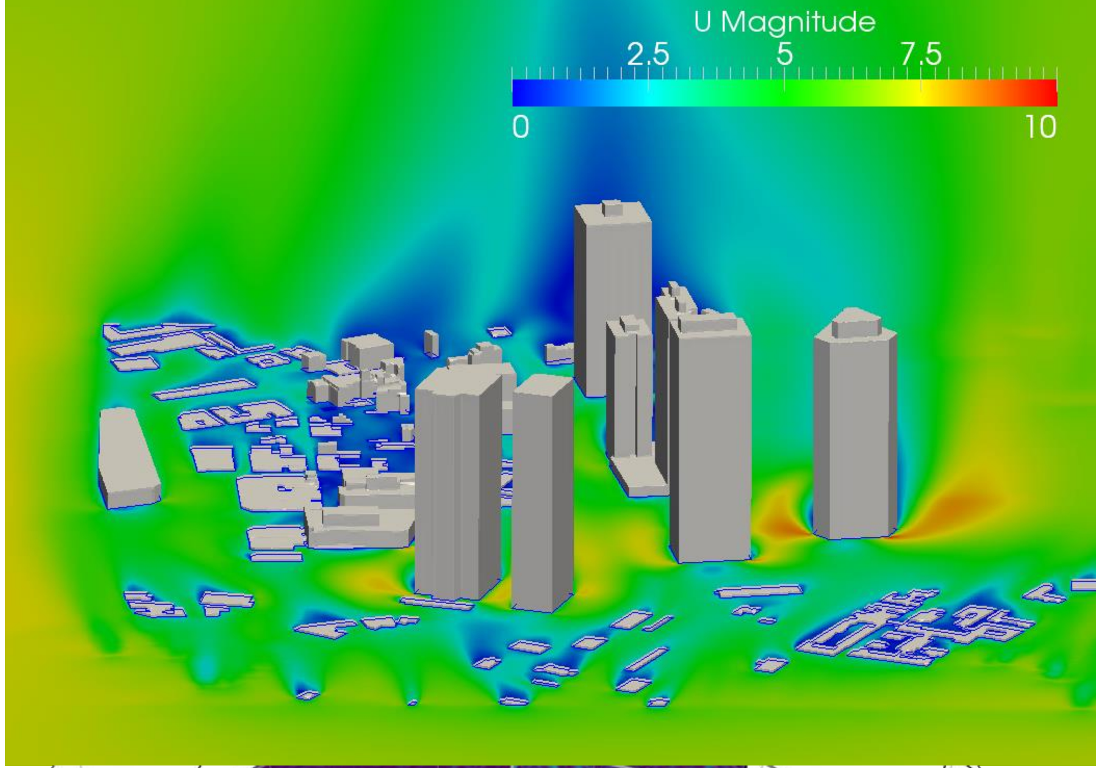
- 7 – Affordable & clean energy (the Big One)
  - Energy system implies flow of heat, energy, particles
  - We need sustainable systems, which must be efficient, affordable, safe
  - Better energy storage
- 9 – Industry, innovation & infrastructure
  - New ideas only work if they are:
    - Feasible
    - Refined

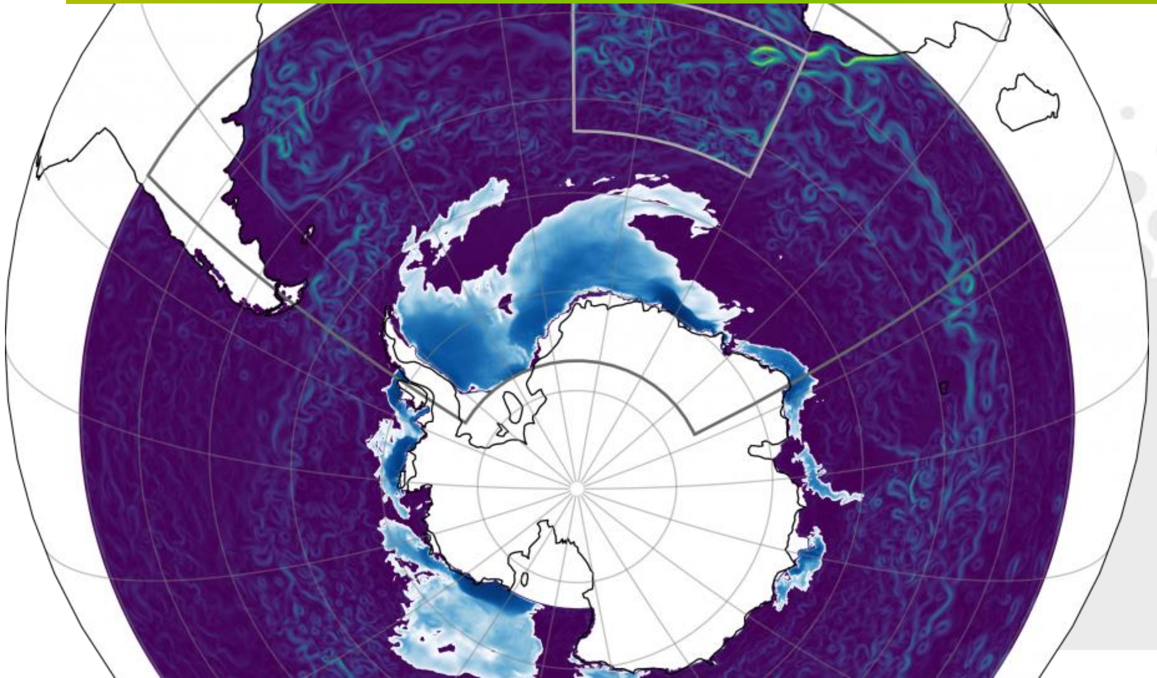


A national initiative of the Department of Science and Innovation and implemented by the CSIR

Sustainable development goals

- 11 – Sustainable cities & communities
  - Find better ways of doing things
  - The challenge – how do we accommodate AI without creating a dystopia?
- 13 – Climate action (the other Big One)
  - Climate is complex, complex, complex. HPC is indispensable for developing the understanding of all the interactions.
  - Very big user of HPC resources everywhere in the world, also at the CHPC

