

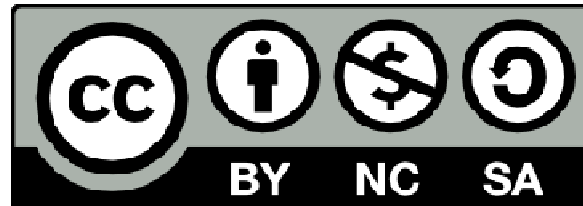
# Compiling for the ARCHER hardware

---

Slides contributed by Cray and EPCC



# Reusing this material



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

[http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en\\_US](http://creativecommons.org/licenses/by-nc-sa/4.0/deed.en_US)

This means you are free to copy and redistribute the material and adapt and build on the material under the following terms: You must give appropriate credit, provide a link to the license and indicate if changes were made. If you adapt or build on the material you must distribute your work under the same license as the original.

Note that this presentation contains images owned by others. Please seek their permission before reusing these images.



# Modules

- The Cray Programming Environment uses the GNU “modules” framework to support multiple software versions and to create integrated software packages
  - As new versions of the supported software and associated man pages become available, they are installed and added to the Programming Environment as a new version, while earlier versions are retained to support legacy applications
  - System administrators will set the default version of an application, or you can choose another version by using modules system commands
  - Users can create their own modules, or administrators can install site specific modules available to many users.



# Viewing the current module state

- Each login session has its own module state which can be modified by loading, swapping or unloading the available modules.
- This state affects the functioning of the compiler wrappers and in some cases runtime of applications.
- A standard, default set of modules is always loaded at login for all users.
- Current state can be viewed by running:

```
$> module list
```



# Default modules example

```
adrianj@eslogin001:~> module list
Currently Loaded Modulefiles:
 1) modules/3.2.6.7
 2) nodestat/2.2-1.0500.41375.1.85.ari
 3) sdb/1.0-1.0500.43793.6.11.ari
 4) alps/5.0.3-2.0500.8095.1.1.ari
 5) MySQL/5.0.64-1.0000.7096.23.1
 6) lustre-cray_ari_s/2.3_3.0.58_0.6.6.1_1.0500.7272.12.1-1.0500.44935.7.1
 7) udreg/2.3.2-1.0500.6756.2.10.ari
 8) ugni/5.0-1.0500.0.3.306.ari
 9) gni-headers/3.0-1.0500.7161.11.4.ari
10) dmapp/6.0.1-1.0500.7263.9.31.ari
11) xpmem/0.1-2.0500.41356.1.11.ari
12) hss-llm/7.0.0
13) Base-opts/1.0.2-1.0500.41324.1.5.ari
14) craype-network-aries
15) craype/1.06.05
16) cce/8.2.0.181
...
```



# Viewing available modules

- There may be many hundreds of possible modules available to users.
  - Beyond the pre-loaded defaults there are many additional packages provided by Cray
  - Sites may choose to install their own versions.
- Users can see all the modules that can be loaded using the command:
  - `module avail`
- Searches can be narrowed by passing the first few characters of the desired module, e.g.

```
adrianj@eslogin001 :~> module avail gc
```

```
----- /opt/modulefiles -----  
gcc/4.6.1          gcc/4.7.2          gcc/4.8.0  
gcc/4.6.3          gcc/4.7.3          gcc/4.8.1(default)
```



