THE MITGCM USER'S GUIDE TO ARCHER2

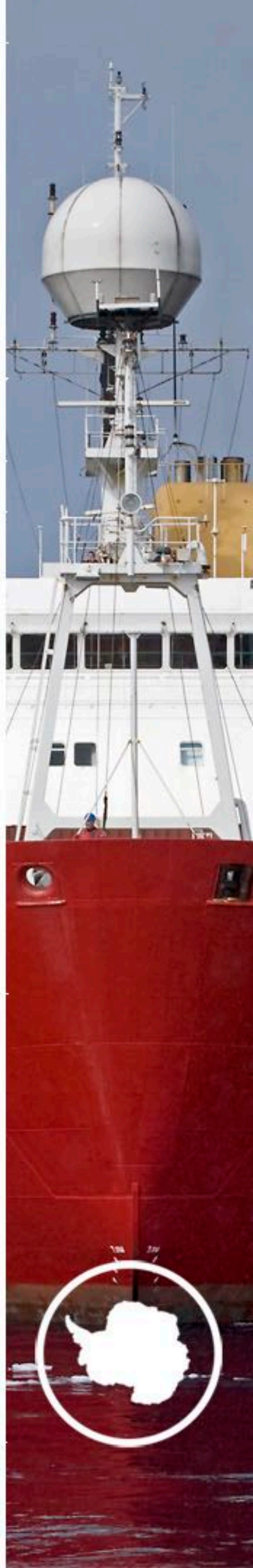
Mike Mineter[1], Emma Boland[2], Dan(i) Jones[2], Kaitlin Naughten[2], Dan Goldberg[1]

1: University of Edinburgh, 2: British Antarctic Survey

POLAR SCIENCE FOR PLANET EARTH

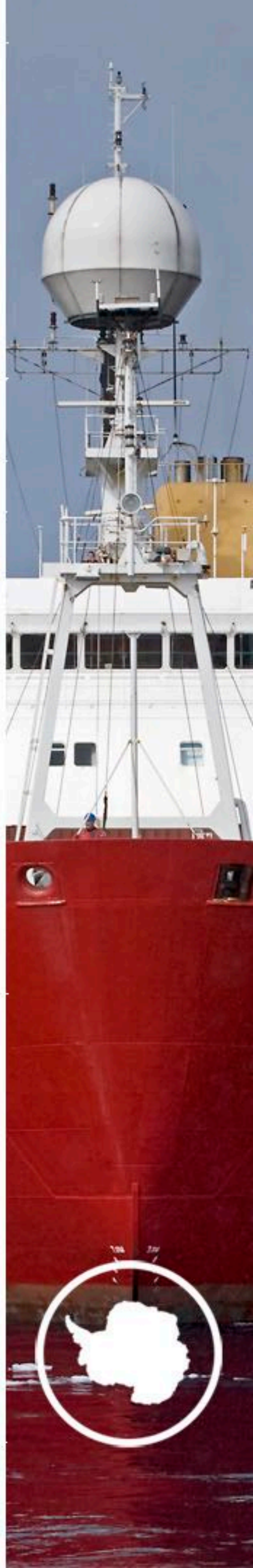
BACKGROUND

- Based on work by Mike Mineter, University of Edinburgh, under eCSE02-06 funded by EPCC
- ARCHER2 “embedded CSE” (eCSE) projects are funded through regular calls to develop sustainable software and improve research on ARCHER2
- Provide funds for a research software engineer - can be provided by the CSE team or from your/other institute.
- eCSE02-06 was titled “Optimising MITgcm on ARCHER2” and provided funding to install and optimised MITgcm
- Find out more at www.archer2.ac.uk/ecse/



OUTLINE

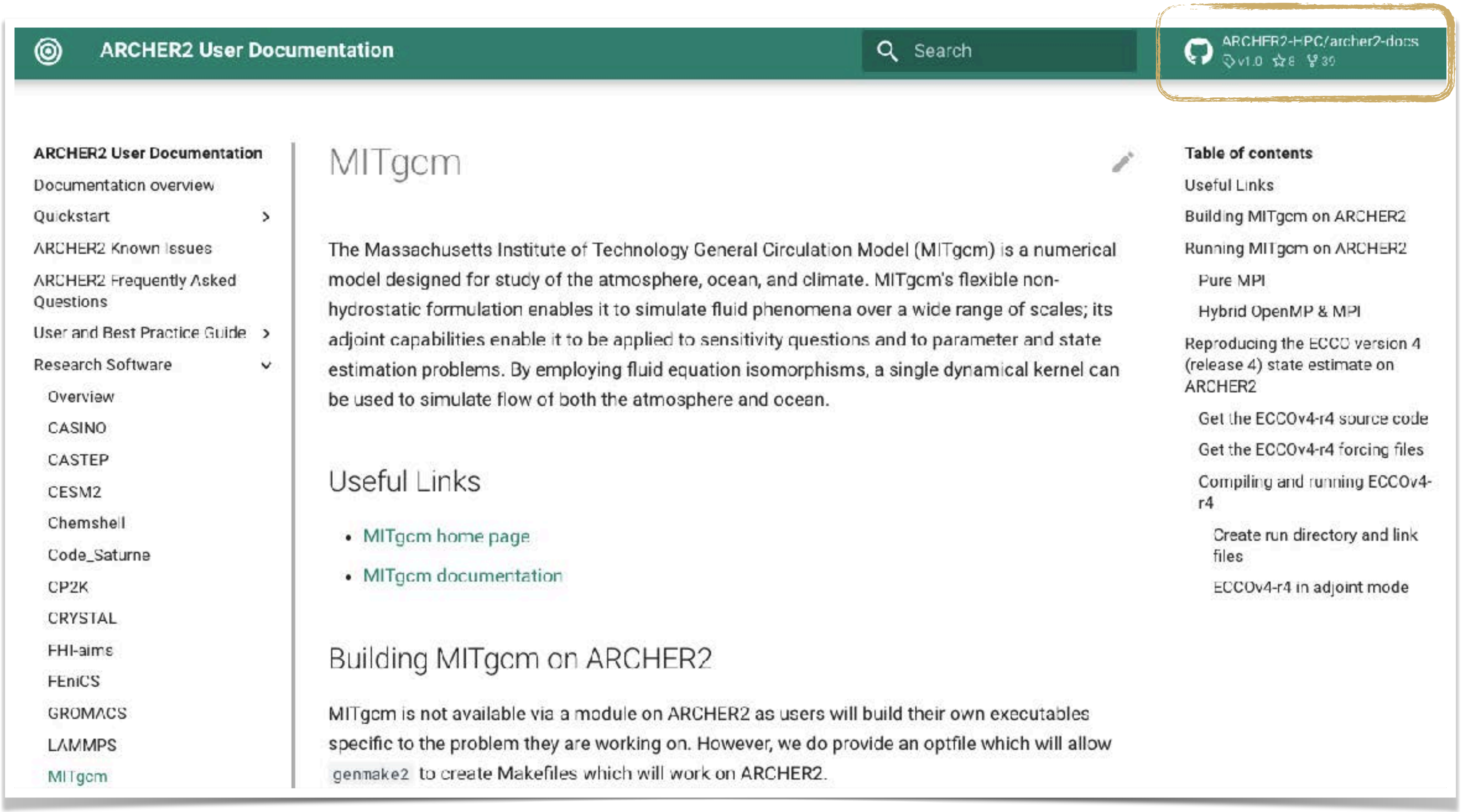
- Getting started with MITgcm on ARCHER2
  - Installing
  - Running
- Tips and tricks for optimal running
  - Cheap vs fast runs
- Advanced usage:
  - Job chaining for long runs
  - Adjoint models
  - Containerisation for bundling extra code
- OptClim - parameter optimisation for climate models

