# eCSE Final Report for ARCHER2-eCSE04-5

The primary purpose of the final report is to showcase the success of your eCSE project and demonstrate the value of the eCSE programme. Each section has this in mind. Not all sections will be relevant to your particular eCSE project.

|  |  |
| --- | --- |
| eCSE ID: | ARCHER2-eCSE04-5 |
| eCSE Title: | Scalable and robust Firedrake deployment on ARCHER2 and beyond |
| PI Name: | Dr David A Ham |
| Author of this document: | Dr David A Ham, Prof Patrick E Farrell, Dr Jack Betteridge |
| Name of technical staff on the project: | Dr Jack Betteridge |
| List of names to acknowledge with the project (by default we will use the PI, Co-Is and Technical staff members): | Dr David A Ham, Prof Patrick E Farrell, Dr Jack Betteridge |
| Project Start Date: | 01/11/2021 |
| Project End Date: | 30/04/2022 |
| Number of Project Months Funded: | 6 |

## Technical Report Components:

* Achievement of Objectives – A very short summary which will be provided to the eCSE panel.
* Availability of software – information for the web site and location of executables
* Performance Improvements and Return on Investment
* Sustainability and usability highlights – If applicable, this allows you to demonstrate how the eCSE has improved the usability and sustainability of your software.
* Publishable Summary – a 1 page document which will be used to showcase your eCSE to the general public. Highlights scientific benefits and impact.
* Technical Report – a technical report on the work completed in the eCSE.

## Achievement of Objectives

List the objectives from your proposal and describe progress against these achievements in the table (short summary, a few sentences). These will be supplied to the eCSE panel to read.