# Modifying the default environment

- Loading, swapping or unloading modules:
  - The default version of any individual modules can be loaded by name
    - e.g.: `module load perftools`
  - A specific version can be specified after the forward slash.
    - e.g.: `module load perftools/6.1.0`
  - Modules can be swapped out in place
    - e.g.: `module swap intel intel/13.1.1.163`
  - Or removed entirely
    - e.g.: `module unload perftools`
- Modules will automatically change values of variables like PATH, MANPATH, LM\_LICENSE\_FILE... etc
  - Modules also provide a simple mechanism for updating certain environment variables, such as PATH, MANPATH, and LD\_LIBRARY\_PATH
  - In general, you should make use of the modules system rather than embedding specific directory paths into your startup files, makefiles, and scripts



```
adrianj@eslogin008:~> module show fftw
```

```
-----  
/opt/cray/modulefiles/fftw/3.3.0.4:
```

```
setenv          FFTW_VERSION 3.3.0.4  
setenv          CRAY_FFTW_VERSION 3.3.0.4  
setenv          FFTW_DIR /opt/fftw/3.3.0.4/sandybridge/lib  
setenv          FFTW_INC /opt/fftw/3.3.0.4/sandybridge/include  
prepend-path    PATH /opt/fftw/3.3.0.4/sandybridge/bin  
prepend-path    MANPATH /opt/fftw/3.3.0.4/share/man  
prepend-path    CRAY_LD_LIBRARY_PATH /opt/fftw/3.3.0.4/sandybridge/lib  
setenv          PE_FFTW_REQUIRED_PRODUCTS PE_MPICH  
prepend-path    PE_PKGCONFIG_PRODUCTS PE_FFTW  
setenv          PE_FFTW_TARGET_interlagos interlagos  
setenv          PE_FFTW_TARGET_sandybridge sandybridge  
setenv          PE_FFTW_TARGET_x86_64 x86_64  
setenv          PE_FFTW_VOLATILE_PKGCONFIG_PATH  
/opt/fftw/3.3.0.4/@PE_FFTW_TARGET@/lib/pkgconfig  
prepend-path    PE_PKGCONFIG_LIBS  
fftw3f_mpi:fftw3f_threads:fftw3f:fftw3_mpi:fftw3_threads:fftw3  
module-whatism FFTW 3.3.0.4 - Fastest Fourier Transform in the West  
-----
```





# Summary of Useful module commands

- Which modules are available?
  - `module avail, module avail cce`
- Which modules are currently loaded?
  - `module list`
- Load software
  - `module load perftools`
- Change programming environment
  - `module swap PrgEnv-cray PrgEnv-gnu`
- Change software version
  - `module swap cce/8.0.2 cce/7.4.4`
- Unload module
  - `module unload cce`
- Display module release notes
  - `module help cce`
- Show summary of module environment changes
  - `module show cce`



# Compiling applications for the Cray XC

---



# Compiler Driver Wrappers (1)

- All applications that will run in parallel on the Cray XC should be compiled with the standard language wrappers.

The compiler drivers for each language are:

- `cc` - wrapper around the C compiler
  - `CC` - wrapper around the C++ compiler
  - `ftn` - wrapper around the Fortran compiler
- These scripts will choose the required compiler version, target architecture options, scientific libraries and their include files automatically from the module environment.
- Use them exactly like you would the original compiler, e.g. To compile `prog1.f90` run  

```
ftn -c prog1.f90
```



# Compiler Driver Wrappers (2)

- The scripts choose which compiler to use from the PrgEnv module loaded

PrgEnv	Description	Real Compilers
PrgEnv-cray	Cray Compilation Environment	crayftn, craycc, crayCC
PrgEnv-intel	Intel Composer Suite	ifort, icc, icpc
PrgEnv-gnu	GNU Compiler Collection	gfortran, gcc, g++

- Use module swap to change PrgEnv, e.g.
  - `module swap PrgEnv-cray PrgEnv-intel`
- PrgEnv-cray is loaded by default at login. This may differ on other Cray systems.
  - use `module list` to check what is currently loaded
- The Cray MPI module is loaded by default (`cray-mpich`).
  - To support SHMEM load the `cray-shmem` module.
- Check that the `craype-ivybridge` module is loaded
- The drivers automatically support an MPI build
  - No need to use specific wrappers such as `mpiifort`, `mpicc` or explicitly link to libraries



## PLEASE NOTE : Cross Compiling Environment

- You are compiling on a Linux login node but generating an executable for a CLE compute node
- Do not use `crayftn`, `craycc`, `ifort`, `icc`, `gcc`, `g++`... unless you want a Linux executable for the login node
  - **ALWAYS** Use `ftn`, `cc`, or `CC` instead
  - Only use the direct compiler commands if the executable is supposed to run on the login nodes (utilities, setup, ...)



