EU H2020 CoE Performance Optimization and Productivity (POP) Successfully Finished

From October 2015 to March 2018, the EU Horizon 2020 Center of Excellence (CoE) for Performance Optimisation and Productivity (POP) provided performance optimisation and productivity services for academic and industrial codes. Europe's leading high-performance computing experts helped application developers get a precise understanding of their respective applications' and systems' behaviour. Both established codes and codes which had never undergone any analysis or performance tuning profited from POP services, which used latest state-of-the-art tools to detect and locate bottlenecks in applications, suggested possible code improvements, and even helped with proof-of-concept experiments for customer codes on their own platforms.

Today's complexity of high-performance computer systems and codes makes it increasingly difficult to get applications running fast and efficiently on the latest hardware. Often expert knowledge and a good amount of experience is needed to figure out the most productive direction for code refactoring. Many domain experts in in industry and academia use computer simulations, but lack this knowledge.

As a result, their codes are often far away from using the hardware efficiently, using much more compute time than needed. This lack of optimization can waste energy, require superfluously oversized and expensive hardware, or just miss research potential, as their codes can only handle smaller or less complex problems in the available amount of compute time.

To overcome this situation, the POP CoE brought users a service that tightly couples two disciplines crucial for the efficient use of parallel computers in the future: First, powerful performance analysis tools, methodologies, and expertise needed to precisely understand and gain real insight into the actual application and system behaviour as well as a deep understanding of programming models and best-practice guidance needed to express algorithms in the most flexible, maintainable and portable way, while still being able to maximise the performance achieved.

POP Services

The POP CoE team comprised six partner organisations with experts in high-performance computing with long-standing experience in performance tools and tuning as well as researchers in the field of programming models and programming practices. All partners have a research and development background and proven commitment to applying their know-how to real academic and industrial use cases. The POP CoE provided three kind of service levels to its customers – depending on their background, knowledge, and demands:

?: Application Performance Audit

This was the primary service of the POP CoE and the starting point for any further work. Applications using this service were analyzed by POP experts after an initial discussion with respect to their best practices and provided a first impression of the code status. Within the Performance Audit, the customer's code performance issues could be identified at the location a customer would normally run his or her code. It also served as a starting point for further analysis or initial code refactoring. The duration for a Performance Audit averaged around two months and a successful Performance Audit may be seen as a code quality certificate in HPC.

!: Application Performance Plan

The Performance Plan service followed the Performance Audit when the customer needed more detailed knowledge where and how to address specific issues in the code. POP experts developed together with the customer a plan how and with which tools to analyse the issues under investigation. The POP experts then analysed the code in detail and gave quantified advice to overcome the problems that could be fixed by the customer. The duration for a Performance Plan is very problem-specific, but always included a closer look into the source code.

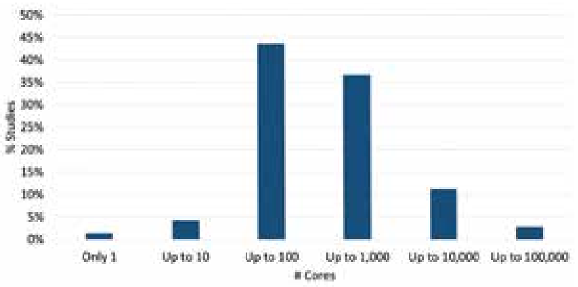
√: Proof-of-Concept

When requested, proof-of-concept studies were performed. This included experiments and mock-up tests for customer codes. The details of the proof-of-concept study were decided in very close collaboration with the customer and could include kernel extraction from the application, parallelisation, or mini-apps experiments to show effects of the POP experts' proposed optimisations. As this very complex task goes into deep detail, proof-of-concept work sometimes required about six months.

Besides the above three key services, the POP CoE also provided a variety of training activities in the field of performance analysis and optimisation to improve basic high-performance programming knowledge and increase the awareness of performance issues and potentials in general.

Codes Analyzed

In its 30 months of operation, POP has undertaken over 150 assessments of codes drawn from a wide range of scientific domains covering astronomy, chemistry, Earth science, energy, engineering, health, mathematics and physics.



Although the bulk (80%) of POP studies have looked at codes that run best on more than 10 but less than 1,000 cores, POP also performed assessments on larger scale codes over 100,000 cores.

Roughly half of the assessments originated from academic institutions, a quarter from research or government laboratories, and a quarter had an industrial background.