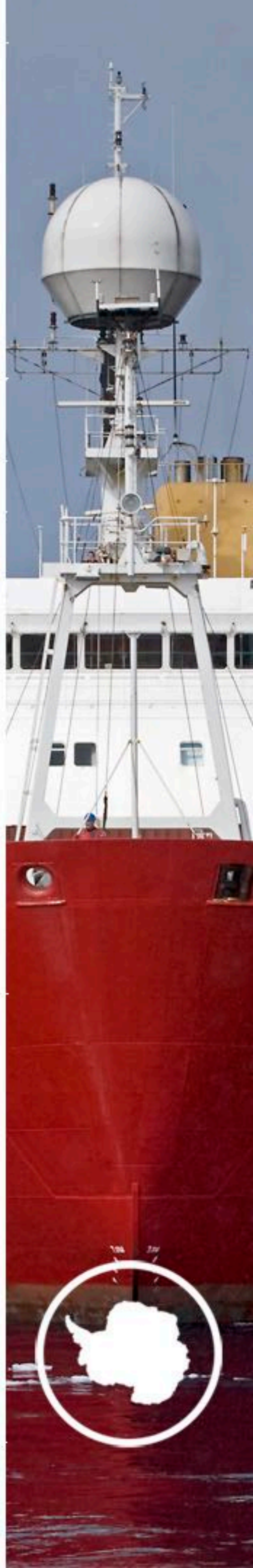


GETTING STARTED WITH MITGCM ON ARCHER2

- All this and more is covered in the ARCHER2 documentation: docs.archer2.ac.uk/research-software/mitgcm/

ARCHER2 documentation is open source - fork it from GitHub and add your own useful info!

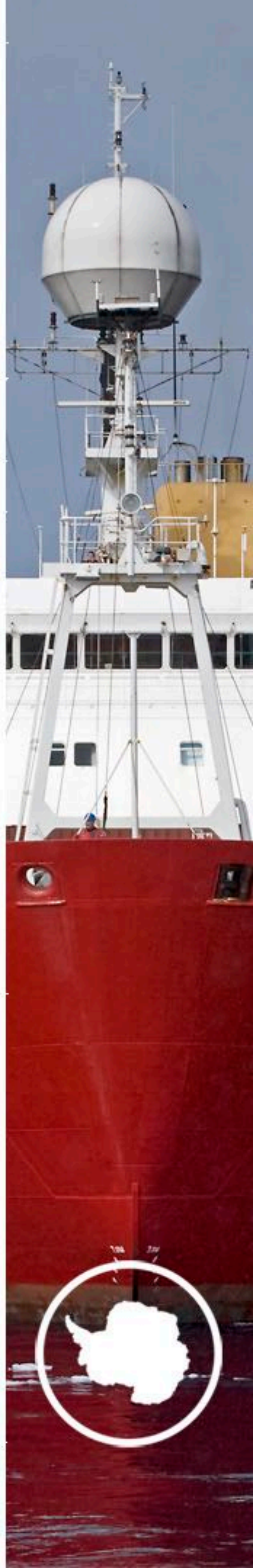




GETTING STARTED WITH MITGCM ON ARCHER2

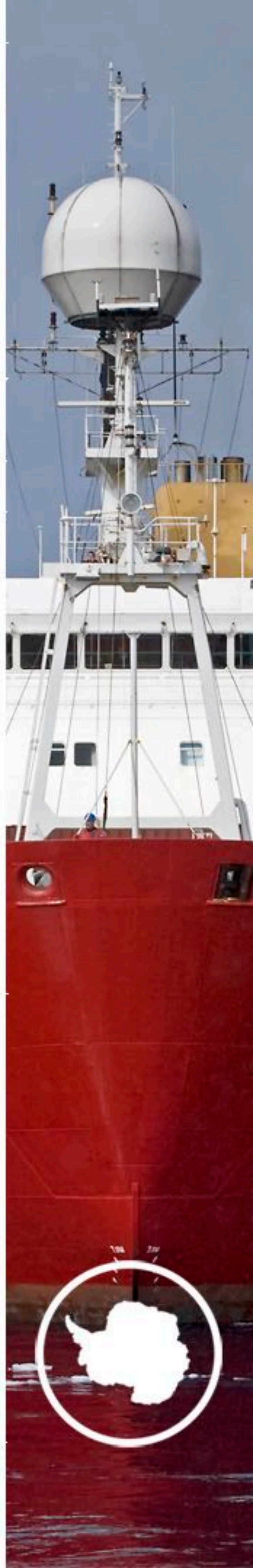
- All this and more is covered in the ARCHER2 documentation:
  docs.archer2.ac.uk/research-software/mitgcm/
- Install MITgcm in your **home** directory, e.g. /home/n01/n01/emmomp
- Best practice is to clone from git:
  "git clone https://github.com/MITgcm/MITgcm.git"
- Compile using our opt file "dev_linux_amd64_cray_archer2"

```
[emmomp@ln03:~> ls MITgcm/tools/build_options/
bgl_gnu_ncar            darwin_ppc_xlf_panther_baylor     linux_amd64_gfortran_greenplanet   linux_amd64_pathf90
bgl_ncar                darwin_ppc_xlf_panther+wienders   linux_amd64_ifort                  linux_amd64_pathf90+redhatlam
cygwin_ia32_g77         darwin_ppc_xlf_tiger_baylor       linux_amd64_ifort11                linux_amd64_pgf77
darwin_absoft_f77       dev_linux_amd64_cray_archer2      linux_amd64_ifort_beagle           linux_amd64_pgf77+mpi_ncar
darwin_amd64_gfortran   dev_linux_amd64_gfortran_archer2  linux_amd64_ifort_discover         linux_amd64_pgf77+mpi_xd1
```



GETTING STARTED WITH MITGCM ON ARCHER2

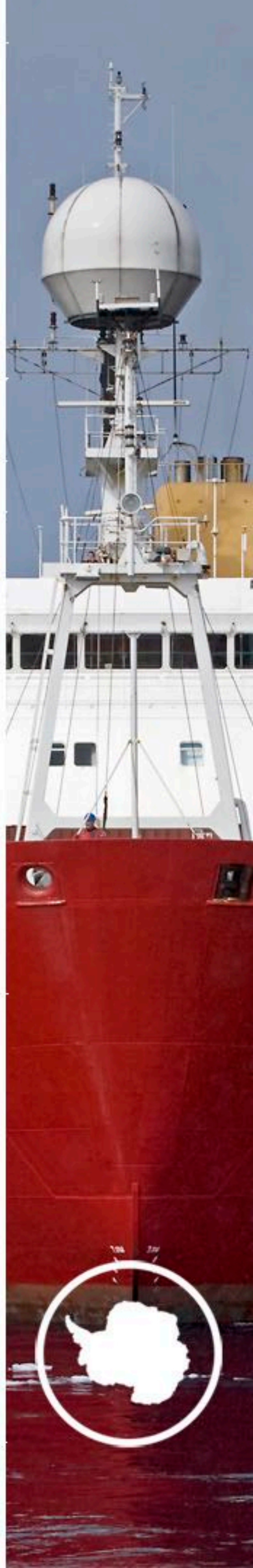
- All this and more is covered in the ARCHER2 documentation: docs.archer2.ac.uk/research-software/mitgcm/
- Install MITgcm in your **home** directory, e.g. /home/n01/n01/emmomp
- Best practice is to clone from git:
  “git clone https://github.com/MITgcm/MITgcm.git"
- Compile using our opt file “dev_linux_amd64_cray_archer2”
- Remember to specify the number of processors (nPx/y) in SIZE.h **before** compiling!