# Compiler Versions

- There are usually multiple versions of each compiler available to users.
  - The most recent version is usually the default and will be loaded when swapping PrgEnvs.
  - To change the version of the compiler in use, swap the Compiler Module. e.g. `module swap cce cce/8.1.6`

PrgEnv	Compiler Module
PrgEnv-cray	cce
PrgEnv-intel	Intel
PrgEnv-gnu	gcc
<del>PrgEnv-pgi</del>	<del>pgi</del>



# About the `-I`, `-L` and `-l` flags

- For libraries and include files covered by module files, you should NOT add anything to your Makefile
  - No additional MPI flags are needed (included by wrappers)
  - You do not need to add any `-I`, `-l` or `-L` flags for the Cray provided libraries
- If your Makefile needs an input for `-L` to work correctly, try using `'.'`
- If you really, really need a specific path, try checking `'module show X'` for some environment variables



```
adrianj@eslogin008:~> module show fftw
```

```
-----  
/opt/cray/modulefiles/fftw/3.3.0.4:
```

```
setenv          FFTW_VERSION 3.3.0.4  
setenv          CRAY_FFTW_VERSION 3.3.0.4  
setenv          FFTW_DIR /opt/fftw/3.3.0.4/sandybridge/lib  
setenv          FFTW_INC /opt/fftw/3.3.0.4/sandybridge/include  
prepend-path    PATH /opt/fftw/3.3.0.4/sandybridge/bin  
prepend-path    MANPATH /opt/fftw/3.3.0.4/share/man  
prepend-path    CRAY_LD_LIBRARY_PATH /opt/fftw/3.3.0.4/sandybridge/lib  
setenv          PE_FFTW_REQUIRED_PRODUCTS PE_MPICH  
prepend-path    PE_PKGCONFIG_PRODUCTS PE_FFTW  
setenv          PE_FFTW_TARGET_interlagos interlagos  
setenv          PE_FFTW_TARGET_sandybridge sandybridge  
setenv          PE_FFTW_TARGET_x86_64 x86_64  
setenv          PE_FFTW_VOLATILE_PKGCONFIG_PATH  
/opt/fftw/3.3.0.4/@PE_FFTW_TARGET@/lib/pkgconfig  
prepend-path    PE_PKGCONFIG_LIBS  
fftw3f_mpi:fftw3f_threads:fftw3f:fftw3_mpi:fftw3_threads:fftw3  
module-whatism FFTW 3.3.0.4 - Fastest Fourier Transform in the West  
-----
```





# OpenMP

- OpenMP is support by all of the PrgEnvs.
  - CCE (PrgEnv-cray) recognizes and interprets OpenMP directives by default. If you have OpenMP directives in your application but do not wish to use them, disable OpenMP recognition with `-hnoomp`.

PrgEnv	Enable OpenMP	Disable OpenMP
PrgEnv-cray	<code>-homp</code> (default)	<code>-hnoomp</code>
PrgEnv-intel	<code>-openmp</code>	(default)
PrgEnv-gnu	<code>-fopenmp</code>	(default)



# Compiler man pages and documentation

- For more information on individual compilers

PrgEnv	C	C++	Fortran
PrgEnv-cray	man craycc	man crayCC	man crayftn
PrgEnv-intel	man icc	man icpc	man ifort
PrgEnv-gnu	man gcc	man g++	man gfortran
Wrappers	man cc	man CC	man ftn

- To verify that you are using the correct version of a compiler, use:
  - **-V** option on a cc, CC, or ftn command with CCE and Intel
  - **--version** option on a cc, CC, or ftn command with GNU
- Cray Reference Manuals:
  - C and C++: <http://docs.cray.com/books/S-2179-81/>
  - Fortran: <http://docs.cray.com/books/S-3901-81/>



# Dynamic compilation

- Default behaviour is to perform static linking
- Dynamic linking possible:
  - Use the -dynamic flag when invoking the compiler for linking.
  - Set the environment variable `CRAYPE_LINK_TYPE=dynamic` without any extra compilation/linking options.
- Will need to have libraries available on /work filesystem



# OpenMP

- OpenMP is **ON by default**
  - This is the opposite default behavior that you get from GNU and Intel compilers
  - Optimizations controlled by **-OthreadN (ftn)** or **-hthreadN (cc/CC)**, N=0-3 [default N=2]
  - To shut off use **-O/-h thread0** or **-xomp (ftn)** or **-hnoomp**
- Autothreading is NOT on by default
  - **-hautothread** to turn on
  - Interacts with OpenMP directives
- If you do not want to use OpenMP and have OMP directives in the code, make sure to shut off OpenMP at compile time