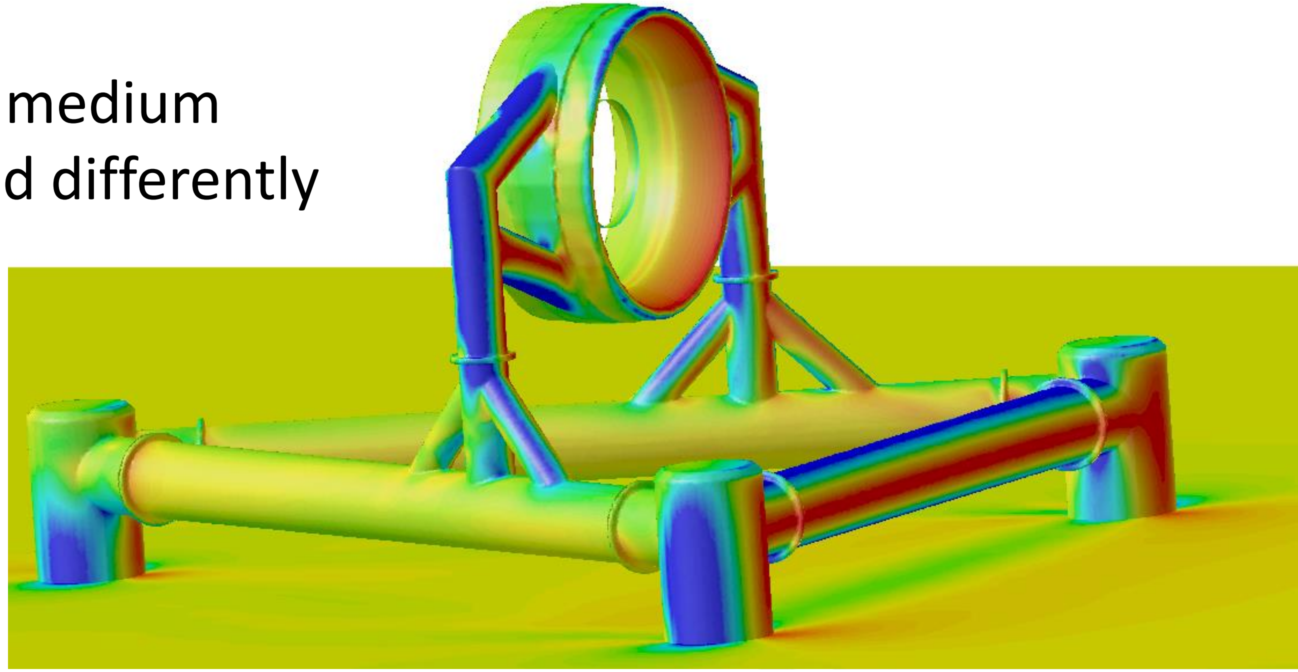




Computational mechanics and earth sciences at the CHPC

NICIS NATIONAL INTEGRATED CYBERINFRASTRUCTURE SYSTEM CHPC

- Computational mechanics (heat transfer, fluids, structures, electromagnetics, particles) and earth sciences have some common computational elements:
  - Vary in three dimensions
  - Vary in time
  - Atmosphere and oceans are fluids
  - Earth's crust → porous (mostly) solid medium
- Actual methods are similar, but developed differently
- Mostly not embarrassingly parallel
- Generally parallelise very well
- Sparse matrices

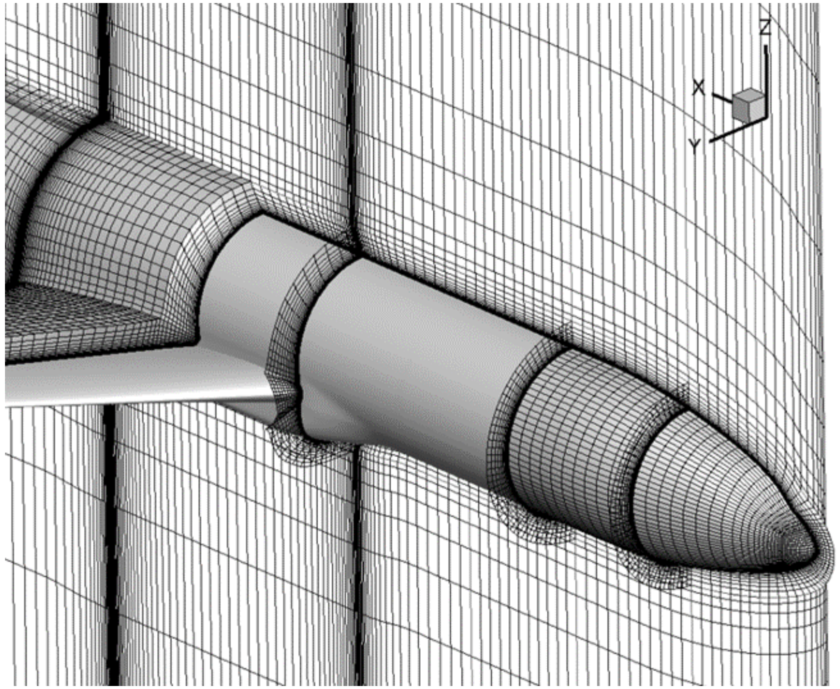


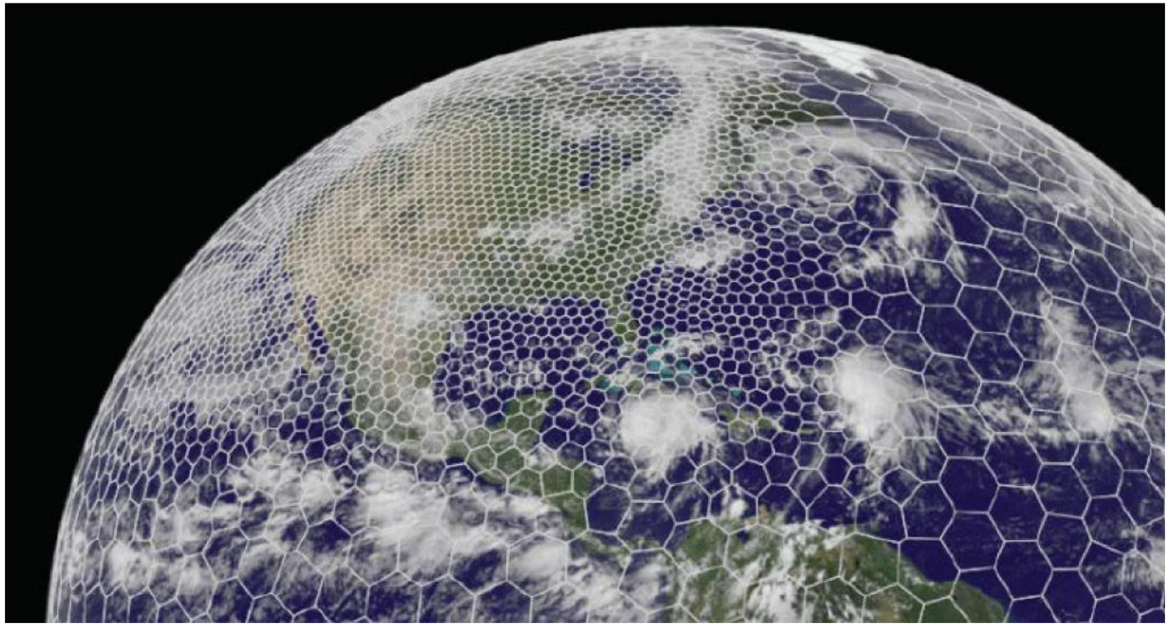
A national initiative of the Department of Science and Innovation and implemented by the CSIR

science & innovation Department: Science and Innovation REPUBLIC OF SOUTH AFRICA CSIR