GETTING STARTED WITH MITGCM ON ARCHER2

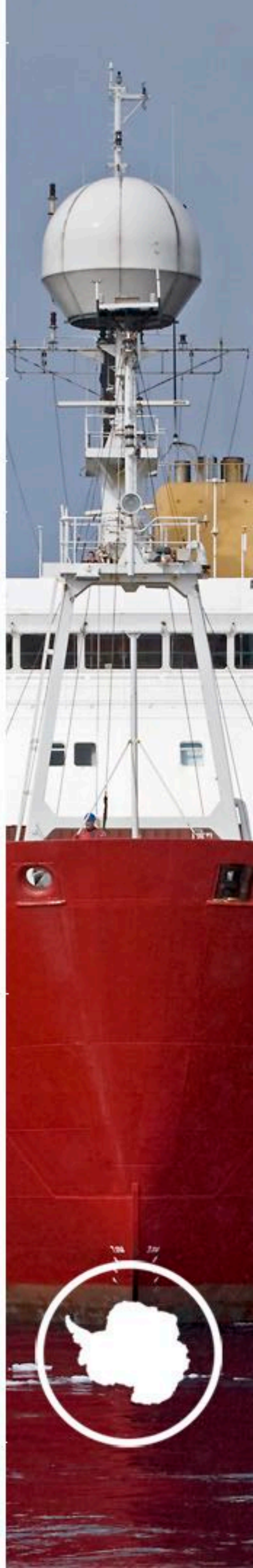
- Set up a folder in your **work** directory with required namelists, inputs etc.
- If you're using large input files, place them in a shared directory like */work/n01/shared* so they don't end up downloaded multiple times.
- Copy over your executable (normally *mitgcmuv*) from your home directory.
- Create a submission script specifying options like names, nodes, time, queues etc. See examples in the ARCHER2 documentation:

```
#!/bin/bash

# Slurm job options (job-name, compute nodes, job time)
#SBATCH --job-name=MITgcm-simulation
#SBATCH --time=1:0:0
#SBATCH --nodes=2
#SBATCH --tasks-per-node=128
#SBATCH --cpus-per-task=1

# Replace [budget code] below with your project code (e.g. t01)
#SBATCH --account=[budget code]
#SBATCH --partition=standard
#SBATCH --qos=standard

# Set the number of threads to 1
#   This prevents any threaded system libraries from automatically
#   using threading.
```



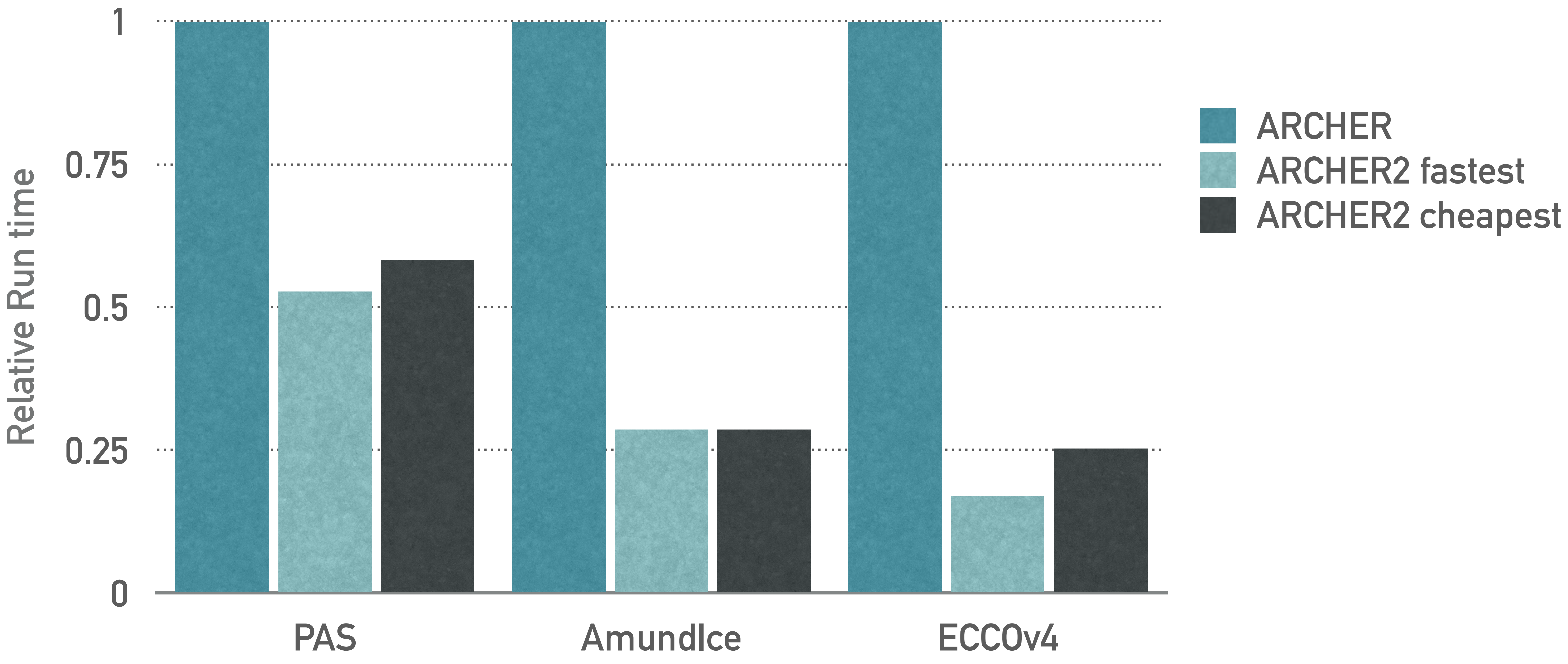
GETTING STARTED WITH MITGCM ON ARCHER2

- Set up a folder in your **work** directory with required namelists, inputs etc.
- If you're using large input files, place them in a shared directory like */work/n01/shared* so they don't end up downloaded multiple times.
- Copy over your executable (normally *mitgcmuv*) from your home directory.
- Create a submission script specifying options like names, nodes, time, queues etc. See examples in the ARCHER2 documentation.
- Submit using a command like "*sbatch run_script.slurm*"
- Aaaand Relax(!)

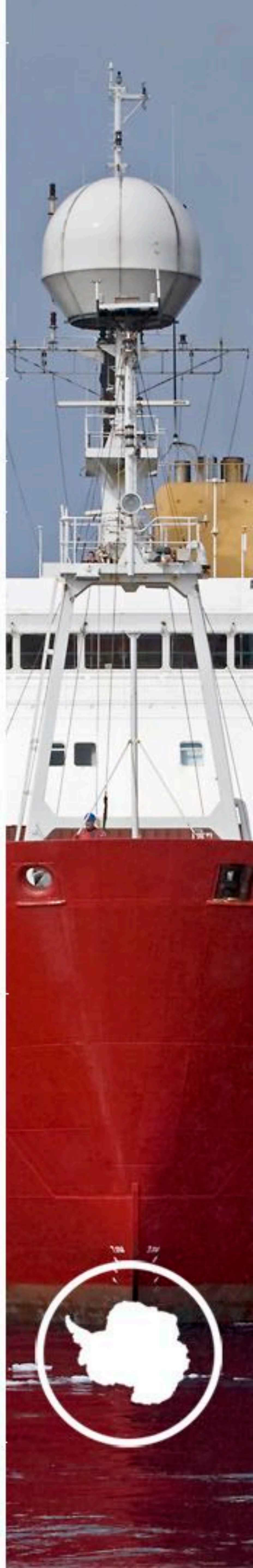


TIPS AND TRICKS FOR RUNNING

- ARCHER2 is fast!

