Profiling Firedrake

David Acreman

July 9, 2021

Contents

1 About this document

This document will become the profiling report deliverable. Profiling is deliverable 5.1.1 and builds are deliverable 5.1.2.

Q: What is the scope: are we only considering Firedrake? We could separate the building and profiling parts if the profiling needs to be used for other applications.

2 Target platforms

The target HPC platforms for the project are Archer-2 (national tier-1), Isambard (national tier-2) and Isca (local tier-3). Isambard has two separate systems based on the ARM64 architecture: the XCI system which is an established production system, and the A64FX system which is a newer system. The A64FX system should be considered a stretch objective as the hardware and software are less well known to us. We have also purchased a local server which is designed to be similar to an Archer compute node (although it will run Ubuntu which will make it easier to build Firedrake). An overview of the hardware in the target platforms is shown in Table 1.

System	Processor	Cores/node	Memory/node	Interconnect
Archer-2	AMD x86_64	128	256 GB DRAM	HPE Cray Slingshot
Isambard XCI	Thunder X2 ARM64	64	256 GB DRAM	Cray Aries
Isambard A64FX	Fujitsu A64FX ARM64	48	32 GB HBM	Mellanox Infiniband
Isca	Intel x86_64	16/20	128 GB DRAM	Mellanox Infiniband
Server	AMD x86_64	128	256 GB DRAM	None

Table 1: Target platform hardware specifications. The Isambard A64FX platform has high bandwidth memory (HBM) which should give better performance than DRAM but has a smaller capacity.

An overview of the software stacks on the target platforms is shown in Table 2. The common factor in the compilers is the GNU compiler which is available on all platforms. Each HPC platform has a compiler from the processor vendor (AMD, Intel, ARM and Fujitsu) and Cray systems also have the Cray compiler. On the Isambard A64FX system some modules are only available after

System	Compilers	MPI libraries	Maths libraries	Profilers
Archer-2	Cray, GNU, AOCC	Cray MPICH2	Cray libsci	Cray PAT
Isambard XCI	Cray, GNU, ARM	Cray MPICH2	Cray libsci	Cray PAT, ARM Forge
Isambard A64FX	Cray, GNU, ARM, Fujitsu	Cray MVAPICH2	Cray libsci	ARM Forge
Isca	GNU	OpenMPI	OpenBLAS	None
Isca	Intel	Intel	Intel MKL	Intel Parallel Studio
Server	GNU	MPICH?	???	None

Table 2: Software stacks on target platforms. AOCC is the AMD Optimizing Compiler Collection. Intel MPI and MVAPICH are MPICH derivatives which support an Infiniband interconnect. **Note: I can't find a Cray PAT module on the A64FX- should there be one?**

running

```
module use /lustre/software/aarch64/modulefiles
```

On the Cray systems compilation is handled by wrapper scripts from the Cray Programming Environment. The wrapper script can run different compilers depending on which programming environment module is loaded at compile time (e.g. `PrgEnv-cray` calls the Cray compiler and `PrgEnv-gnu` calls the GNU compiler). The MPI library and maths library are then linked by the wrapper script. On Cray systems the standard profiling tool is Cray Performance Analysis Tools (PAT). On ARM-based Cray systems there is also the ARM/Allinea Forge profiler.

The software environment on Isca is managed using Esaybuild which has the concept of a toolchain. There are two toolchains on Isca: the GCC-foss toolchain and the Intel toolchain. Each toolchain has a different MPI library and maths library. Intel Parallel Studio has an MPI profiling tool called Intel Trace Analyzer and Collector (ITAC) which needs to use Intel MPI.

3 Builds

Plan:

1. Working builds on main target platforms
2. Optimised builds on main target platforms: use optimised maths libraries but beware threading

Status:

- Archer2:
- Isambard XCI: my build is currently failing due to VTK not building.
- Server:
- Isca:
- Isambard A64FX

4 Profiling

Within Firedrake and available on all platforms:

- PyOP2 timed stage
- PETSc profiling

External profilers:

- ARM/Allinea Forge
- Cray PAT
- Intel Parallel Studio

Plan for testing external profilers:

1. Test with a simple example e.g. Mandelbrot
2. Test with Python
3. Test with Firedrake

What to test? Could look at whether the profiler can spot load imbalance vs. overhead and identify the MPI call responsible.

4.1 Cray PAT

Although this is a Cray tool it does work with compilers other than Cray (the other PrgEnv modules load perftools-base). Experience building SWIFT showed that the `perftools-lite` module can confused autogen and configure so only load the module prior to the make command.

- Cray PAT has two modes: standard and "lite"
- There is a graphical viewer (Apprentice 2) which reads the profiling output
- Can view a time line from a full trace

4.2 ARM Forge

4.3 Intel Parallel Studio