|  |  |  |
| --- | --- | --- |
| ***Objective 1****: Build a Singularity container on ARCHER2 containing a fully functioning Firedrake installation. We will build a “vanilla”, complex mode and a teaching Firedrake container, the latter containing the Jupyter notebooks package for delivering training material. This singularity container can be merged into the existing Firedrake CI suite. Currently, if all tests pass on the build hardware, then a Docker container is built from the master branch. We will replace this Docker container, which is not suitable for HPC nor convertible to a Singularity container. At this stage of the project end users will already*  *benefit from the containers, since they will be publicly available.* | ***Success Metrics****: All Firedrake tests pass locally on ARCHER2 in both real and complex mode inside a containerised install. Additionally, Singularity containers built on CI build hardware replacing existing Docker containers.* | *We successfully built a Singularity container for use on ARCHER2. It was possible on a local machine (with administrator privileges)and on ARCHER2 (without) to build a Singularity image from the current Docker image. This allows Firedrake to have both a Docker and Singularity container and maintain a single source of truth.* ***All Firedrake tests when running inside the Singularity container****\*. We are currently looking into hosting options for the generated containers since Singularity hub is no longer available.  \*With the exception of parallel tests due to a bug in the testing framework. Parallel functionality was tested and working, fixing the test framework is ongoing work.* |
| ***Objective 2****: Container image optimisation for HPC. Firedrake has complicated parallel library dependencies (e.g: MPI, HDF5), which are unlikely to work out of the box in a massively parallel configuration, a container allows Firedrake developers to correctly set up this*  *environment. Additionally, synchronous Python initialisation over many nodes is a performance issue due to contention for network file system access. We will build a*  *launch utility, exploring solutions such as Spindle, to mitigate this bottleneck and set defaults for MPI process placement to maximise performance and efficiency.* | ***Success Metrics****: It will be possible to launch Firedrake scripts using the new launch utility and the correct network interface will be used for MPI, within Singularity.* | *The Singularity image has been configured in such away that it aligns with the material in the ARCHER2 course “Reproducible Computational Environments Using Containers” in the section on “Running MPI parallel jobs using Singularity containers”.*  *Due to incompatibilities with ARCHER2 as well as other modern HPC facilities it was not possible to integrate Spindle into the container. Since* ***launch works correctly without a dedicated utility****, this idea was not pursued further. This may be revisited in the future if the Spindle package is updated to work with newer versions of SLURM.* |
| ***Objective 3****: Development of a Spack package. For some users a containerised install will be*  *unsuitable, as often users will want a custom installation of one of Firedrake’s*  *dependencies or to integrate with another software package. To make HPC installation simpler for these users we will leverage the features of the Spack package manager, which is designed to make building software on HPC simpler, by ensuring the desired compilers and libraries are used and dependencies are built with the same tools as well as any additional configuration flags required by the target package.* | ***Success Metrics****: Firedrake will be installable on ARCHER2 using the Spack package*  *manager without manual intervention.* | *Over the course of the project an effective workflow has been developed for using Spack on ARCHER2. We believe this to be of great value to* all *ARCHER2 users, as it allows a huge range of packages to be installed in an efficient manner. In addition a package has been developed specifically for Firedrake and for all the Firedrake dependencies missing from Spack.* ***Firedrake is installable on ARCHER2 using the Spack package***  ***manager without manual intervention, instructions available*** [***here***](https://hackmd.io/Sg3fYXuCTl61d_LAg4QnMw) ***(working document).*** *However, this does require carefully configuring Spack for the HPC environment.* |
| ***Objective 4****: Benchmark container implementation. To ensure performance of the optimisations implemented in (2.) we can benchmark the container against naïve script-based installation. Our launch utility must be at least as fast as current solutions for improving Firedrake’s performance on HPC, else users will not benefit.* | ***Success Metrics****: Python initialisation time using new launch utility will be reduced and we obtain low latency/high throughput for MPI communication as a “bare metal”*  *installation. Firedrake script performance within a container will be better than for a*  *naïve installation and comparable to any existing performance enhancing solutions.* | *Due to being unable to use Spindle as a solution for launch, no new launch utility was developed. The best method to improve Python initialisation time is still to tarball an installation (including the new Spack installation) in `/tmp` and unpack this once per compute node.*  *Time constraints did not allow for the thorough benchmarking we initially hoped to carry out, but initial results obtained using the Singularity container look promising. We hope to further improve these numbers in future Firedrake development.* |
| *Other achievements you wish to highlight* | | *This eCSE also fixed a bug in PETSc/petsc4py which caused Firedrake to randomly hang. This can result extreme costs for HPC users as deadlocks do not terminate jobscripts, burning through CUs.*  *The bug fix allows for distributed objects created in petsc4py to be cleaned up safely in parallel.* ***This now removes random hanging of Firedrake scripts*** *experienced by some Firedrake users when running in parallel, making the overall installation more robust.* |

## Availability of Software

List the application codes you worked on during the eCSE project. Note the executable names on ARCHER2 – we will search CPU usage against these executable names to assess the number of codes running on ARCHER2 that have benefitted from the eCSE programme.

|  |  |
| --- | --- |
| Application Name | Name(s) of executables on the ARCHER2 system |
| Spack | spack |
| Apptainer/Singularity | singularity |
| Firedrake | (This is a Python package, so monitoring cannot be performed by examining the executable) |
| PETSc and petsc4py | (System library and Python package) |

Please provide a short paragraph of text explaining how you can get access to these application codes on ARCHER2. We will place this text on the ARCHER2 web site. Include links to relevant application web pages.

|  |
| --- |
| All three projects are open source:  Firedrake:  Website <https://www.firedrakeproject.org/>  Repository <https://github.com/firedrakeproject/firedrake>  Spack:  Website <https://spack.io/>  Repository <https://github.com/spack/spack>  PETSc and petsc4py  Website <https://petsc.org/release/>  Repository <https://gitlab.com/petsc/petsc>  Singularity is already installed on ARCHER2 and is available as a module. The Firedrake Singularity image will be made available once a suitable host is found, but can presently be converted from a Docker image.  Furthermore, there are extensive instructions on how to install Firedrake using Spack as a working document available here  <https://hackmd.io/@TzVnFeL0TMCb3FaAi9qYBA/BJ4okaRN9>  and use the singularity image here  <https://hackmd.io/_6XZhc93RtyGzjpqDuyT-w>  These will be incorporated into the Firedrake wiki (<https://github.com/firedrakeproject/firedrake/wiki>) once various upstream changes have been made. |

## Performance Improvement and Return on Investment

Many (but not all) eCSE projects result in a performance improvement to an application code, which either allows previously untenable science to be carried out, or science to be carried out for less CPU resource (resulting in a significant financial saving and freeing up resources for other science). Understanding and quantifying this helps to demonstrate the value of the eCSE programme.