Working with transport equations

Divide and conquer

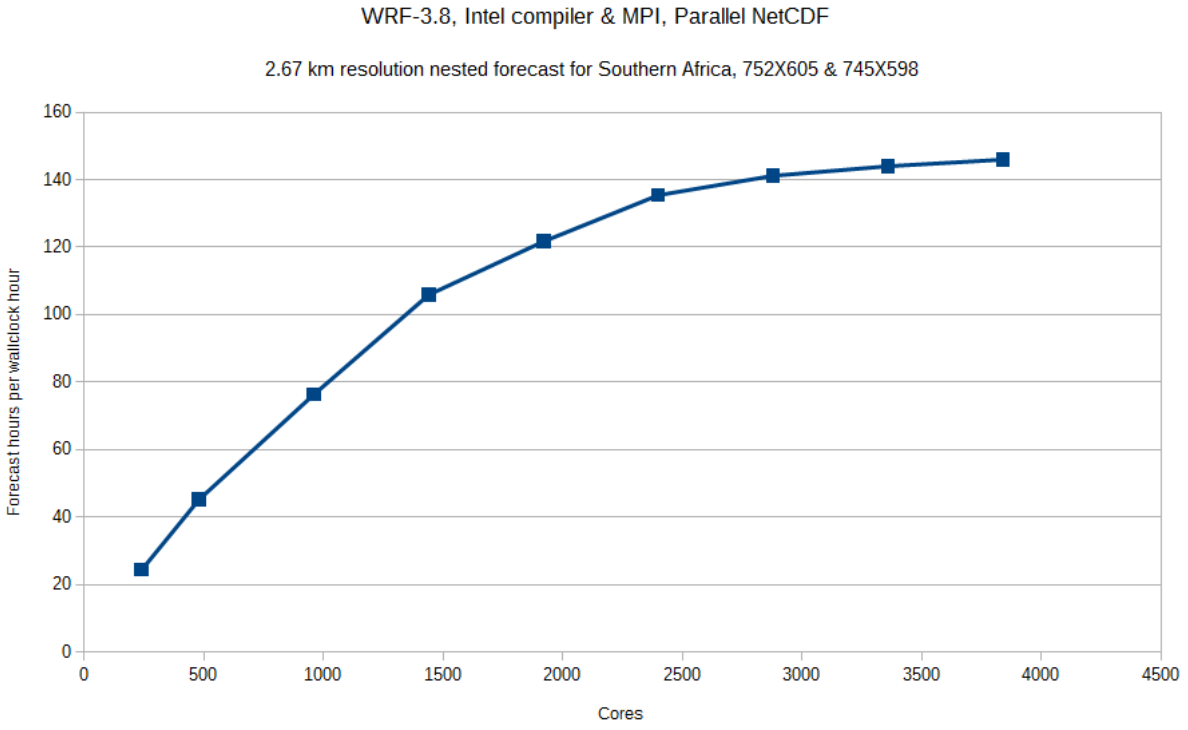




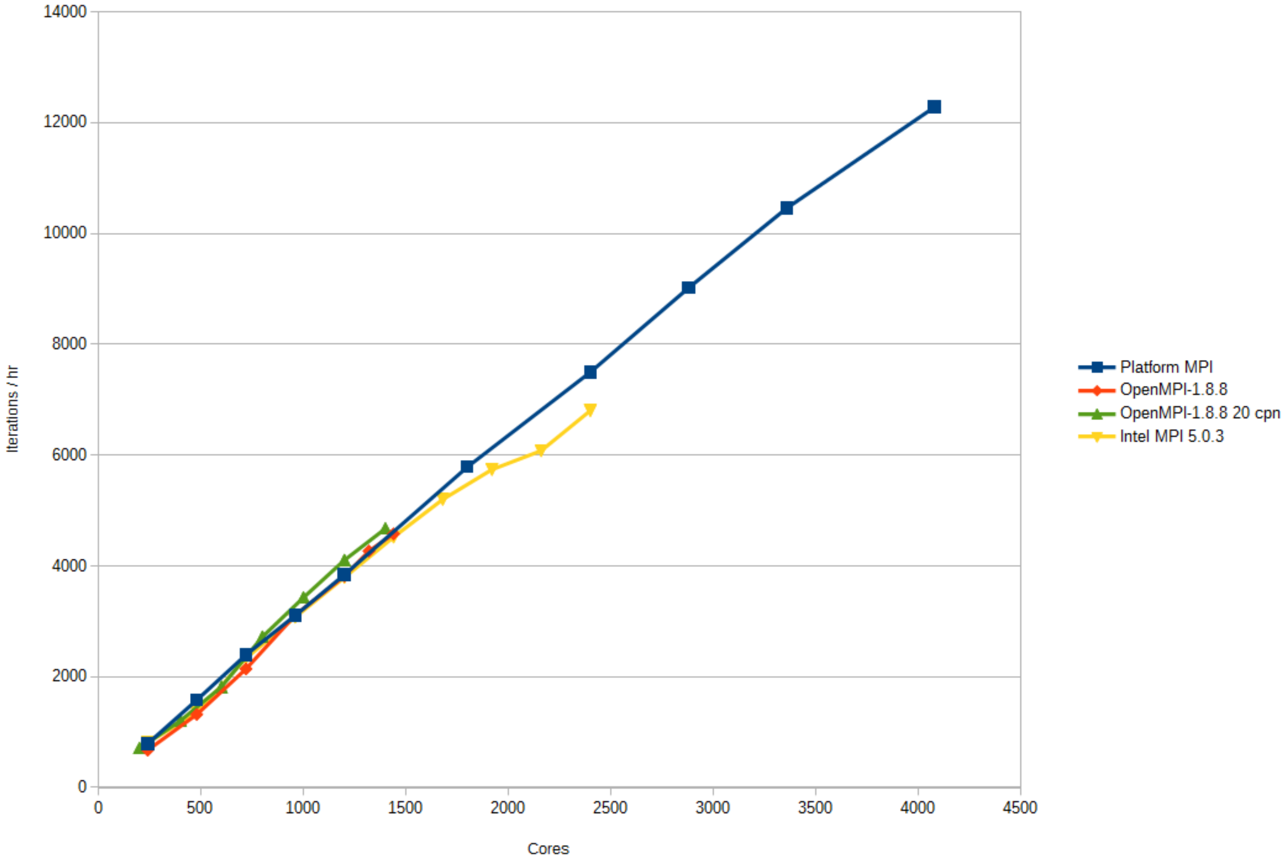
- Break up reality into small pieces – finite volumes or finite elements
- Parcel out chunks of pieces to different processors

Parallel performance

Transport equation solvers scale nicely ..

