Performance Characterisation and Optimisation in the Supercomputing Wales Project

Ed Bennett

Summary

Supercomputing Wales funded 15 RSEs across four universities, with a mix of different management models. It was found that central RSEs were better placed to perform technical interventions into software, while RSEs managed within research groups would tend to prioritise performing (software-enabled) research. Of the groups with central RSE resource, only Swansea University had sufficient critical mass to spend time developing and applying a methodology for performance analysis. While the approach was flexible from project to project, it typically involved an initial phase of basic parameter tuning (selecting the most performant compilers, compiler flags, and run-time options), followed by work with a profiler to obtain a roofline model and a listing of hotspots. Discussion with the team translated this into recommendations for performance engineering, including areas to focus on, parallelisation patterns which the software would tend to lend itself to, and prerequisites that would need to be in place before the code could reasonably be worked on (for example, version control and a canonical working copy of the software, and a basic suite of regression tests).

Background

Supercomputing Wales was a £15M investment by the Welsh Government, the European Regional Development Fund, and four Welsh universities (Aberystwyth, Bangor, Cardiff, and Swansea), to provide the first national research facility for Wales. It funded two hardware hubs in Cardiff and Swansea, with a total capability on the same level as an EPSRC Tier 2 system; project and systems staff; and additionally a distributed team of up to fifteen Research Software Engineers to support researchers in making optimal use of the system. The project was initially funded from 2015 to 2020 (with funding awarded retroactively from 2017), with no-cost extensions to the end of 2022. RSEs were recruited in 2017, and the hardware hubs deployed in 2018. The last RSEs finished working on the project in 2022; the hardware hubs are now in the process of being replaced.

The recruitment and management of RSEs was delegated to each of the partner institutions, and each adopted a slightly different model:

Institution	Number of RSEs at peak	Model
Aberystwyth	1	Central
Bangor	2	Central + Embedded
Cardiff	6	Embedded
Swansea	6	Central

All RSEs were recruited to have research experience in one or more of the areas for which their institution used HPC. Embedded RSEs typically sat within a single research group, lying very much on the research end of the research–software engineering spectrum, and were line managed within that group. Central RSEs sat closer to the middle of the distribution, with significant research experience but focusing primarily on software performance aspects in their day-to-day work.

The author was co-lead of the Supercomputing Wales Swansea RSE team for the majority of the funded period. The primary aim of this document is to summarise the approaches to software performance characterisation that were deployed during the Supercomputing Wales project. Some of this comes from memory or from documentation retained within Swansea from this period. The remainder comes from additional conversations with other former Supercomputing Wales RSEs and team members, specifically:

- Dr Colin Sauzé, former Aberystwyth RSE, now Senior RSE at National Oceanographic Centre
- Professor Martyn Guest, Technical Director of Supercomputing Wales, Cardiff University

I am grateful to them taking the time to provide their input. While they have had the opportunity to review a draft version of this document, any errors in the it remain the author's.

Cross-project activities

Supercomputing Wales RSEs benefited from training provided by suppliers as part of the "community benefit" associated with the EU procurement. This included packages from Intel, NVIDIA, Atos, and Mellanox. Some of this focused on performance characterisation, including examples from industrial applications where vendors had previously supported this.

There were additionally cross-project performance hackathon events. One of these targeted the RSE team itself and their active projects, and was supported by vendors as part of the community benefit programme. Others were associated with training events run by the RSE team, with RSEs supporting users in profiling and optimising their own software, in order to gain practical experience with the skills developed during the training.

Aberystwyth

Having only a single RSE to serve the HPC needs of the entire university, including acting as the primary liaison between the Supercomputing Wales technical team and the research

community, as well as onboarding new users and providing training, the amount of performance characterisation work that was taken on as part of the project was relatively limited.

Much performance characterisation was reactive: reports generated centrally by the technical team in Cardiff would flag projects as having low CPU efficiency; the RSE would investigate using low-level tools such as sacct, and identify options for basic improvements in the job setup to improve its throughput. A similar workflow was followed when users requested support in improving their software's performance; an interrogation of sacct along with a discussion of the user allowed basic suggestions to be made. In some cases this was followed up with some work on improving performance from the RSE side[1]; in some cases this included profiling the software, while in others trivial parallelism was implemented using tools such as GNU Parallel.

In a few cases there were more urgent situations raised by the system administrators; for example, where one user's job created a million files, degrading the performance of the Lustre filesystem.