# CCE – GNU – Intel compilers

- More or less all optimizations and features provided by CCE are available in Intel and GNU compilers
  - GNU compiler serves a wide range of users & needs
    - Default compiler with Linux, some people only test with GNU
    - **GNU defaults are conservative** (e.g. -O1)
      - -O3 includes vectorization and most inlining
    - Performance users set additional options
  - Intel compiler is typically more aggressive in the optimizations
    - **Intel defaults are more aggressive** (e.g. -O2), to give better performance “out-of-the-box”
      - Includes vectorization; some loop transformations such as unrolling; inlining within source file
    - Options to scale back optimizations for better floating-point reproducibility, easier debugging, etc.
    - Additional options for optimizations less sure to benefit all applications
  - **CCE is even more aggressive** in the optimizations by default
    - Better inlining and vectorization
    - Aggressive floating-point optimizations
    - OpenMP enabled by default
- GNU users probably have to specify higher optimisation levels



# Cray, Intel and GNU compiler flags

Feature	Cray	Intel	GNU
Listing	-ra (fnt) -hlist=a (cc/CC)	-opt-report3	-fdump-tree-all
Free format (fnt)	-f free	-free	-ffree-form
Vectorization	By default at -O1 and above	By default at -O2 and above	By default at -O3 or using -ftree-vectorize
Inter-Procedural Optimization	-hwp	-ipo	-flto (note: link-time optimization)
Floating-point optimizations	-hfpN, N=0...4	-fp-model [fast fast=2 precise  except strict]	-f[no-]fast-math or -funsafe-math-optimizations
Suggested Optimization	(default)	-O2 -xAVX	-O2 -mavx -ftree-vectorize -ffast-math -funroll-loops
Aggressive Optimization	-O3 -hfp3	-fast	-Ofast -mavx -funroll-loops
OpenMP recognition	(default)	-fopenmp	-fopenmp
Variables size (fnt)	-s real64 -s integer64	-real-size 64 -integer-size 64	-freal-4-real-8 -finteger-4-integer-8



# ARCHER PBS Batch System

---

EPSRC

NERC SCIENCE OF THE ENVIRONMENT



CRAY  
THE SUPERCOMPUTER COMPANY

epcc



# Requesting resources from PBS

Jobs provide a list of requirements as #PBS comments in the headers of the submission script, e.g.

```
#PBS -l walltime=12:00:00
```

These can be overridden or supplemented as submission by adding to the qsub command line, e.g.

```
> qsub -l walltime=11:59:59 run.pbs
```

Common options include:

Option	Description
-N <name>	A name for job,
-q <queue>	Submit job to a specific queues.
-o <output file>	A file to write the job's stdout stream in to.
--error <error file>	A file to write the job's stderr stream in to.
-j oe	Join stderr stream in to stdout stream as a single file
-l walltime=<HH:MM:SS>	Maximum wall time job will occupy
-A <code>	Account to run job under (for controlling budgets)





# Requesting parallel resources

Jobs must also request “chunks” of nodes:

This is done using the select option, e.g.

```
-l select=<numnodes>
```

Option	Description
select=<numnodes>	Requests <numnodes> nodes from the system.
select=bigmem=true	High memory nodes

```
qsub -l select=<numnodes> ./myjob.pbs
```

```
qsub -l select=<numnodes>:bigmem=true ./mybigjob.pbs
```



# Launching Parallel applications

- Cray terminology
  - refer to compute resources in terms of *Processing Elements*
  - one MPI process corresponds to one PE
- aprun is the parallel job launcher
  - aprun launches parallel jobs on the compute nodes.
  - aprun man page contains several useful examples
  - The most important parameters to set is -n:

Description	Option
Total Number of PEs used by the application	-n
Number of PEs per compute node	-N

```
aprun -n 24 ./mymprog.exe # default -N 24  
aprun -n 24 -N 12 ./mymprog.exe # uses 2 nodes
```



# File Systems and Batch Jobs

- After login, you are placed in the /home filesystem
  - e.g. /home/y14/y14/guest01/
- Login nodes can see /home and /work filesystems
- Compute nodes can only see /work
- You must launch all parallel jobs from /work
  - `cd /work/y14/y14/guest01/nobelprizejobs/`
  - `qsub nobelprize.pbs`
- Very common mistake
  - jobs submitted from /home/ will almost certainly fail *at runtime*



