Proposal for a eCSE Application

Applicants for a **eCSE application** should use this template.

The ARCHER2 eCSE04 call will close at **16:00 on Tuesday 8 June 2021**. No applications will be accepted after this time.

Please note:

- There is a hard page limit for each section which is shown at the start of each section.
- The font size should be no smaller than 11pt and the margins should be no smaller than 2.5cm.

Please upload your proposal when you fill in your eCSE application online form via the eCSE Funding Calls section within the SAFE: https://safe.epcc.ed.ac.uk/.

Please read the Applications Guidance (linked in to the page https://www.archer2.ac.uk/ecse/calls/) for further instructions on how to complete your eCSE online application.

If you have any queries or require assistance regarding your application, please contact the ARCHER2 service desk: support@archer2.ac.uk.

Project Information

1. Project Title (as given in on-line SAFE form)

Scalable and robust Firedrake deployment on ARCHER2 and beyond

2. PI Name and Institution

David A. Ham
Imperial College London

3. Project Objectives and success metrics (max. 1 page)

Our goal for this project is to make the installation procedure for the Firedrake framework (and dependencies) simple and robust on any HPC platform. A revised installation procedure will allow anyone to install and run Firedrake scripts, even with minimal HPC experience. Two new approaches for installation will be developed: Singularity for containerisation and Spack for packaging. This will offer end users multiple installation procedures to cover individual use cases and different machines.

These improvements will make the installation of Firedrake much quicker and easier allowing users to utilise the facility with resources most appropriate for them. We will target tier 1,2 and 3 machines allowing researchers to target the platform best suited to their project, rather than the one Firedrake currently has bespoke build support for. Currently the script-based installation works well for end users on laptops and workstations but requires significant manual intervention on HPC, as every system is different. An added benefit of a Singularity container is the installation time will be reduced from approximately an hour building all dependencies, to a few minutes downloading and initialising the container. Our objectives are broken down into the following steps:

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 metric: 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.
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 metric: 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.
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 metric: Firedrake will be installable on ARCHER2 using the Spack package manager without manual intervention.
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 metric: 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.

4. Project Overview, Technical Information and Workplan (max. 2 pages)

Overview

Firedrake (firedrakeproject.org/) is a high-level Python based system for the specification and solution of PDEs using the finite element method. It uses sophisticated code generation to provide mathematicians, scientists, and engineers with a very high productivity way to create sophisticated high performance simulations.

By combining the features of a large number of external packages, Firedrake has become a flexible and extensible framework for continuum mechanics simulation. The cost of having all these dependencies is the initial difficulty installing, which is why an installation script is provided to resolve dependencies for end users. However, this script is not a robust solution for installing Firedrake on HPC, where the user does not have the same (administrator level) control over the system.

Currently there several installation procedures independently maintained by the Firedrake project for a small range of different HPC facilities (github.com/firedrakeproject/firedrake-archer, github.com/firedrakeproject/isambard/blob/alternative_install/README.md). Several more are maintained separately by individuals for other HPC facilities ranging from institutional machines to national supercomputers. Each script or repository is bespoke and contains workarounds to enable the current installation script to run providing the end user with a working Firedrake environment. These scripts are subject to bit-rot as the dependencies for Firedrake and the installation script change regularly. Both objectives 1. and 3. below will simplify installation procedure for HPC users.

Objectives:

1. Provide Singularity containers suitable for ARCHER2.
2. Improve performance when launching Firedrake scripts on HPC.
3. Create a Spack package as an alternative install path.
4. Benchmark the installation solutions.

Technical Information and Workplan (over 6 months = 26 weeks)

Target 1: (5 weeks) Firedrake currently uses containerisation in the form of Python VirtualEnvs as well as Docker containers. Both are primarily designed for local machine and single node cloud use, neither are suitable portable containers for HPC use. Modern HPC facilities allow the use of Singularity containers, which do not require administrator privileges, correctly interface with the job scheduler, and allow MPI implementations to utilise the fast interconnect through the Open Fabrics Interface (OFI) hardware abstraction layer. Using a Singularity container allows the developer (us) to build a suitable MPI distribution and other parallel packages within the container without paying a performance penalty. The first goal of this project would install Firedrake inside a Singularity container on ARCHER2. This solution will be suitable for most end users as they only require a correctly configured installation with no modifications for running simulations in. The same container will also be integrated into the existing Firedrake CI solution.

Target 2: (7 weeks) An issue that arises from using a Python based tool on HPC is significant network filesystem contention at Python initialisation. On invocation and on every MPI rank, the Python interpreter touches a very large number of files. Currently users can use the rather ad-hoc method of compressing the entire install and unpacking to /tmp on each node.

We will create a launch utility to mitigate these issues, which would incorporate tools like Spindle[1] (https://computing.llnl.gov/projects/spindle) and ensure sensible defaults for MPI process placement. This gives all users of Firedrake on HPC a performance boost without impacting their workflow. Part of the project will be testing and evaluating Spindle. Currently the tool supports dynamic libraries on disk at invocation but requires additional work to handle libraries created by Firedrake's code generation at run time.