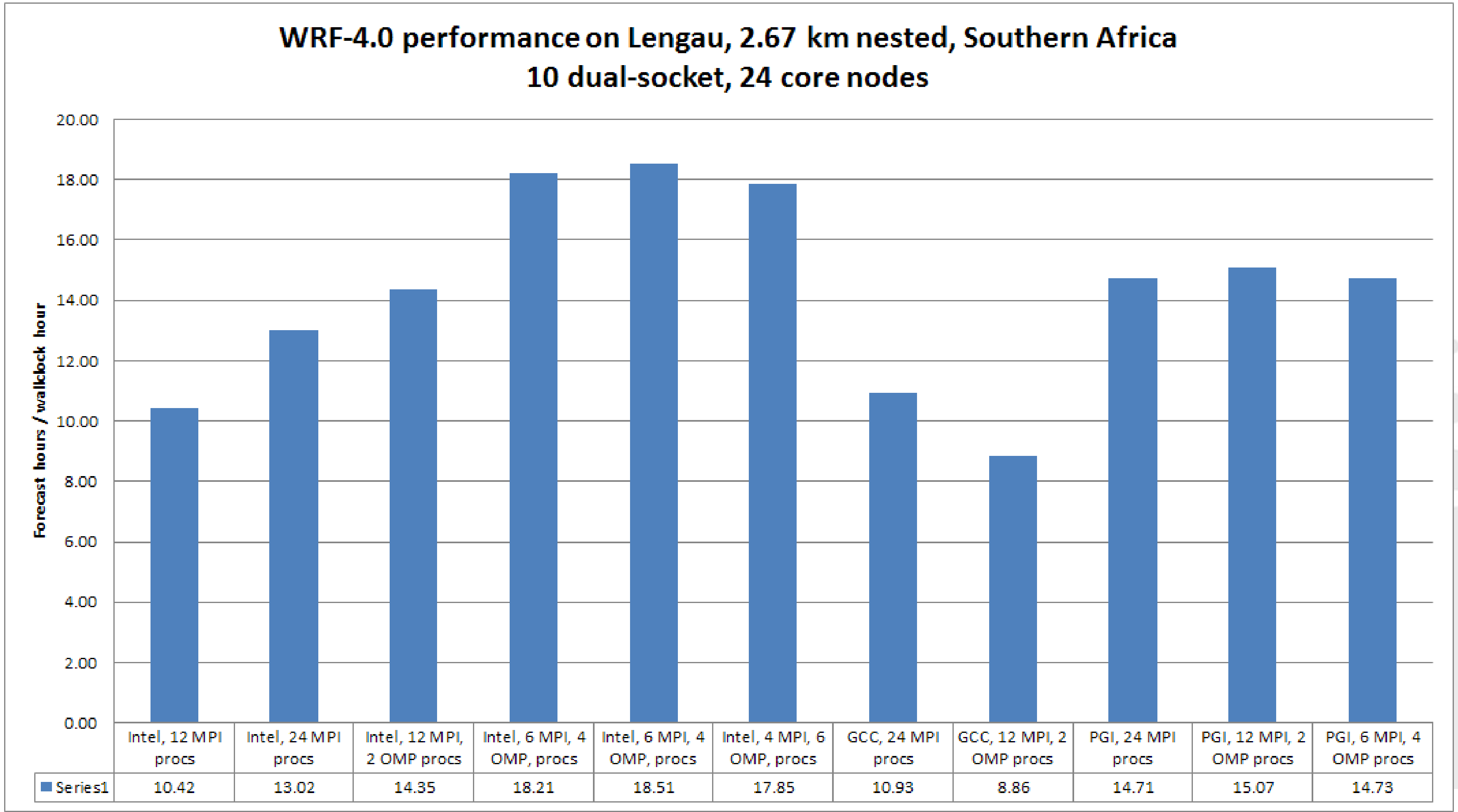


105 M cell Le Mans car, Coupled Star-CCM+ 11.04.010



Parallel performance

…. but things like compiler choice may matter



A national initiative of the Department of Science and Innovation and implemented by the CSIR

Computational Mechanics Simulations

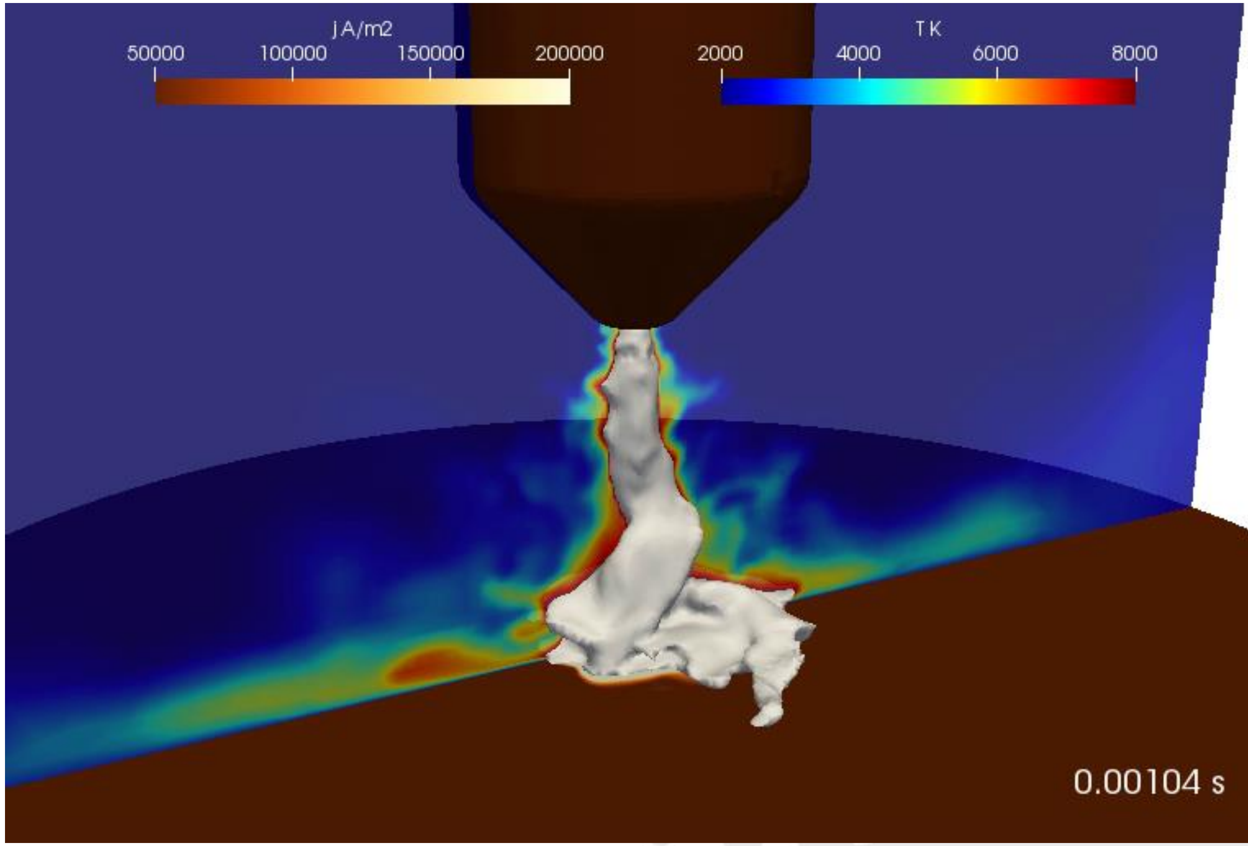
NATIONAL INTEGRATED CYBERINFRASTRUCTURE SYSTEM
CHPC

- Classic definition: “A simulation is an approximate imitation of the operation of a process or system that represents its operation over time.”
- Can be computer-based, like CFD
- Can be physical, like wind-tunnel or crash testing
- Can be both, with a “human in the loop” or “hardware in the loop”, like a flight simulator

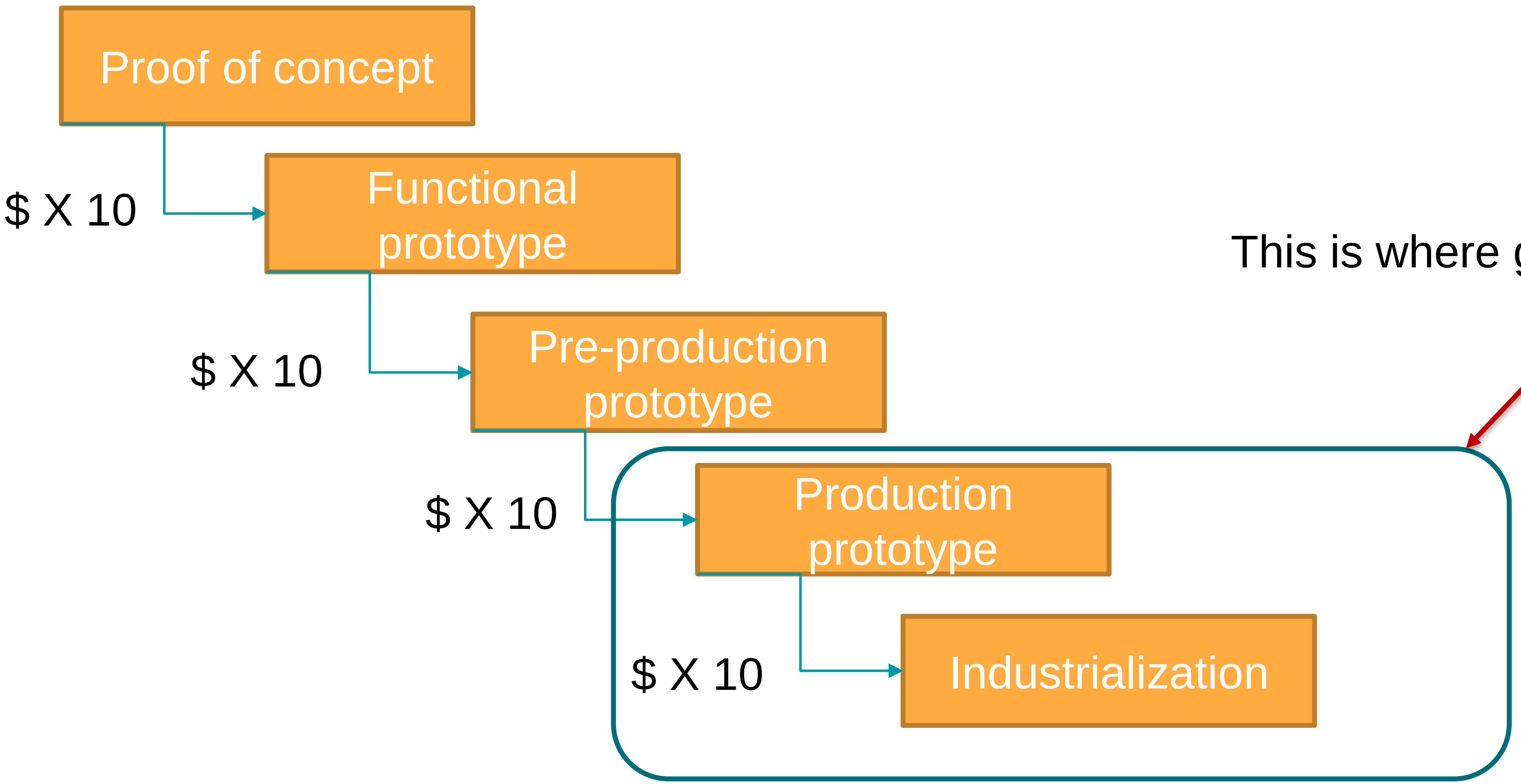
A national initiative of the Department of Science and Innovation and implemented by the CSIR