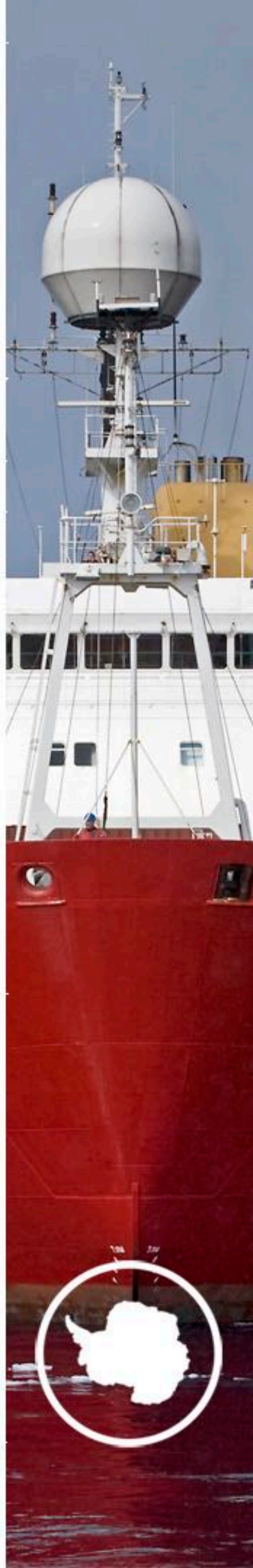


Data courtesy of M Mineter from eCSE02-06



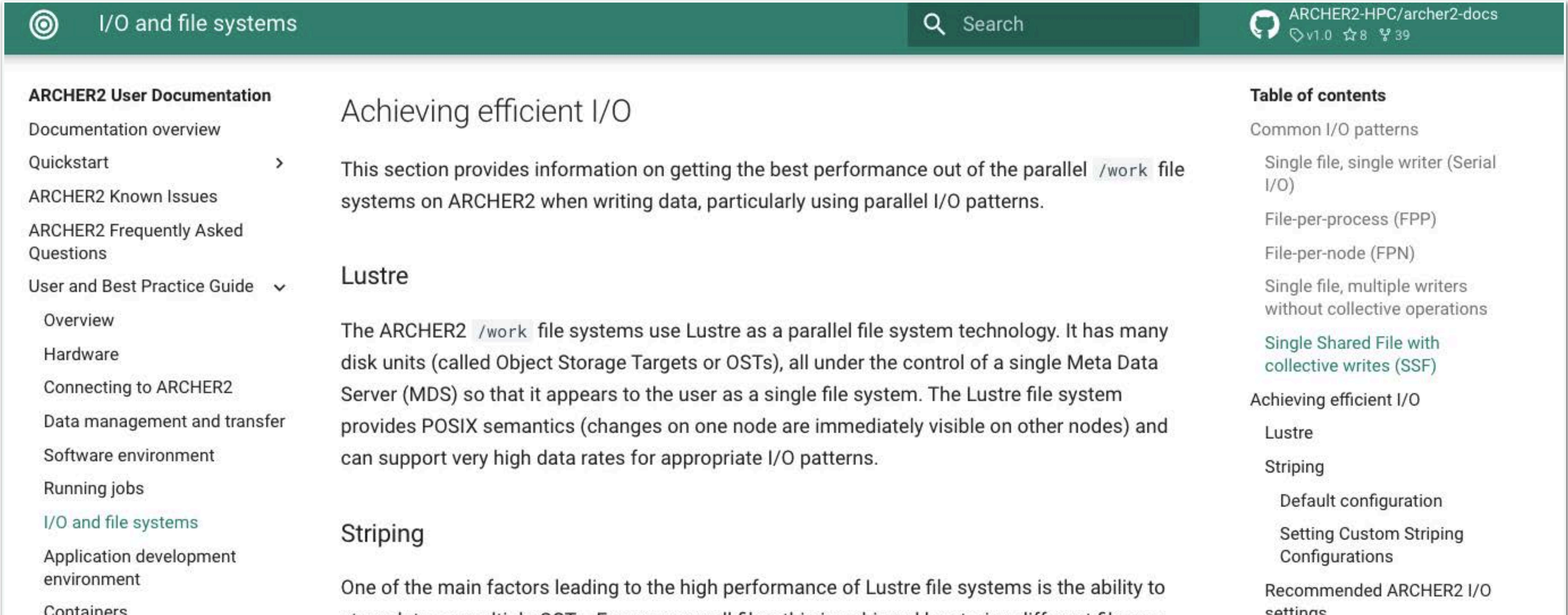
TIPS AND TRICKS FOR RUNNING

- ARCHER2 is fast!
- How fast depends on particular cases - we saw speed ups of 2-5 x over ARCHER
- **Fastest** runs were with **more** nodes…
- …but beware of diminishing returns
- **Cheapest** runs were with **fewest** nodes
- Our options file is optimised for the test cases here, if you have a particularly complicated setup it may be worth experimenting…
- We found the Cray compiler most reliable but we also have a Gnu options file, just ask…



OPEN QUESTIONS

- Our study focused on runtime efficiency.
- I/O options remain unexplored - see the ARCHER2 documentation for some guidance.
- Please share with the community if you have useful tips!



I/O and file systems

Search

ARCHER2-HPC/archer2-docs
v1.0 8 39

ARCHER2 User Documentation

Documentation overview
Quickstart
ARCHER2 Known Issues
ARCHER2 Frequently Asked Questions
User and Best Practice Guide
- Overview
- Hardware
- Connecting to ARCHER2
- Data management and transfer
- Software environment
- Running jobs
- I/O and file systems
- Application development environment
- Containers

Achieving efficient I/O

This section provides information on getting the best performance out of the parallel `/work` file systems on ARCHER2 when writing data, particularly using parallel I/O patterns.

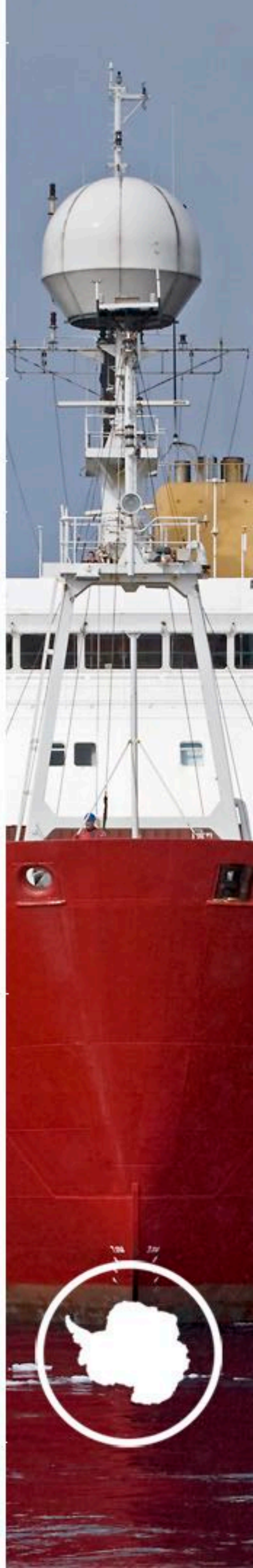
Lustre

The ARCHER2 `/work` file systems use Lustre as a parallel file system technology. It has many disk units (called Object Storage Targets or OSTs), all under the control of a single Meta Data Server (MDS) so that it appears to the user as a single file system. The Lustre file system provides POSIX semantics (changes on one node are immediately visible on other nodes) and can support very high data rates for appropriate I/O patterns.