Languages and Parallelism

As one might expect, Fortran codes dominate, with over half the studies (82 of 151) written either entirely in Fortran or Fortran combined with C and/or C++. C++ seems more prevalent than C, but C is more likely than C++ to be combined with Fortran:

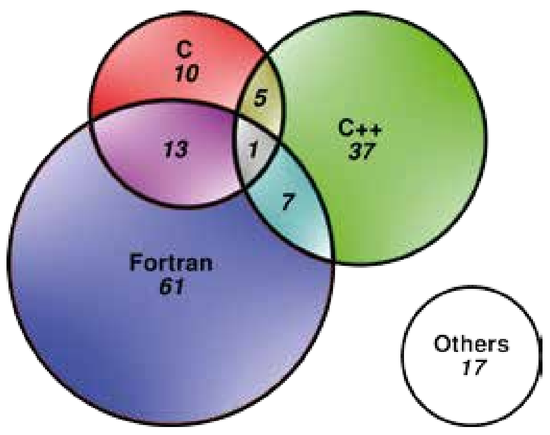


Fig.2: Programming languages used by POP customer applications. The areas of the circles in the Venn diagram are proportional to the number of studies they contain.

The "Others" category makes up about 10% of studies. Of these, 13 involved Python, either stand-alone (3) or in conjunction with one or more compiled language, and the remainder were a combined C/Fortran/Octave code, a Java code, a Matlab code, and a Perl code. The fact that 10% of codes that POP assessed are written in languages other than C/C++/Fortran demonstrate that it's important to have tools and methodologies capable of handling a wide range of languages and not just the usual suspects.

A similar situation arises looking at the types of parallelism used by the codes studied by POP. MPI is the most common form of parallelisation, with nearly 80% of codes using either pure MPI or MPI+OpenMP. Over a third of codes POP has assessed are hybrid combining MPI+OpenMP.

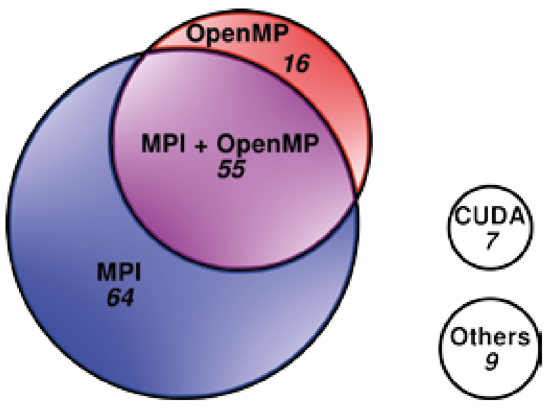


Fig.3: Parallelization paradigm used by POP customer applications. The areas of the circles in the Venn diagram are proportional to the number of studies they contain.

7 codes could also use CUDA and there is again an „Others" category, but unlike the corresponding category for the languages every member of this set is unique. Examples of the other types of parallelism we encountered include Intel Threading Building Blocks (TBB), C++ threading and Coarray Fortran.

Causes of Low Efficiencies

The POP efficiency metrics [1] provide a methodology for characterising the performance of parallel codes and for providing insight into where the most pressing problems lie. Looking across all the POP studies we can use the metrics to discern whether there are any overarching performance trends.

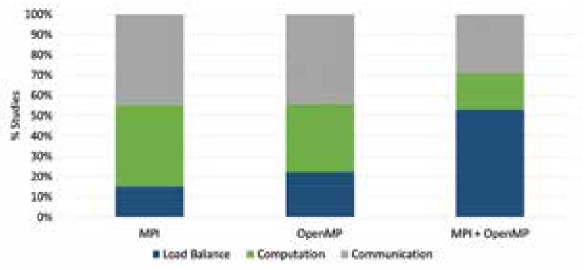
Of the analysed codes, 66% had a Parallel Efficiency less than 80%, meaning that they typically required improvement to run efficiently in parallel. Indeed, 22% of codes had Parallel Efficiency below 50%, which means that less than half of their runtime is dedicated to computation. Note that analysis generally omits initialisation and finalisation, so in practice their efficiency is even worse.

Looking at the actual numbers reported in the studies, we find that Load Balance Efficiency is often either very good or very bad. This suggests that load balance is something users must ensure is done correctly or else can have significant impact on efficiency, particularly when scaling to larger numbers of cores.

We can also use the hierarchical nature of the metrics to look at the common underlying causes of low efficiencies. Low Communication Efficiency is mostly caused by data transfer (high volume of data or high number of communications) rather than serialisation of communication. Low Computation Efficiency is often caused by poor instruction scalability rather than reduced instructions-per-cycle IPC values; when strong scaling, growth in the total number of instructions executed often corresponds to undesirable code replication.

POP staff can look further into the categories of problems to see if there is a link between programming approach and the types of problems. Although there was no obvious correlation between language and inefficiency (e.g. we couldn't conclude things like "C programmers were more likely to write badly load-balanced code"), there was an interesting distinction to be drawn based on the type of parallelism the code employed.