The CHPC's computational mechanics users

- Different classes of users doing different types of work:
  - University research groups
  - Science councils
  - Commercial users, typically consultants
- Classes of applications
  - Mostly fluids
  - Some granular flows
  - Some multi-physics
  - Some structures



The trouble with product development



This is where great ideas go to die

Good simulation work may be able to bring X 10 down to something more manageable

OpenFOAM – a framework for solving 3D unsteady PDEs

- Heavily abstracted C++, takes care of I/O, boundary conditions, differencing schemes, parallel processing, linear solvers, etc.
- Can code solver for a PDE in symbolic tensor notation
- Template scripts for making new solvers
- Open-source code:
  - Leverage the power of HPC
  - Avoid lock-in
  - Durability

But ….:

- The need to invest in people, skills and capability
- Minimal documentation
- Trying to use it off-design can be difficult
- Forked code-bases and community

A national initiative of the Department of Science and Innovation and implemented by the CSIR

OpenFOAM – coding example

myThermalConductionSolver.C

```
    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WIT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public Licen
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

Application
    myThermalConductionSolver

Description

\*---------------------------------------------------------------------

#include "fvCFD.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

int main(int argc, char *argv[])
{
    #include "setRootCase.H"
    #include "createTime.H"
    #include "createMesh.H"
    #include "createFields.H"

    for (int i=0; i<10; i++)
    {
       solve( fvm::laplacian(k,T) + su + fvm::Sp( sp,T) ) ;
       runTime++;
       runTime.write();
    }

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

    Info<< nl;
    runTime.printExecutionTime(Info);

    Info<< "End\n" << endl;

    return 0;
}
```

This is the simplest of examples!

"Real" solvers are a lot more involved than this …

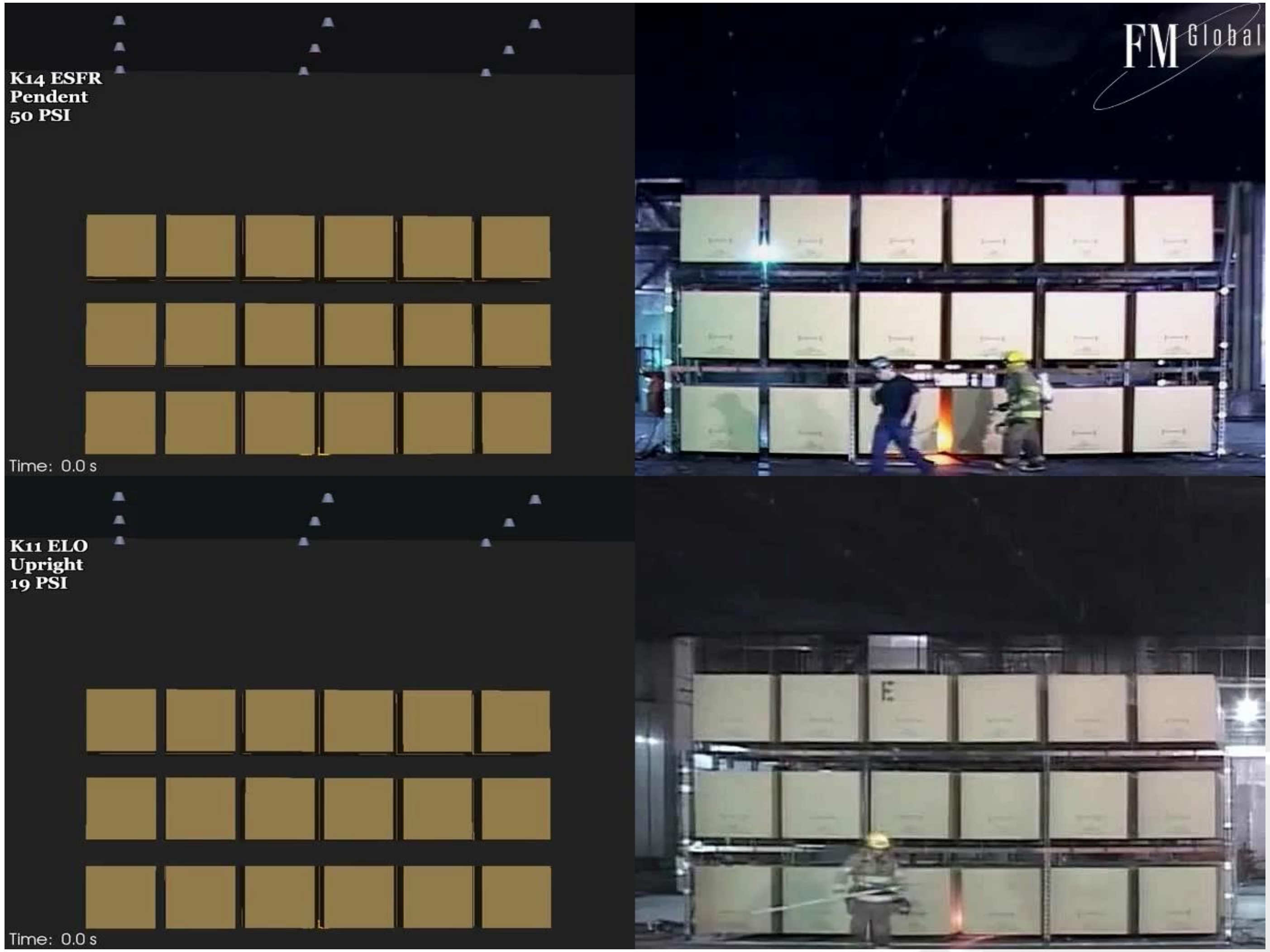
*createFields.H

```
volScalarField T
(
    IOobject
    (
        "T",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    mesh
);
IOdictionary transportProperties
(
    IOobject
    (
        "transportProperties",
        runTime.constant(),
        mesh,
        IOobject::MUST_READ_IF_MODIFIED,
        IOobject::NO_WRITE
    )
);
dimensionedScalar k
(
    "k",
    dimArea/dimTime,
    transportProperties
);
dimensionedScalar su
(
    "su",
    dimTemperature/dimTime,
    transportProperties
);
dimensionedScalar sp
(
    "sp",
    pow(dimTime,-1),
    transportProperties
);

Info << "k: " << k << endl;
Info << "su: " << su << endl;
Info << "sp: " << sp << endl;
```