Striping

One of the main factors leading to the high performance of Lustre file systems is the ability to

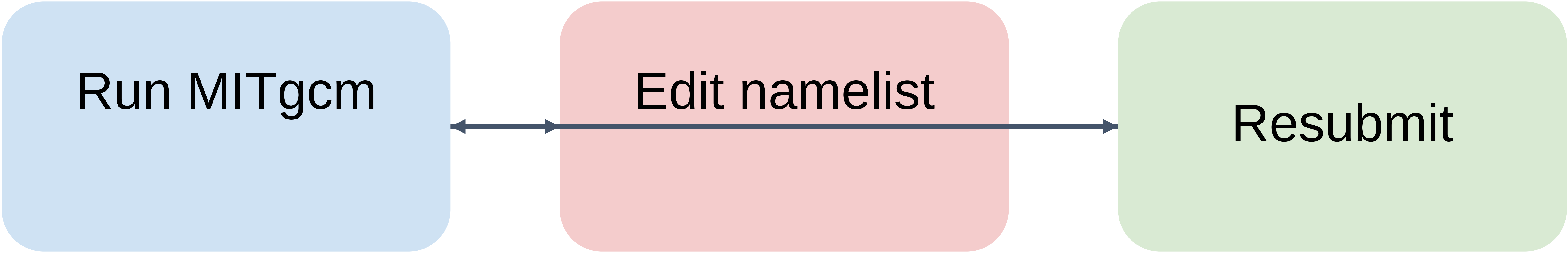
Table of contents



ADVANCED USAGE

Job chaining: motivation

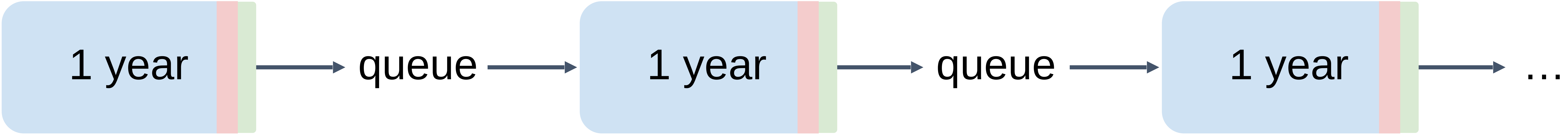
What if your simulation needs more than the maximum walltime? (24h)



When do you decide to stop and restart?

Job chaining: old approach

Run a set length of simulation (eg 1 year) for each segment



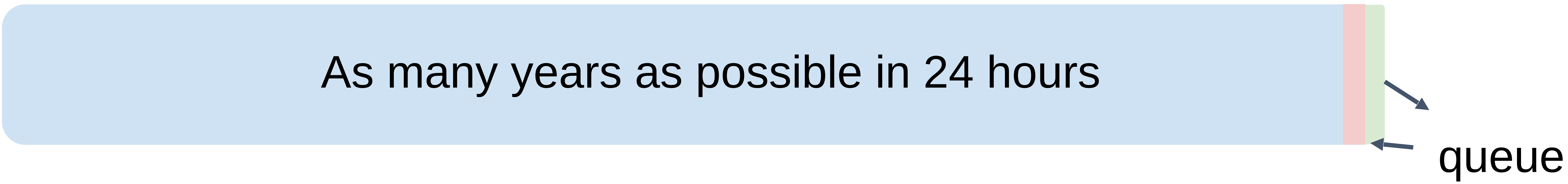
Disadvantages:

- Have to overestimate walltime for each segment
- Run risk of not finishing a segment in time
- More time spent on model initialisation
- Queue more often, so more opportunity for bottlenecks

Job chaining: new approach

Request maximum queue length and run MITgcm for as long as possible

Stop a few minutes before the end (**timeout** utility), identify latest restart point, edit namelist, resubmit



Makes most efficient use of the queue, and reduces guesswork for user

Job chaining: code structure

https://github.com/knaughten/UaMITgcm/tree/archer2/example/PAS_999/mitgcm_run/scripts/standalone_mit

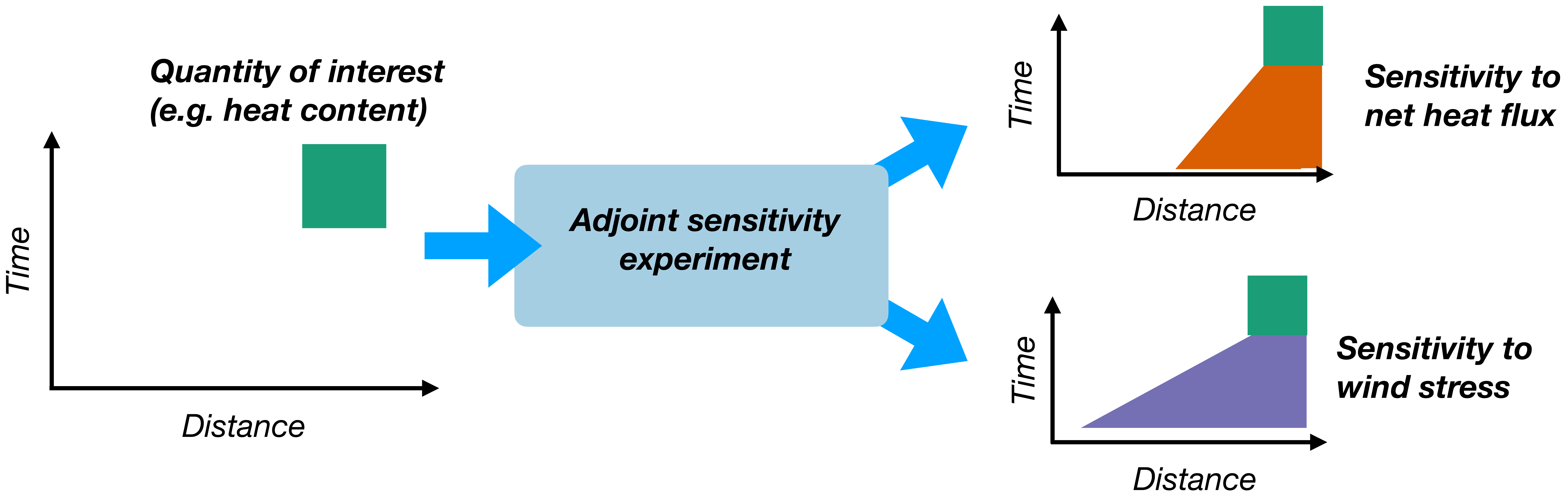
user runs: **sub_run.sh**

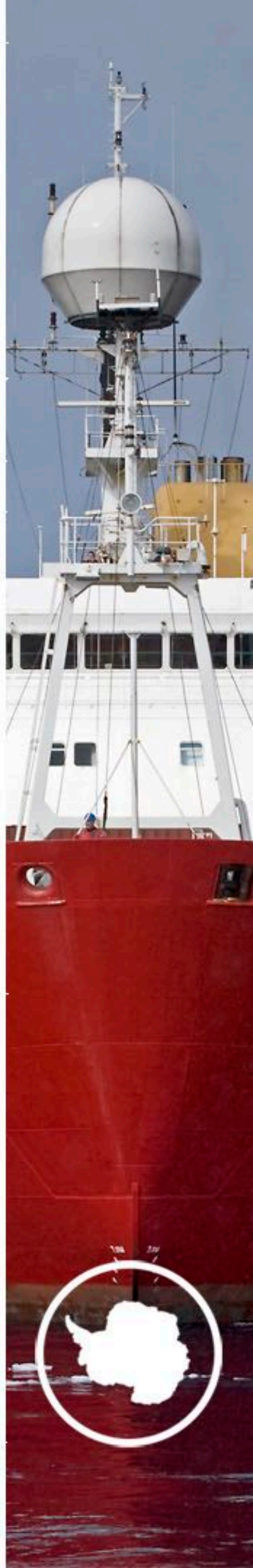
code submits: **run_repeat.slurm**

code submits: **netcdf_out.slurm**
(relies on **netcdf_out.py**)