Target 3: (6 weeks) Using containers to deliver a prebuilt Firedrake installation will be an appropriate solution for many ARCHER2 users, however in some cases, detailed below, this method of installation will not be appropriate. The same is true for users of other HPC facilities, where Singularity may not be available, or the architecture is not supported. For these users we will design a package for the Spack package manager.
Spack (https://spack.io/) is a popular choice for HPC users to install packages with complex dependencies. Currently Firedrake does not support this method of installation, but Spack is one of the few package managers that would be able to deliver the fine-grained control over building dependencies (currently handled by the install script) and is designed with HPC in mind. Furthermore, Spack can take advantage of any dependencies already available on a given system, either available through the OS or through the module system.
For example the PETSc library is available on ARCHER2 as a module but doesn't have all the external dependencies a Firedrake user requires to run Firedrake. For instance, it has MUMPS, but not Chaco. Spack can use the modules that *are* available without rebuilding them, build the dependencies that aren't and then build and link PETSc against all necessary external dependencies. An alternative use case would be an end user who wishes to build their own PETSc, as many do, which could be added to Spack's internal database allowing Firedrake to be built without rebuilding PETSc or its dependencies. This gives users the same control over their installation that they would have if they were installing on a personal machine, which is functionality that cannot be provided by a containerised solution.

Target 4: (4 weeks) To ensure there is no performance regression going from "bare metal" to either Singularity or Spack installations we will benchmark both against an existing installation. We expect the MPI communications (latency and throughput) to be comparable to the existing install since we will make use of the OFI libraries available on ARCHER2. We expect the initialisation time to decrease significantly for large scale runs using the launch script developed in Target 2.

Target 5: (4 weeks) Finally we will communicate our findings back to the community. Upon creation of the container, we will contribute a page to the ARCHER2 user documentation detailing Singularity as the preferred method of installing Firedrake. We will also create additional documentation detailing how the same procedure can be applied to other projects that have complex dependencies and may wish to use the same procedure to distribute their software on ARCHER2. A report detailing the outcomes of this project will also be generated.

References:
[1] Frings, Wolfgang and Ahn, Dong H and LeGendre, Matthew and Gamblin, Todd and de Supinski, Bronis R and Wolf, Felix, Massively parallel loading Proceedings of the 27th international ACM conference on international conference on supercomputing (2013)

5. Impact and Benefits (max. 1 page)

This project will simplify the Firedrake installation procedure on HPC as well as improve HPC launch performance when running simulations at scale. Our work will reduce the time it takes for a new HPC user starting from scratch to get their Firedrake code running on ARCHER2. The launch utility will further improve the efficiency with which users spend their allocation by reducing Python initialisation overhead.

Firedrake is used by hundreds of users with applications in coastal engineering, numerical weather prediction, geophysical modelling, as well as for developing cutting edge numerical methods, amongst many other applications. Difficulty installing Firedrake on HPC generates multiple requests for user assistance every month. Very few users will yet be up and running on the new ARCHER2 machine, so this is the ideal opportunity to simplify the Firedrake installation procedure those migrating. Firedrake and derived projects are supported by a range of EPSRC/NERC grants, it will enable easier HPC access to members of these existing teams.

We anticipate an enormous reduction in the setup time on HPC. Many projects currently set aside a significant amount of grant time (sometimes up to a month) just getting software running on HPC, which with these proposed changes would no longer be necessary and could be spent on the intended research. This simplified installation will also make Firedrake more accessible and a more attractive choice to other researchers who wish to solve PDEs at scale.

Work done by this eCSE will enable appropriate use of HPC facilities by making it easier to install on any tier of supercomputer. This ensures that researchers can select a machine suitable for their workload, rather than filling ARCHER2 with small jobs or overwhelming an institutional machine just because these machines are known to support Firedrake. Being more standardised, the installation methods are also more likely to succeed on future and experimental HPC systems. It also means a single document detailing HPC installation can be maintained, rather than the many that currently exist for different platforms and become outdated quickly.

The installation containers will also allow the Firedrake team to deliver training directly on ARCHER2. Firedrake already has range of tutorials demonstrating functionality, which are delivered multiple times per year to groups of current and prospective users both around the UK and overseas, normally with the aid of a third party cloud solution. With a containerised install users will be able to run jobs directly on ARCHER2 during the tutorial. Furthermore, after the tutorial participants will have a working installation that they can continue to use. Currently, participants are left to go away and try to install Firedrake themselves after the tutorial is over and after the cloud platform used to deliver the training is shutdown. Tutorials will increase user awareness of the Firedrake framework and containers will largely eliminate the additional step required to move simulations to a larger machine such as ARCHER2.

Firedrake is an open source project with an active development community spanning multiple universities and many active grants. We detail the code availability, maintenance and sustainability in the technical evaluation.