Training was delivered on tools and techniques for improving parallel performance, including basic techniques such as Slurm and GNU Parallel, theoretical tools such as Amdahl's law, libraries such as MPI, and how to measure performance. It is unlikely that any new MPI applications were written following the MPI training; however, a better understanding of the fundamentals allows more effective running of existing MPI software.

Additionally, Cron scripts were used to monitor user project performance weekly, reporting on the efficiency of the most intensive system users, and a Grafana dashboard was used to monitor GPU utilisation (but not to optimise based on this).

Bangor

Supercomputing Wales funded two RSEs at Bangor: one to serve the needs of the bioinformatics community, and a second to cover the rest of the institution's HPC-utilising research. From the content reports submitted to the Supercomputing Wales Infrastructure Committee, the former's activities were similar to those of the RSEs discussed under Cardiff below, focusing almost exclusively on research, while the latter's were similar to the Aberystwyth RSE's. In neither case was significant performance characterisation or analysis work documented; the latter RSE did spend effort on performance engineering of some new HPC software for the Bangor user community, in addition to more numerous, shorter pieces of onboarding work.

[1] A typical challenge in this scenario was getting a runnable example from the user

Cardiff

Cardiff recruited six RSEs to support the Supercomputing Wales project, with recruitment and line management delegated to the individual research groups most heavily using HPC. Two RSEs were recruited within Chemistry, one in Life Sciences, one in Computer Science, one in Physics, and one in Engineering.

While in principle it was agreed that 20% of each RSE's time would be spent supporting activities within their school outside of their research group, the degree to which this was achieved in practice was limited—instead, the activities of the RSE almost entirely aligned with the interests of their PI, except for presence at cohort training events and some administrative work undertaken by one or two RSEs. As a result, there is little remaining cross-institutional knowledge of the practices employed by the RSEs; this either remained within the research group, or left the institution with the RSE.

Reviewing documentation prepared during the project—including reports prepared for the Supercomputing Wales Infrastructure Committee and for the Welsh European Funding Office—there is little record of any performance characterisation work being performed by Supercomputing Wales RSEs at Cardiff. There is one mention of performance analysis of a radial solver within KKR Green's function formalism, but no record of the result of this analysis or the methodology used for it remains. A handful of pieces of performance engineering work—for example optimisation of MPItrampoline, improving the distributed- and shared-memory parallelism within Molpro—are also discussed. One RSE, in Engineering, has performed comparison benchmark studies both of machines and of software tools; however, these were not documented in the reports, and were only published after the RSE had left the institution.

Professor Martyn Guest, Technical Director of Supercomputing Wales at Cardiff University, and member of the SHAREing board, has substantial experience of performance characterisation of machines both before, within, and since the funded period of Supercomputing Wales. This was discussed in depth in the presentation given at the SHAREing Performance Assessment Seminar on 2025-11-07; to avoid repetition, it is not presented further in this document.

Swansea

There were two primary reasons that the Swansea RSE team performed performance characterisation studies.

The first type of study was in support of applications for HPC time or other similar funding, where the exercise was to demonstrate the current performance of the specified software on tasks close to what would be performed in the final project. A secondary aim of such studies was then to identify the optimal parameters with which to compile or run the software to

maximise the performance. (This is limited to the software environment; adjusting algorithmic or scientific parameters is outside of the scope of the exercise.)

The second kind of study is in preparation for potential optimisation work on the software. Projects with high use of the cluster were approached directly to perform a study of their software, first as a sense check that it was being run efficiently (similarly to the secondary aim of the study supporting HPC time applications), and then to identify possible opportunities to optimise the code. Projects could also approach the team directly seeking assistance in making their code faster; a performance report was a first step in this direction.

A standard template was prepared for these, in both Microsoft Word and LaTeX formats; these are attached to this document. This provided a skeleton of headings for the RSE performing the work to consider in their study, as well as providing consistent aesthetic styling to reports generated by the team.

Initial tests

The first stage in either kind of study was to obtain from the research group one or more sample problems that would run in reasonable time (ideally minutes, up to 1–2 hours), while still demonstrating the characteristics of problems that would run for longer (so would be expected to scale similarly). These were typically requested in the form of input files and/or scripts to run the software, as well as a link to the code itself, and currently-used compile parameters where available.

From here, the software was compiled on the target architecture; typically the Supercomputing Wales SUNBIRD[2] machine in Swansea. Where reports were in support of applications for machine time on other facilities, and where we had access to said machine, we would use it; otherwise, we would use SUNBIRD and make projections for performance on comparable systems.