ADVANCED USAGE – ADJOINT MODELS

- What is an adjoint model?
- Unlike a traditional sensitivity experiment (perturb then run fwd to see what happens), adjoints run in reverse:

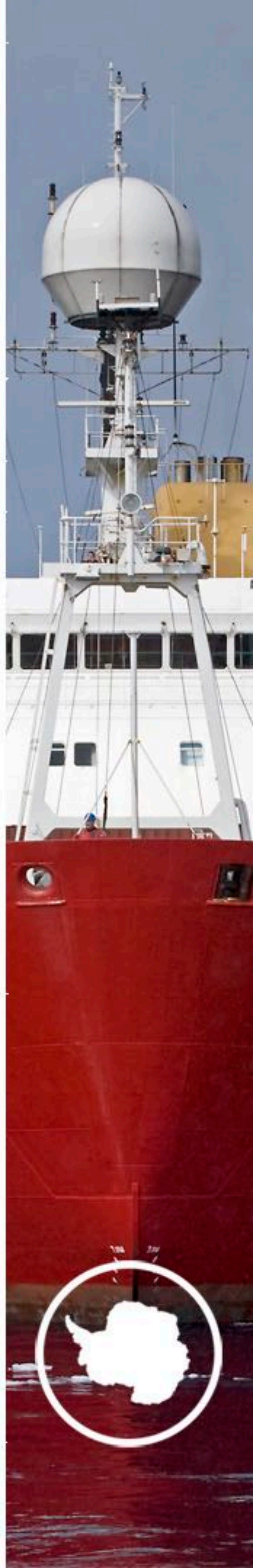




ADVANCED USAGE – ADJOINT MODELS

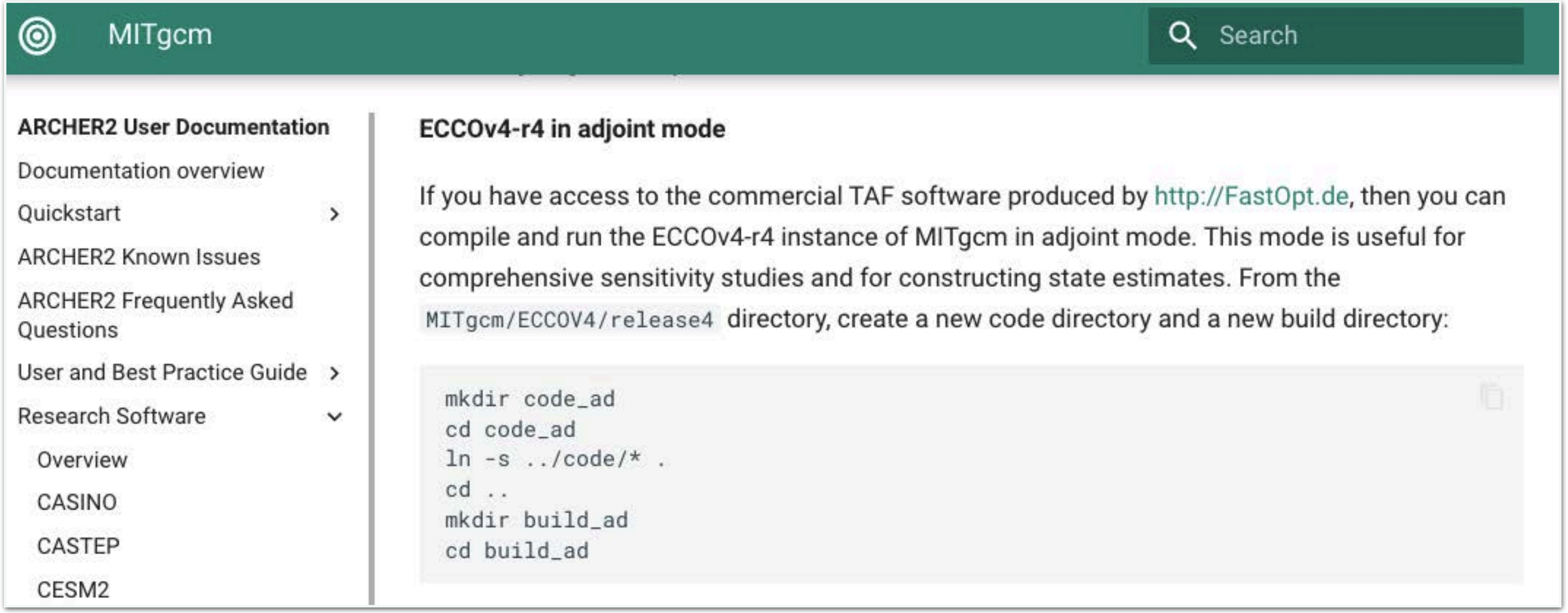
Software to auto-differentiate your model:

- OpenAD:
  - Open Source
  - Harder to use off the shelf
- TAF:
  - Requires a license
  - Works reliably
  - Used for the ECCOv4 state estimate



ADVANCED USAGE – ADJOINT MODELS

➤ Follow the ARCHER2 documentation for generating the ECCOv4 adjoint, a state estimate which uses the adjoint sensitivities to minimise model-observation differences.



MITgcm

Search

ARCHER2 User Documentation

Documentation overview

Quickstart

ARCHER2 Known Issues

ARCHER2 Frequently Asked Questions

User and Best Practice Guide

Research Software

Overview

CASINO

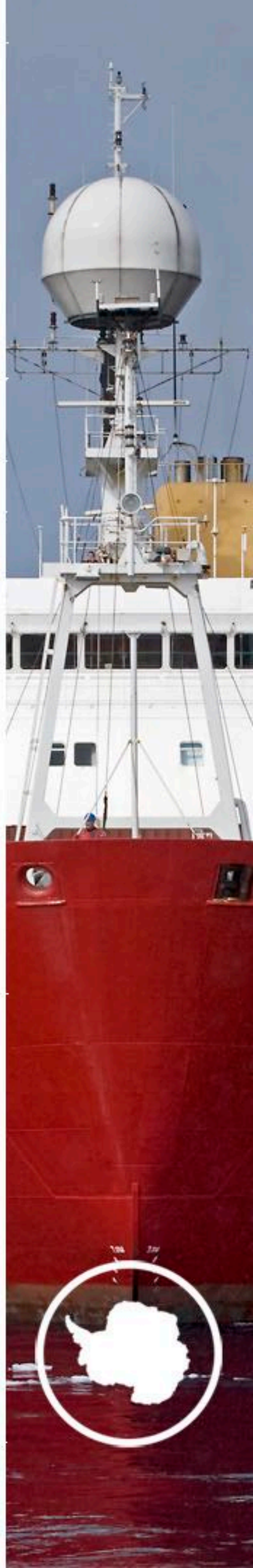
CASTEP

CESM2

ECCOv4-r4 in adjoint mode

If you have access to the commercial TAF software produced by http://FastOpt.de, then you can compile and run the ECCOv4-r4 instance of MITgcm in adjoint mode. This mode is useful for comprehensive sensitivity studies and for constructing state estimates. From the `MITgcm/ECCOV4/release4` directory, create a new code directory and a new build directory:

```
mkdir code_ad
cd code_ad
ln -s ../code/* .
cd ..
mkdir build_ad
cd build_ad
```