Comparison of simulations

Open-source software fireFoam used in the specification of a warehouse fire suppression systems

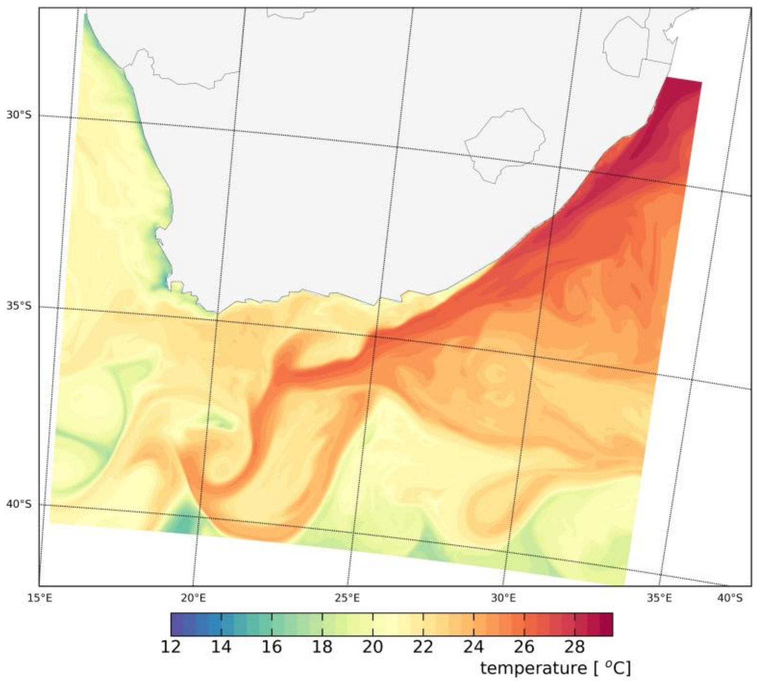


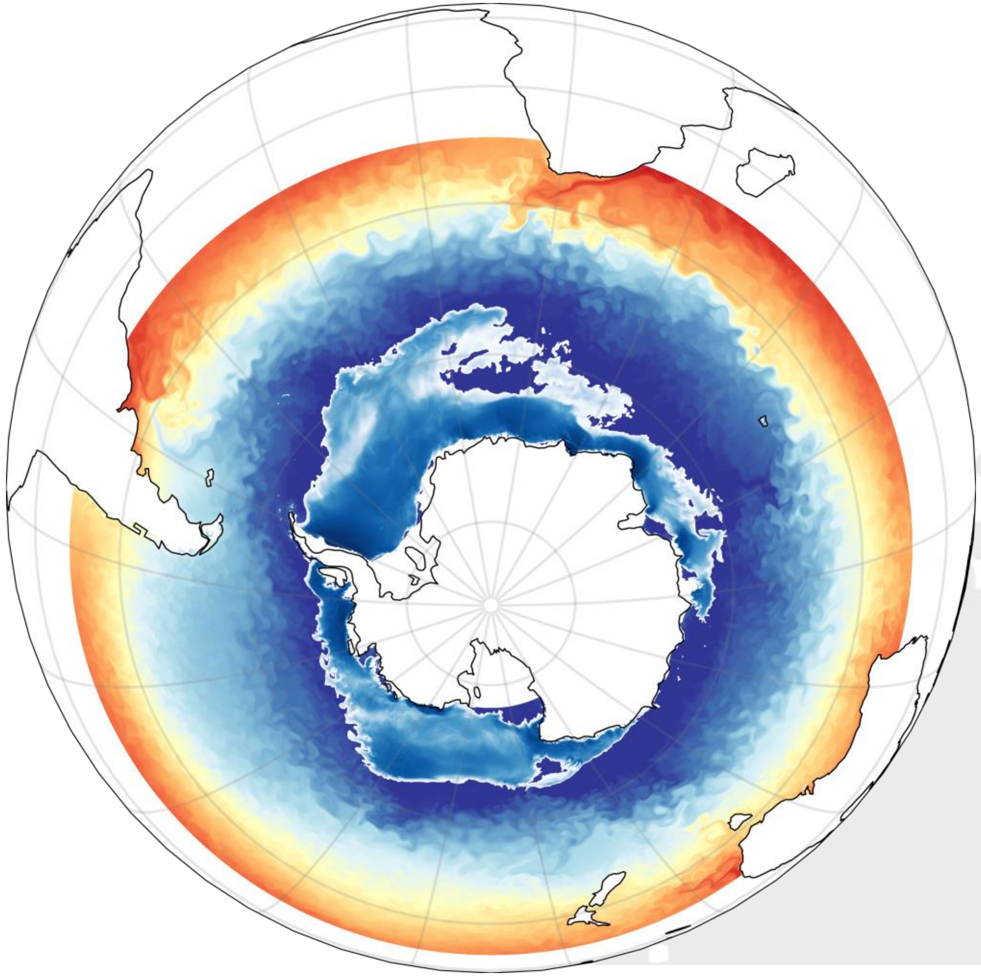
A national initiative of the Department of Science and Innovation and implemented by the CSIR

Earth science simulations

- WRF (widely used)
- RegCM
- Nemo (Ocean circulation) – Data also correlated by observational sciences
- Oasis3-MCT, ocean & atmosphere coupling
- CCAM (Australian climate code)
- Unified Model (From UK Met Office), used by SAWS for production forecasts & research