# Example batch script

```
#!/bin/bash --login

# PBS job options (name, compute nodes, job time)
#PBS -N Example_MPI_Job
#PBS -l select=64
#PBS -l walltime=00:20:00

# Replace [project code] below with your project code (e.g. t01)
#PBS -A [project code]

# Make sure any symbolic links are resolved to absolute path
export PBS_O_WORKDIR=$(readlink -f $PBS_O_WORKDIR)

# Change to the directory that the job was submitted from
# (remember this should be on the /work filesystem)
cd $PBS_O_WORKDIR

# Launch the parallel job
# Using 1536 MPI processes and 24 MPI processes per node
aprun -n 1536 ./my_mpi_executable.x arg1 arg2
```



# PBS configuration

- Users usually submit to a single (default) queue
  - unless there is a special reserved queue, e.g. during a course
  - PBS decides when to run your job based on requested resources
  - assumes you will run **for all the time requested on all the nodes**
  - maximum runtime: 24 hours
  - maximum job size: entire machine
- Limits
  - maximum number of jobs in the system: 16 (max 8 running)
  - after this jobs are rejected
  - jobs rejected at *submission* if
    - budget code does not have enough time
    - you specify a budget you do not have access to
  - jobs may queue indefinitely
    - if a valid budget has insufficient resources when PBS attempts to *run* the job



# Special Queues

- Low priority – run only when resources are lightly used
  - `qsub -q low submit.pbs`
  - maximum 3 hours and 512 nodes; user not charged
- Long jobs
  - `qsub -q long submit.pbs`
  - maximum of 48 hours and 256 nodes
- Short / debug queue – enabled 10:00 – 17:00 Mon - Fri
  - `qsub -q short submit.pbs`
  - maximum of 20 minutes and 8 nodes



# Postprocessing / Serial nodes

- 2 nodes each with 40 Intel cores and 1 TB of memory
  - different architecture from compute nodes
  - not supported by Cray programming environment
  - general purpose: not as tightly controlled as the compute nodes
- Compiling
  - compile post-processing jobs directly calling gcc, gfortran, icc or ifort
- Running
  - `qsub -l select=serial=true:ncpus=1 submit.pbs`



# Project Management

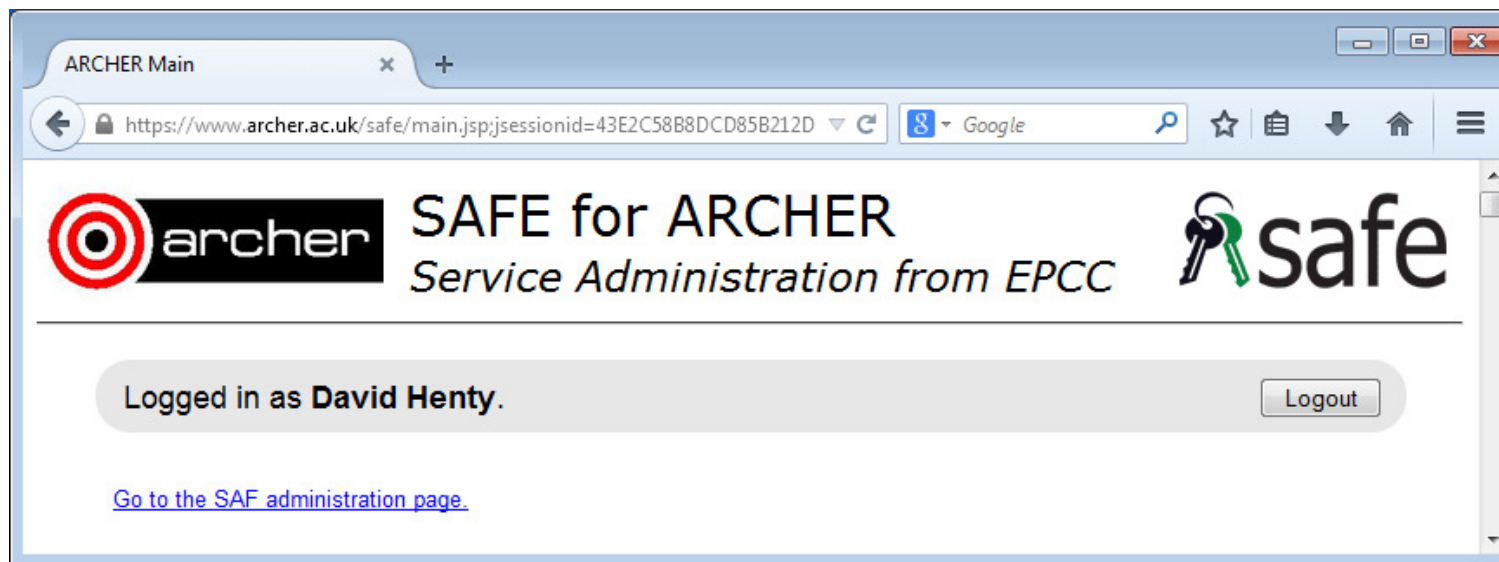
---





# ARCHER SAFE

- All users have a web account on the SAFE: [www.archer.ac.uk/safe/](https://www.archer.ac.uk/safe/)