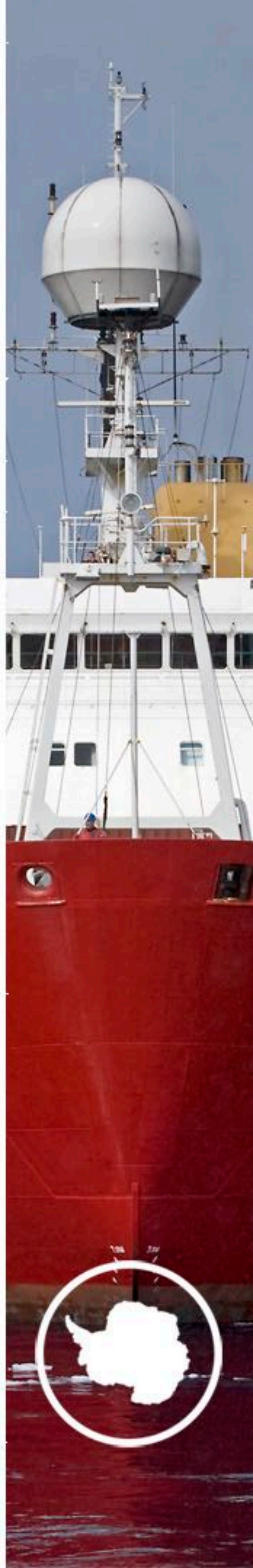


ADVANCED USAGE – ADJOINT MODELS

- Instead of “make all” -> “make adtaf” then “make adall”
- Creates *mitgcmuv_ad* instead of *mitgcmuv*
- Turn on the ECCO package for calculating your cost function/QoI
- Turn on the CTRL package to generate sensitivities
- Define your QoI in *data.ecco* following the MITgcm documentation:

Table 10.5 Implemented gencost_barfile options (as of checkpoint 67x) that can be used via cost_gencost_boxmean.F (Section 10.1.2).

variable name	description	remarks
`m_boxmean_theta`	mean of theta over box	specify box
`m_boxmean_salt`	mean of salt over box	specify box
`m_boxmean_eta`	mean of SSH over box	specify box
`m_boxmean_shifwf`	total shelfice freshwater flux over box	specify box
`m_boxmean_shihf`	total shelfice heat flux over box	specify box
`m_horflux_vol`	volume transport through section	specify transect



ADVANCED USAGE – ADJOINT MODELS

- Everything is documented in **section 10** of the MITgcm documentation
- There is a dedicated ECCO mailing list that can help with problems

See ecco-group.org

Introduction to Containers: an MITgcm user perspective

Dan Goldberg
School of GeoSciences
University of Edinburgh

Overview

- Singularity containers – from a User perspective
- Use Cases:
  - Open-Source algorithmic differentiation tool
  - Visualisation on ARCHER2
  - MITgcm/PETSc on rack server

What are containers?

- Virtual Machines – Virtualise the *HARDWARE*
- Containers – Virtualise the *OPERATING SYSTEM* but interact with physical file server

- *Why?*
  - You might not have root privileges on the machine you are using
  - You might not know how to install software in the target environment, even if you can do so on your local computer
  - The target environment may not have libraries that you need
  - Reproducibility!

Docker versus Singularity

- Singularity images are *files* which can be transferred, moved, etc – while Docker images must be in a central location
- Singularity is "closer to the hardware" so may yield better performance
- Singularity is *more secure* than Docker

But

- Docker is more mature, with better documentation/support (Singularity cannot be easily installed on Mac/Windows)

Use Case -- OpenAD

- OpenAD is an *open-source, source-to-source* AD tool from Argonne National Laboratory
  - Has been applied to MITgcm – not used as extensively
  - Adjoint performance is not great – *except with* STREAMICE

Problem: OpenAD is no longer updated, and binaries cannot be compiled with gcc 5.X or later!

Note – this is distinct from using it to generate compilable source code

ARCHER2 administrators were asked to install – recommended singularity

Use Case -- OpenAD

- Strategy:
  - Compile OpenAD executables in a *Singularity* container (.sif file). *Requires Singularity installation on "home" computer, along with .def file, and "user group" permissions*
  - Build MITgcm-OpenAD adjoint on ARCHER2, calling OpenAD in container when necessary. *Requires singularity installation and .sif file.*
  - MITgcm executable is NOT compiled in container – and hence is not run in a container.

Use Case -- OpenAD

- Resources:
  - https://www.archer2.ac.uk/training/courses/220119-containers/
  - https://docs.sylabs.io/guides/3.5/user-guide/definition_files.html
  - https://mitgcm.readthedocs.io/en/latest/autodiff/autodiff.html#adjoint-code-generation-using-openad
  - Dr Magnus Hagdorn, School of GeoSciences

```
Bootstrap: library
From: ubuntu:18.04
Stage: build

%setup
    touch /file1
    touch ${SINGULARITY_ROOTFS}/file2

%files
    /file1
    /file1 /opt

%environment
    export LISTEN_PORT=12345
    export LC_ALL=C

%post
    apt-get update && apt-get install -y netcat
    NOW=`date`
    echo "export NOW=\"${NOW}\"" >> $SINGULARITY_ENVIRONMENT

%runscript
    echo "Container was created $NOW"
    echo "Arguments received: $*"
    exec echo "$@"

%startscript
    nc -lp $LISTEN_PORT
```

```
# canonicalizer
ad_input_code_sf.pre.f90 : \$(CB2M_AD_FILES)
        \${SINGULARITYCMD} \${OPENADFORTTK_BASE}/tools/SourceProcessing/preProcess.py --timing --r8 -H -S  -o \$@ \$^

# replace stop statements (to avoid the implied unstructured control flow)  with print statements
ad_input_code_sf.pre.s2p.f90 : ad_input_code_sf.pre.f90
        cat \$< | sed -f \$(OADTOOLS)/stop2print.sed > ad_input_code_sf.pre.s2p.f90

# F -> WHIRL
ad_input_code_sf.pre.s2p.B: ad_input_code_sf.pre.s2p.f90
        \${SINGULARITYCMD} \${OPEN64ROOT}/crayf90/sgi/mfef90 -r8 -z -F ad_input_code_sf.pre.s2p.f90

# WHIRL -> XAIF
ad_input_code_sf.pre.s2p.xaif : ad_input_code_sf.pre.s2p.B
        \${SINGULARITYCMD} \${OPENADFORTTK}/bin/whirl2xaif -s -n --debug 1 -o \$@ \$<

# XAIF -> XAIF'
ad_input_code_sf.pre.s2p.xb.xaif : ad_input_code_sf.pre.s2p.xaif xaif.xsd xaif_base.xsd xaif_inlinable_intrinsics.xsd xaif_deriv
        \${SINGULARITYCMD} \${XAIFBOOSTERROOT}/xaifBooster/algorithms/BasicBlockPreaccumulationReverse/driver/oadDriver -f -t fo
```