For each study we recorded the main cause of inefficiency that was identified (i.e. load balance, computation or communication) and looked at how this varied across the three main types of parallelism:



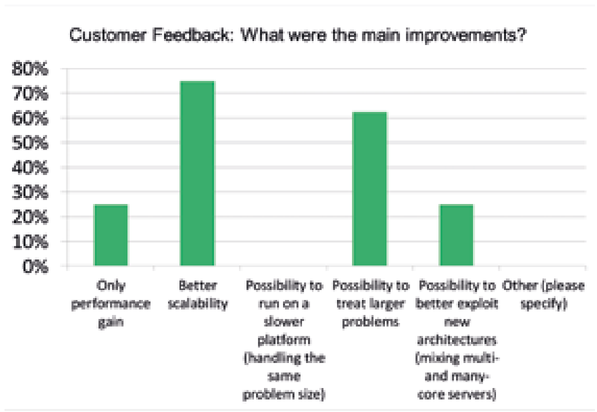
From the graph, we can see that studies of hybrid codes were much more likely to report problems with load balance than studies of pure MPI or pure OpenMP codes. This is perhaps understandable—when writing hybrid code, users need to take into account both how the work is divided across MPI ranks and also how it is split up between threads of a process.

Results

The POP project was very successful: more than 90% of the customers were either very satisfied or satisfied with the service they received. More than half of the customers of a Performance Audit requested a follow-up service. More than 2/3 of the customers with Performance Plans indicated they plan to use POP services again.

The proof-of-concept studies, where POP experts worked together with the developers on improving their codes, were especially successful: in many cases, they were able to demonstrate a doubling of the performance and/or scalability of the codes investigated. In some cases, a six- to ten-fold improvement could be implemented. A more detailed description of these accomplishments can be found on the POP blog under the tag "success stories" [2].

In the case of Performance Plans, where developers improved their codes themselves based on the recommendations given by the POP experts, they reported a 25% performance or scalability improvement on average, in some cases even 50% to 70%, allowing them to treat larger problems or better exploit new architectures. Customers also reported that, in most cases, only a few days' effort was necessary to perform this work; the remainder required either a few weeks' or a few months' effort.



The return-on-investment (ROI) in these cases are enormous, as the following two examples demonstrate: In the first case, where an application running on the UK national academic supercomputer (Archer) was first analyzed and then in a proof-of-concept study, had a 72% improvement in time-to-solution could be implemented. The saving in compute time for a typical run of this code was €15.58, which resulted in a yearly saving of €56,000 for this specific customer based on their monthly usage data. In the other example, the customer reported that the costs for implementing the recommendations of the POP experts were €2,000 and resulted in a 62% performance improvement. By this, €12,400 of the customer's annual €20,000 operating cost could be saved, resulting in a ROI of 620%.

Impact

The POP CoE, with its service and training activities, had a wide impact within all areas of research and industry, making it a real transversal activity:

• It provided access to computing application expertise that enables researchers and industry to be more productive, leading to scientific and industrial excellence.

• It improved competitiveness for the centre's customers by generating a tangible return-on-investment (ROI) in terms of savings, elimination of waste, errors, and delays by making their applications leaner and issue-free.

• As the centre represents European world-class expertise in this area, its deployment strengthened Europe's leading position in the development and use of applications that address societal challenges or are important for industrial applications through better code performance and better code maintenance and availability.

• The centre's services included training on the use of computational methods and optimisation of applications; especially successful were webinars [3].

It is planned to continue the POP CoE project for additional three years (2019 to 2021). A proposal was prepared and submitted by the partners for the call INFRAEDI-02-2018: HPC PPP – Centres of Excellence on HPC.

Partners

Barcelona Supercomputing Centre (BSC), High-Performance Computing Center Stuttgart of the University of Stuttgart (HLRS), Jülich Supercomputing Centre (JSC), Numerical Algorithms Group (NAG), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), and TERATEC.

Timeframe

October 2015 – March 2018

POP Coordination

Prof. Jesus Labarta, Judit Gimenez Barcelona Supercomputing Center (BSC)

Email: pop@bsc.es

Web: https://www.pop-coe.eu

Acknowledgements

This project was supported by the European Commission under H2020 Grant Agreement No. 676553

References

[1] https://pop-coe.eu/node/69

[2] https://pop-coe.eu/blog/tags/success-stories

[3] https://pop-coe.eu/blog/tags/webinar

Written by Bernd Mohr, Brian J. N. Wylie
Jülich Supercomputing Centre (JSC)
Contact: b.mohr@fz-juelich.de, b.wylie@fz-juelich.de

José Gracia, Christoph Niethammer
University of Stuttgart (HLRS)
Contact: gracia@hlrs.de, niethammer@hlrs.de