1 January 2008, modelled sea surface temperature

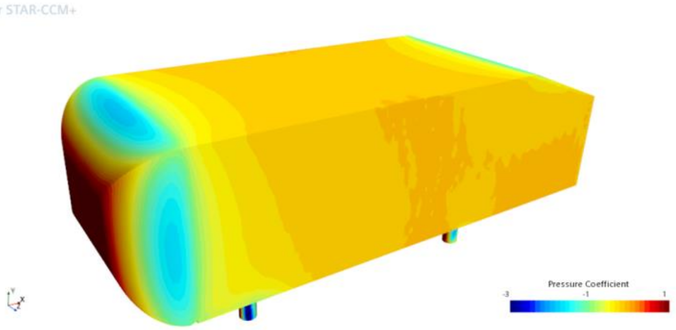




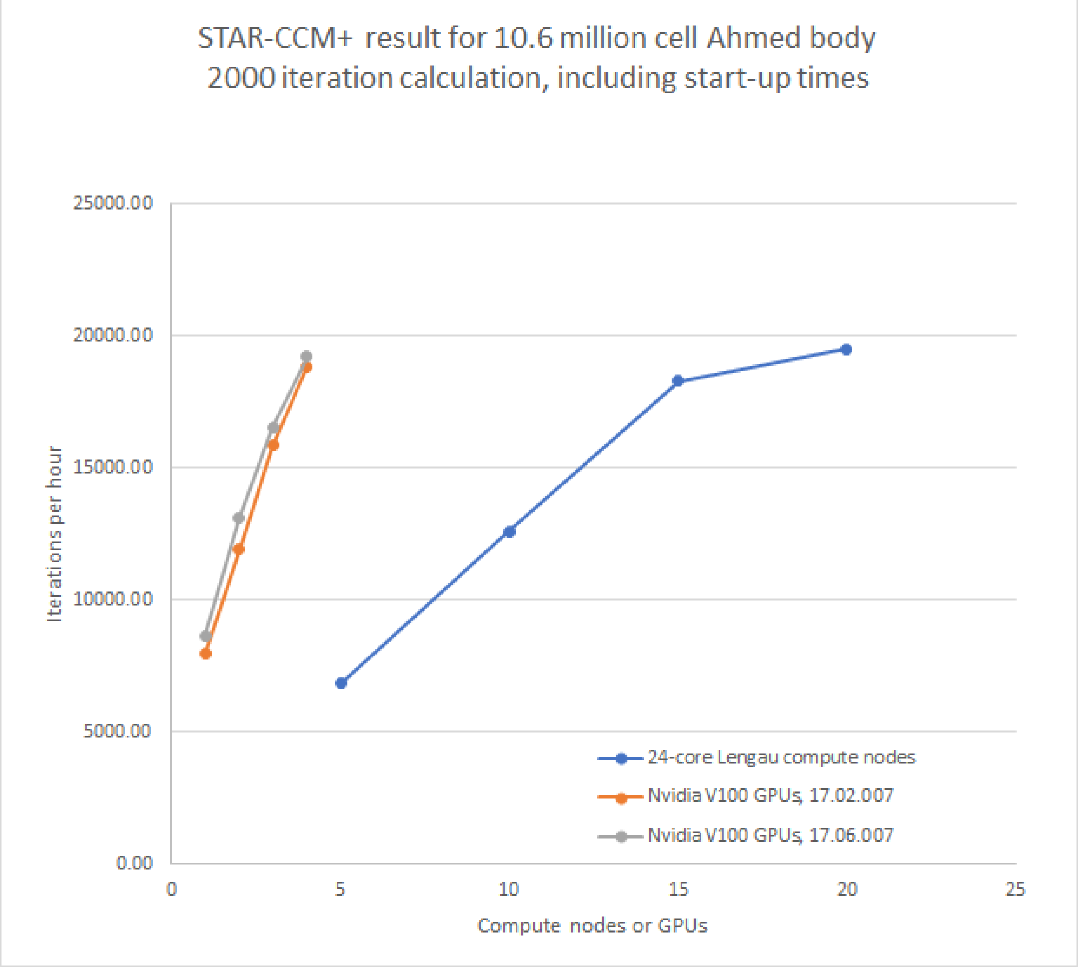
A national initiative of the Department of Science and Innovation and implemented by the CSIR

NATIONAL INTEGRATED CYBERINFRASTRUCTURE SYSTEM
CHPC

GPU performance



Labour intensive GPU porting can have a dramatic impact



Conclusions

- The world must change - quickly, smartly, comprehensively and profoundly
- The physical world is multi-disciplinary and multi-physics
- Modelling, simulation, coding and high-performance computing will be integral parts of this process
- A large part of the challenge is learning how best to tie in with what already exists

Some practical examples

NATIONAL INTEGRATED CYBERINFRASTRUCTURE SYSTEM
CHPC

File formats:

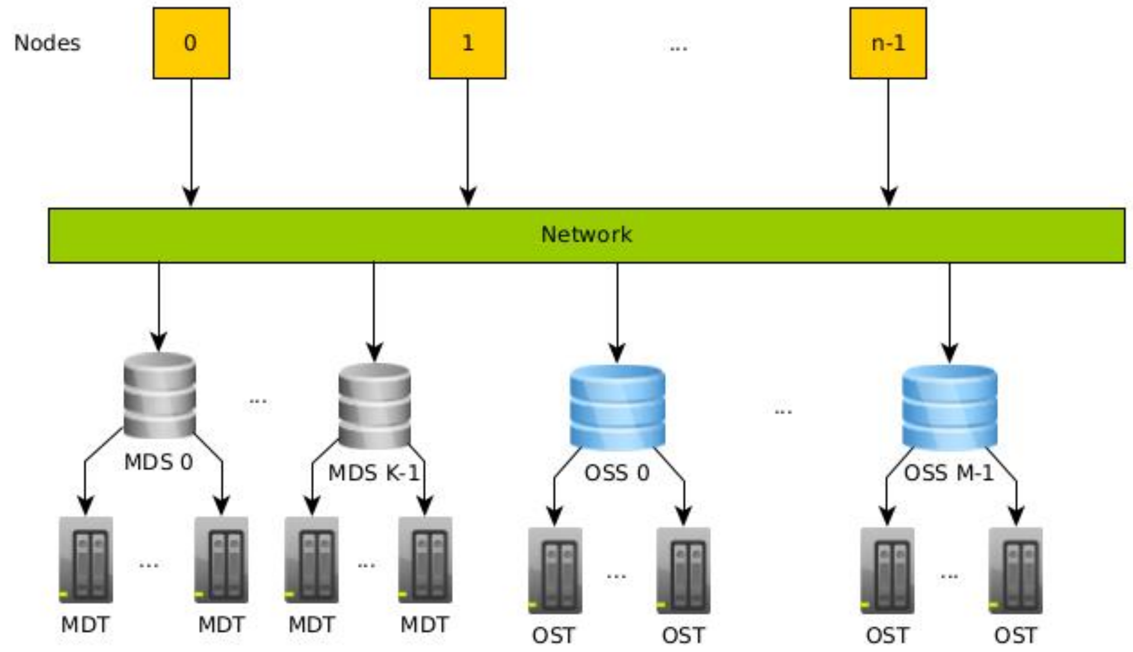
- ASCII or csv …. (useful for debugging & small amounts)
- Unformatted …(better know what's in the file, and how it was written)
- Self-documenting formats:
    - hdf5
    - netCDF
    - cgns
    - …. And others, as well as proprietary formats

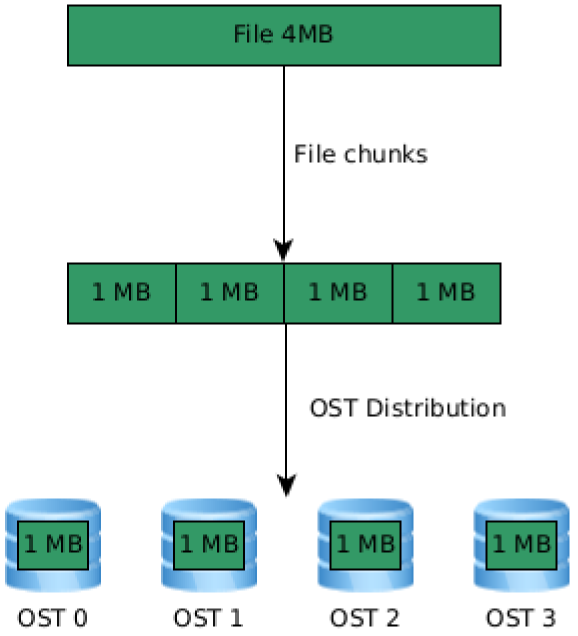
```
! Add some notes and and units to the file
    status = nf90_put_att(ncid, NF90_GLOBAL, 'note', 'Testing file, consisting of x,y,z and p, in F90')
    status = nf90_put_att(ncid, varid_x, 'units', 'm')
    status = nf90_put_att(ncid, varid_y, 'units', 'm')
    status = nf90_put_att(ncid, varid_z, 'units', 'm')
    status = nf90_put_att(ncid, varid_p, 'units', 'Pa')
```

```
! Store the variables in the file and time the operation
   timestart = omp_get_wtime()
   status = nf90_put_var(ncid, varid_x, x)
   status = nf90_put_var(ncid, varid_y, y)
   status = nf90_put_var(ncid, varid_z, z)
   status = nf90_put_var(ncid, varid_p, p)
! Close the file and check that it has been closed before proceeding
   status = nf90_close(ncid)
   call check(status, 'close')
   timeend = omp_get_wtime()
```

Parallel I/O

HPC systems are necessarily parallel, and the same applies to the file system, although to the user it looks like a Posix file system





Parallel I/O

How can we use this parallelism?