SINGULARITYCMD = singularity exec ${SINGULARITYFILE}

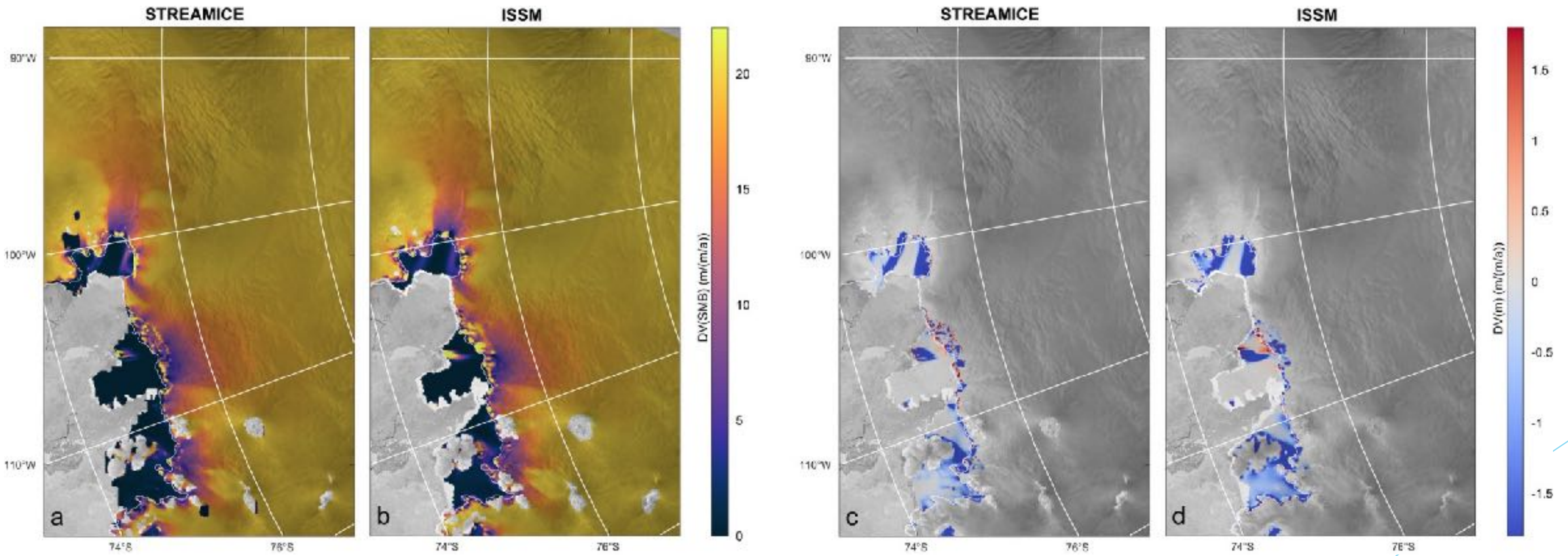
Use Case -- OpenAD

…. And then science!

Geophysical Research Letters

Research Letter | Full Access

Mapping the Sensitivity of the Amundsen Sea Embayment to Changes in External Forcings Using Automatic Differentiation



Other use cases: `display`

- I like to check progress of ARCHER2 batch by plotting diagnostics
- Problem – was unable to generate interactive figures with IPython on ARCHER2 login nodes (unable to set correct "backend" in matplotlib)
- Can save figures to file – but where is `display`???
- Correct solution: email helpdesk. My solution: install `display` in a container, move image to ARCHER2

```
dispfunc() {
        singularity exec -B $PWD display.sif display $1
}
```

Other use cases: MITgcm on rack server

- The catch – you need a native installation of MPI, and the image must be ”consistent” with the MPI installation (i.e. mpich→mpich)

```
singularity exec –B [file share] mitgcm.sif MITgcm/tools/genmake2  –mods=../code
–mpi=/opt/mpi
singularity exec –B [file share] mitgcm.sif make depend
singularity exec –B [file share] mitgcm.sif make –j 8

mpirun –n 24 singularity exec –B [file share] mitgcm.sif ./mitgcmuv
OR
singularity exec –B [file share] mitgcm.sif mpirun –n 24 ./mitgcmuv
```

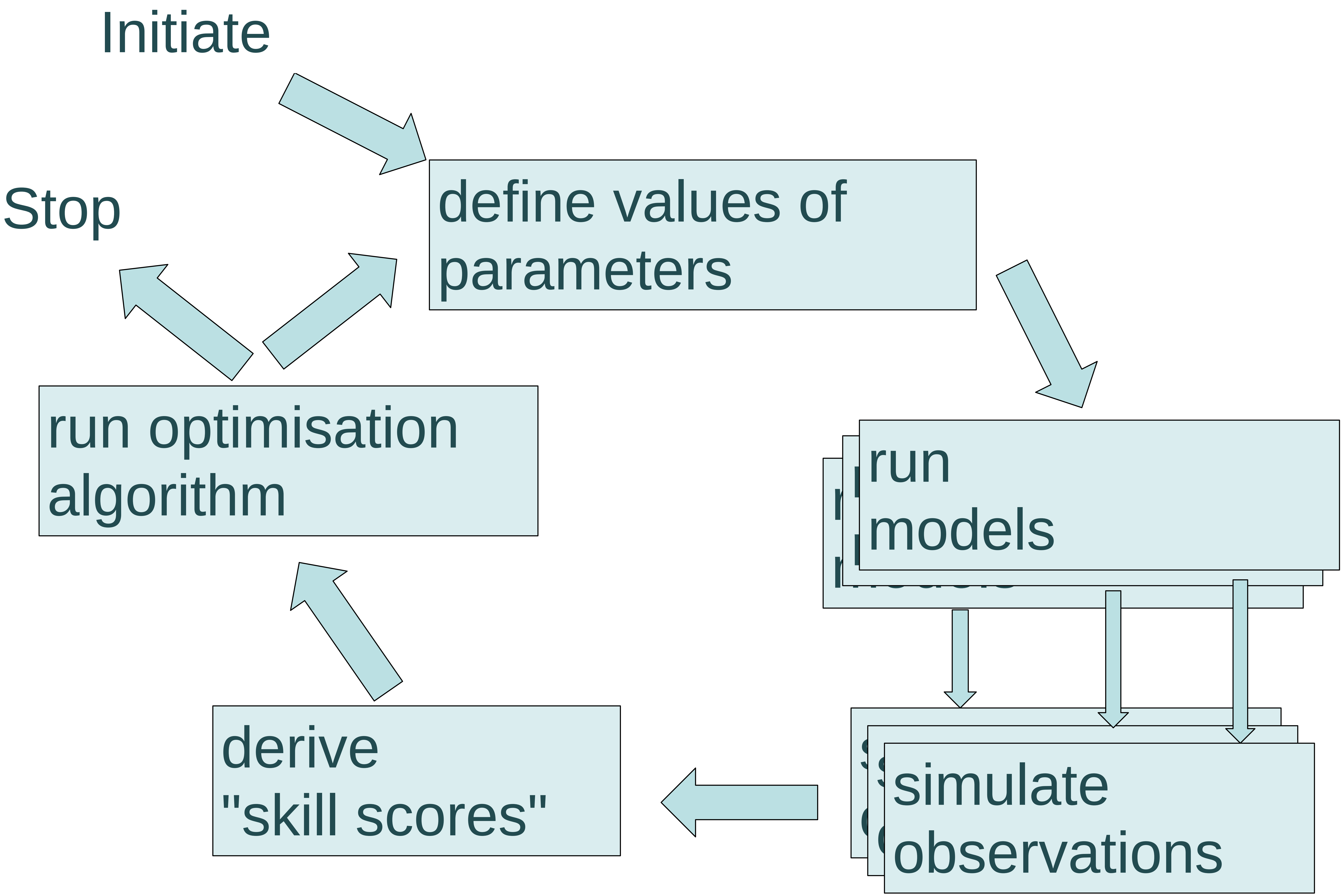
OptClim: software to optimise models

Mike Mineter, Simon Tett
GeoSciences, University of Edinburgh

Coralia Cartis
Mathematics, University of Oxford

m.mineter@ed.ac.uk

OptClim


Initiate
define values of parameters
run models
simulate observations
derive "skill scores"
run optimisation algorithm
Stop

OptClim

- Ported to ARCHER2
- Explores parameter values to optimise a model against observation
- Works with MITgcm, CESM, UKESM

- Ask for more information m.mineter@ed.ac.uk