If you have achieved a performance improvement please complete the table below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Application Code Name | Average execution time before the eCSE\* | Average execution time after the eCSE | Estimate of percentage of future CPU time that will benefit from the improvement\*\* | Comments / Notes |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

\*This is an estimate and an average across processor counts.

\*\*This is an estimate of what proportion of runs of the code will benefit from the performance improvement. For larger codes it may only be certain types of use cases that benefit.

## Sustainability and usability highlights

While some eCSE projects provide performance improvements, others provide other forms of benefits, for example improving a code’s usability, enhancing sustainability, or potentially increasing the user base through the addition of new functionality. If this is the case for your eCSE, please provide a paragraph of text explaining these improvements, detailing who will benefit and how they will benefit. We may use this to create a short eCSE highlight / blog.

|  |
| --- |
| The main aim of this eCSE was to improve the overall HPC experience for those wishing to use Firedrake: An automated system for the solution of partial differential equations using the finite element method and sophisticated code generation.  Firedrake is a relatively small Python package, but has a large number of dependencies. On an end user’s PC these dependencies are handled by a custom installation script. By creating a Spack package we have simplified the build process on HPC to be just as straightforward. A HPC user can now install Firedrake in a single line using the Spack package manager, while still taking advantage of system modules such as the vendor MPI distribution. Previously it would take expert HPC knowledge and significant time investment to build Firedrake on HPC. By creating a Spack package, we not only standardise the build procedure, but make it possible for non-experts (who are familiar or willing to familiarise themselves with Spack) to create their own Firedrake installation.  For users who do not want to develop Firedrake on HPC, we have created a Singularity container. This offers a zero installation route for ARCHER2 users who wish to just run their Firedrake scripts with near bare-metal performance.  This eCSE also fixed a bug in PETSc/petsc4py which affected the usability of Firedrake. The bug fix allows for distributed objects created in petsc4py to be cleaned up safely in parallel. This now removes random hanging of Firedrake scripts experienced by some Firedrake users when running in parallel. |

## Publishable Summary

Max 1 page, please provide a short summary of the eCSE work carried out. This should be written to be accessible to a member of the general public and is your opportunity to showcase the success of your eCSE project. This section will be made public in full or in part. You may reuse text from sections above. Please make sure you include:

* A summary of the science that will benefit from the eCSE. Describe any potential impact. Why is this work of value to society?
* Provide an overview of the work carried out. What has been done to the application code? What has been achieved?
* A brief summary of the application codes involved.
* Include any relevant publications
* If at all possible, please provide an image.

## Technical Report

This section should contain a technical report which describes the work carried out on the eCSE project, for the benefit of the ARCHER2 community. This section will be made public in full.

This section should be no longer than 3-4000 words (roughly 4-6 pages including figures and references). Please structure the article as you see fit, with no more than three levels of headings, and please do include an abstract. Display items (figures, tables, etc.) are encouraged. Please use the Vancouver (numbered) system of references.

You may decide to submit this to a journal. If so, please supply a copy of the paper for an internal record. We are happy to wait and publish the report / link to the report when the journal allows.

We can assist you to gain a DOI for the report.

You should submit the report as a PDF (either with the earlier sections combined with the final technical section or as two separate documents), but please also separately include the source file(s) for the whole report: DOC, ODT, or any image files. Please ensure that any change tracking within the document has been completed when you submit (i.e. no changes are showing that still need to be accepted or rejected).