- Single point of contact for
  - managing machine account(s) for users (e.g. password request)
  - managing projects for Principal Investigators



# Job charging

- Jobs must be charged to a “budget” or CPU account
  - standard budget is the project name, e.g. #PBS –A y14
  - individual projects may set up sub-budgets, e.g. y14-dev
  - this is all controlled by the Principal Investigator via the SAFE
- Allocation done in units of “kAUs”
  - 1 kilo Allocation Unit = 1000 Gflop-hours (on Linpack benchmark)
- On ARCHER, 1 kAU = 56 pence (for EPSRC/NERC users)
  - 1 core-hour = 0.015 kAU; 1 node-hour = 0.36 kAU
  - charged for a full node regardless of how many cores you use
  - charged for how long your job actually runs
    - regardless of what you requested or whether job completed successfully



# Getting access to ARCHER

- Standard research grant
  - Request Technical Assessment using form on ARCHER website
  - Submit completed TA with notional cost in Je-S
  - Apply for time for maximum of 2 years
- ARCHER Resource Allocation Panel (RAP)
  - Request Technical Assessment using form on ARCHER website
  - Submit completed TA with RAP form
  - Every 4 months
- Application for computer time only
  - Instant Access – Pump-Priming Time
  - Request Technical Assessment using form on ARCHER website
  - Submit completed TA with 2 page description of work



# Funding calls

- Embedded CSE support
  - Through a series of regular calls, Embedded CSE (eCSE) support provides funding to the ARCHER user community to develop software in a sustainable manner for running on ARCHER. Funding will enable the employment of a researcher or code developer to work specifically on the relevant software to enable new features or improve the performance of the code
  - Apply for funding for development effort
  - Regular calls are every 4 months
    - actively encouraging applications from New Scientific Communities
- See <http://www.archer.ac.uk> for details



# Support

- Helpdesk
  - Email [support@archer.ac.uk](mailto:support@archer.ac.uk)
  - via ARCHER SAFE <http://www.archer.ac.uk/safe>
  - phone: +44 (0)131 650 5000
  - By post, to:
    - ARCHER Helpdesk
    - EPCC
    - James Clerk Maxwell Building
    - Peter Guthrie Tait Road
    - EDINBURGH EH9 3FD
  - <http://www.archer.ac.uk/community/techforum/>



# Training opportunities

- ARCHER Training (free to academics):
  - <http://www.archer.ac.uk/training/>
- EPCC MSc in HPC
  - <http://www.epcc.ed.ac.uk/msc/>



# Virtual Tutorials

- Live online interactive sessions
  - a forum for users of ARCHER to ask any questions you may have about the ARCHER service.
- Q&A sessions, starting with short lecture on specific topic
  - An opportunity for attendees of ARCHER training courses to discuss any issues related to a course or questions about course material that may have arisen since attending the course.
- Broadcast using Blackboard Collaborate.
- Every second Wednesday of the month
  - <http://www.archer.ac.uk/training/virtual/>.



# Feedback and follow-up

- <http://www.archer.ac.uk/training/feedback/>





# What now?

- You can attempt the ARCHER driving test
  - [www.archer.ac.uk/training/course-material/online/driving\\_test.php](http://www.archer.ac.uk/training/course-material/online/driving_test.php)
- On successful completion, eligible users can apply for
  - account on ARCHER
  - 1,200 kAUs of time (80,000 core-hours) over 12 months
- Further information
  - This online material: [www.archer.ac.uk/training/course-material/online/](http://www.archer.ac.uk/training/course-material/online/).
  - Documentation: <http://www.archer.ac.uk/documentation/>.
  - Helpdesk: [support@archer.ac.uk](mailto:support@archer.ac.uk)
  - Training: <http://www.archer.ac.uk/training/>.