- Simple embarrassingly parallel, like OpenFOAM – case is decomposed into separate directories, and each MPI rank has its own directory
- This is just about clunky enough to make an excellent stress benchmark!

```
[charles@cnode0069:~/lustre/SimpleBenchMarkLarge2006]$ ls
0              processor16  processor27  processor38  processor49  processor6   processor70  processor81  processor92        runSimpleLarge90nX12
bananas.foam   processor17  processor28  processor39  processor5   processor60  processor71  processor82  processor93        SimpleBenchMarkLarge.OpenFOAM
constant       processor18  processor29  processor4   processor50  processor61  processor72  processor83  processor94        simple.out
decompose.out  processor19  processor3   processor40  processor51  processor62  processor73  processor84  processor95        stderr
processor0     processor2   processor30  processor41  processor52  processor63  processor74  processor85  runPost            stdout
processor1     processor20  processor31  processor42  processor53  processor64  processor75  processor86  runSimpleLarge10n  system
processor10    processor21  processor32  processor43  processor54  processor65  processor76  processor87  runSimpleLarge2n
processor11    processor22  processor33  processor44  processor55  processor66  processor77  processor88  runSimpleLarge30n
processor12    processor23  processor34  processor45  processor56  processor67  processor78  processor89  runSimpleLarge4n
processor13    processor24  processor35  processor46  processor57  processor68  processor79  processor9   runSimpleLarge60n
processor14    processor25  processor36  processor47  processor58  processor69  processor8   processor90  runSimpleLarge75n
processor15    processor26  processor37  processor48  processor59  processor7   processor80  processor91  runSimpleLarge90n
```

Parallel I/O

- Portable self-documenting file format with parallelization
- Available in hdf5 and NetCDF
- NetCDF-4 builds on top of hdf5, but no benefit from putting parallel netCDF on top of parallel hdf5

Parallel hdf5: `./configure --enable-parallel --disable-cxx`

Parallel NetCDF: First install pnetcdf, then NetCDF with:

```
./configure --enable-pnetcdf
```

Parallel NetCDF in WRF

NICIS
NATIONAL INTEGRATED CYBERINFRASTRUCTURE SYSTEM
CHPC

- Compile WRF with parallel NetCDF support
- Changes in namelist:

```
 io_form_history                     = 11
 io_form_restart                     = 11
 io_form_input                       = 2
 io_form_boundary                    = 11
 debug_level                         = 0
 nocolons                            = T
```

```
numtiles                            = 12
nproc_x                             = 10
nproc_y                             = 18
```

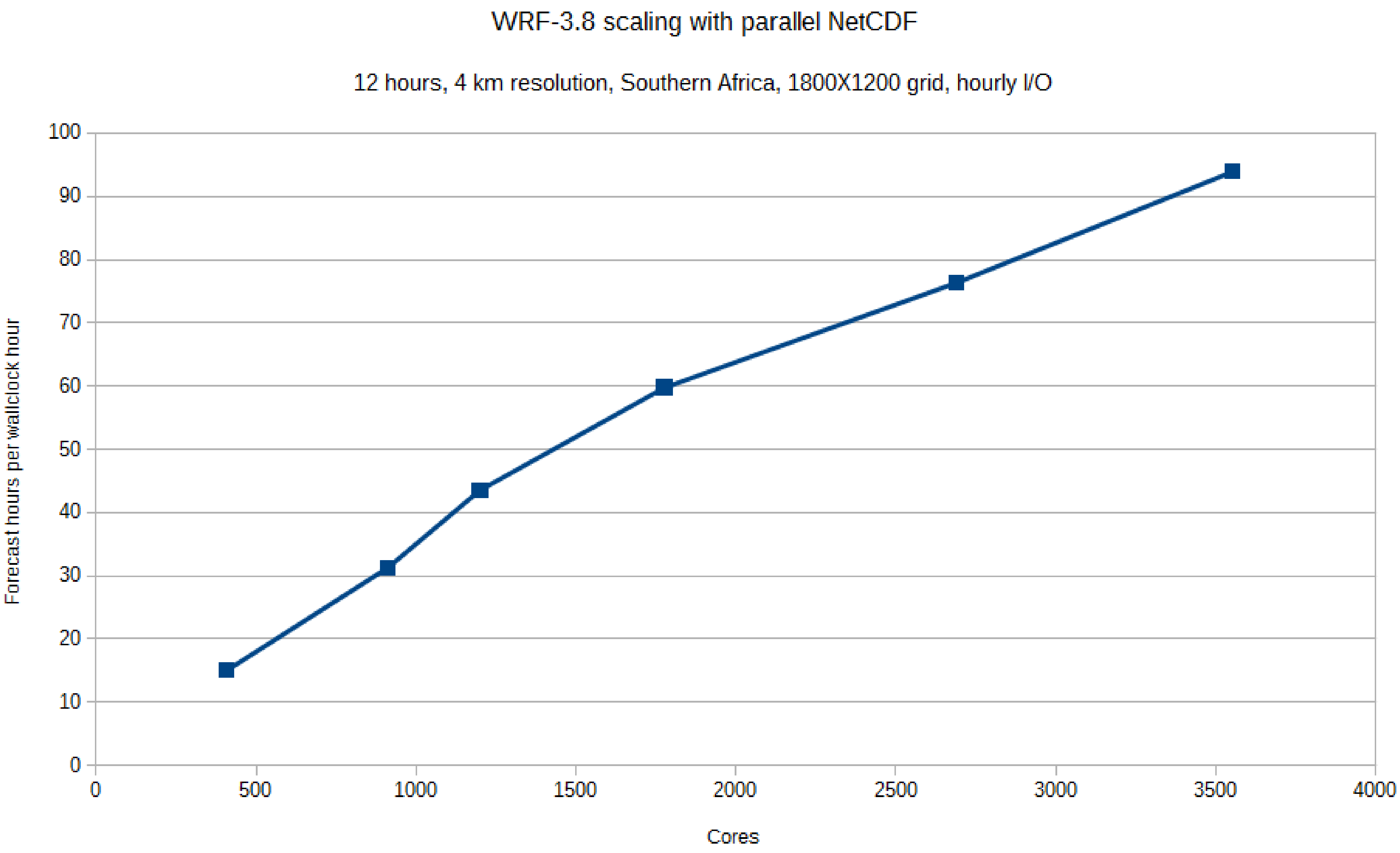
```
&namelist_quilt
 nio_tasks_per_group            = 6,
 nio_groups                     = 2,
 /
```

- Set aside `(nio_tasks_per_group * nio_groups)` MPI ranks to write outfiles

```
Timing for main: time 2016-02-22_01:00:00 on domain   1:    3.16576 elapsed seconds
Timing for Writing wrfout_d01_2016-02-22_01_00_00 for domain        1:    5.60983 elapsed seconds
Timing for Writing wrfout_d02_2016-02-22_01_00_00 for domain        2:    5.79784 elapsed seconds
Timing for main: time 2016-02-22_01:00:15 on domain   2:    6.50131 elapsed seconds
```

```
Timing for main: time 2016-02-22_01:00:00 on domain   1:    3.01744 elapsed seconds
Timing for Writing wrfout_d01_2016-02-22_01_00_00 for domain        1:    0.28624 elapsed seconds
Timing for Writing wrfout_d02_2016-02-22_01_00_00 for domain        2:    0.39313 elapsed seconds
Timing for main: time 2016-02-22_01:00:15 on domain   2:    1.02392 elapsed seconds
```

Parallel NetCDF in WRF



A national initiative of the Department of Science and Innovation and implemented by the CSIR

Practical visualization

NATIONAL INTEGRATED CYBERINFRASTRUCTURE SYSTEM
CHPC

- Many different ways, depends on user, research group, software, purpose, etc …
- We have people using Jupyter, pandas, matplotlib, scilab, octave, gnuplot and xplot, proprietary applications, ParaView, VisIt, Ensight, etc.
- Some users simply download results to laptop and use desktop tools there
- VNC is useful
- Paraview in client/server mode
- Parallel Mesa library