Since time to optimise code is long, and there are frequently quick wins that can be had by changing compilers or compiler options, the next step after ensuring that the software could be compiled and run was to study the effect of changing these options. While a complete exhaustive search is not possible in reasonable time, the focus was to try both Intel and GNU compilers, both Intel MPI and Open MPI for MPI-enabled code, and to try an older and a newer[3] compiler version than the initial test version.

[2] Atos/Dell EMC CPU cluster comprising 123 nodes each with 2x Intel Xeon Scalable Gold CPUs, 384GB RAM, 1x Mellanox EDR Infiniband, linked with 3:1 blocking fabric, and sharing 1PB of storage.

[3] In one case, the developers of a piece of software had identified that only older versions of the Intel Fortran compiler could be used, which we confirmed in observations. Subsequent details analysis showed that this is because the code—seemingly deliberately—relied on undocumented behaviour in these versions where out-of-bounds access would return zeros rather than giving arbitrary memory contents or throwing errors. This fault was fixed.

From there, individual compiler and runtime options could be probed. For serial code, using the compiler's automatic parallelisation was tested. For parallel code (including auto-parallelised code), scaling tests were performed—either strong, weak, or both, depending on the problem in question. Tests of parallelism included adjusting process pinning, checking if underpopulating nodes improved performance, and adjusting the distribution of work between processes (for example, 2×2 vs 1×4 parallelisation). Based on these results, an interim report was shared with details on optimal compile and runtime options.

Detailed tests

The next step in the process was to profile the code with suitable tooling. The exact tools used depended on the experience and appetite to learn new tooling of the individual RSE; options used included Intel's VTune Amplifier and (Vector) Advisor, the open-source Score-P, Scalasca and Tau tools from LLNL and Jülich, and Arm MAP[4]. In general, a call graph with timings and a hotspot analysis were the first instrumentation steps performed. A roofline model was typically also generated to understand the constraints on the code's performance, and from this and line profile information, optimisation targets were identified. For MPI-parallelised codes, Intel MPI Profiler was also used to study the communication patterns and potential bottlenecks.

Code readiness

In parallel with this work, where projects were requesting optimisation support, we also identified (and reported on) aspects of the software that would be a barrier to the team performing work on it. Suggestions for remediation were given, and an indication of whether this would need to be performed before the team would touch the code, or whether the team could assist with it.

For example, we were approached by a user with a 10kLOC Fortran file, of which 7kLOC comprised commented-out code (that sometimes required uncommenting), and for which there was no automated testing. Discussion with the user indicated that there were also other divergent versions of the file with additional features not present in the version we were given access to. In this case, we indicated that we could not work on the code until a single version was prepared (since the difficulty of porting our changes to other versions would make it unlikely they would be used in production), that commented code would need to be refactored into separate functions instead, and that we would need to work together to build a basic suite of regression or unit tests before we could start optimising (so that we would know if our changes broke any aspect of the implementation).

[4] Formerly Allinea MAP, now Linaro MAP.

Record keeping

All performance work, including performance reporting, but also optimisation and shorter responses to user queries, were kept in a private GitHub repository to which all team members had access.

System monitoring

In addition to longer performance reporting work, we additionally monitored efficiency[5] of the system utilisation on a weekly basis. Projects with an efficiency below 40% were investigated in more detail by interrogating Slurm's accounts database, as were high-usage projects with efficiency such that significant numbers of core-hours were being spent idle.

Where the problem was obvious from the `sacct` output, the investigating team member would send a friendly email asking that the user try making a corrective change. Where it was less obvious, we would ask for a meeting with the user to look in more detail at the possible causes. The most frequent cause was using `ntasks` to request multiple cores for a shared-memory computation; the system setup meant that these tasks were frequently spread across nodes, so processes ended up thrashing on the first allocated node while leaving others idle.

It was vital to ensure that the contact remained as friendly and welcoming as possible; we found that users were predisposed to see any communication on these lines as chastisement at using the machine inefficiently and wasting resources (and so avoid using the machine further), whereas our main aim was to ensure that the researchers got maximum throughput for their work, while also avoiding disruption to other users.

[5] Defined on a per-job basis as CPU time / Wall clock time, normalized by the number of nodes. A program that only invoked `sleep` would have an efficiency of 0, while a program that spins all allocated cores for the full length of the job would have an efficiency of 1.