Performance Optimisation and Productivity

A Centre of Excellence in Computing Applications

POP Newsletter 8 – Issue March 2018

Welcome to the eighth newsletter from the EU POP Centre of Excellence. This is the last newsletter for the first phase of POP as funding for this phase will end on 31 March 2018. For new requests, please see section Free Code Optimisation Help at the bottom of this newsletter.

This issue includes:

- POP Webinar - **Impact of sequential performance on parallel codes** on 28th March 2018 2PM GMT | 3PM CET;
- POP Lead Jesus Labarta Summarises the POP project;
- Sectors that Benefited from POP;
- Parallel Models and Programming Languages Used by Codes that Benefited from POP;
- Performance Profiling Education;
- POP member co-authors new book "Using OpenMP - The Next Step: Affinity, Accelerators, Tasking, and SIMD".

For information on our services and past editions of the newsletter see the POP website.

POP Webinar – Impact of Sequential Performance on Parallel Codes

Wednesday 28th March 2018 - 14:00 BST | 15:00 CET

The analysis of parallel programs often focuses on the concurrency achieved, the impact of data transfers, or the dependences between computations in different processes. In the previous POP Webinar #3 we looked at how to characterize such aspects of application behaviour and provide insight on the fundamental causes of parallel efficiency loss. The total performance and scaling behaviour of applications depends not only on its parallel efficiency, but also on how the sequential computations between calls to the runtime change with increased core count.

This webinar will first present a model characterizing the evolution of the performance of the user level code between calls to the runtime (MPI or OpenMP). We will show a methodology based on three metrics derived from hardware counter measurements that gives a global view of the scaling of sequential computations and how it impacts the overall application scaling. The webinar will then show how advanced analytics can be used to identify the regions in the code with different sequential cost and performance, and how it evolves when scaling core counts. Finally, we will show some analyses examples of the insight that can be obtained with these techniques.

In this 30-minute live webinar we will:

- Discuss the impact of sequential performance in parallel codes
- Present three metrics derived from hardware counter measurements

- Introduce some advanced analytics
- Show examples of the insight this analysis provides

Click here for more information and to register for the Impact of Sequential Performance on Parallel Codes webinar.

POP Lead Jesus Labarta Summarises the POP Project

The project closes, but the impact continues.

In Q4 2015 the European Commission funded the POP project with the objective to set up a parallel performance analysis Centre of Excellence (CoE). The 2.5-year project is coming to an end, having completed more than 125 assessments of parallel application performance studies and proof of concept demonstrations on how to achieve higher application performance.

The project has refined and promoted a high-level analysis and modelling methodology. Performance reports were produced giving an external assessment that provides insight into the performance characteristics of codes developed by parallel application developers. A high-level model quantifies factors such as load balance, serialisation, transfer or computational efficiencies providing fundamental insight on how application performance scales and recommendations were made on most appropriate directions to refactor the code to improve the application performance.

Service orientation has been a corner stone of the Centre of Excellence activities. Customers from different scientific and technical domains benefited from this service. Researchers in public institutions (e.g. universities and research laboratories), industry and SMEs were targeted to demonstrate that efficient parallel computing is valuable in all scientific and technical sectors. Around 25% of our studies are for SMEs in our attempt to promote performance analysis in non-traditional sectors.

Improving awareness of the importance of understanding application performance and promoting programming best practices requires cultural change with important benefits in the long term. We believe that the POP project has contributed significantly in this direction. Many of the insights and suggestions provided have been very well received by the customers. In many cases, performance gains between 10%, or 10x or 20x have been achieved by the proof of concept service or through code optimisations by the developers themselves. In cases where not all POP recommendations can be implemented in the short term, they will have an impact in steering the activities of code developers in the coming months and years.

We have the satisfaction of seeing the effort of our analysts has been very well received by our customers. The customer advocacy activities within the project allow us to closely monitor how customers perceive our services and has been an important tool in improving our procedures.

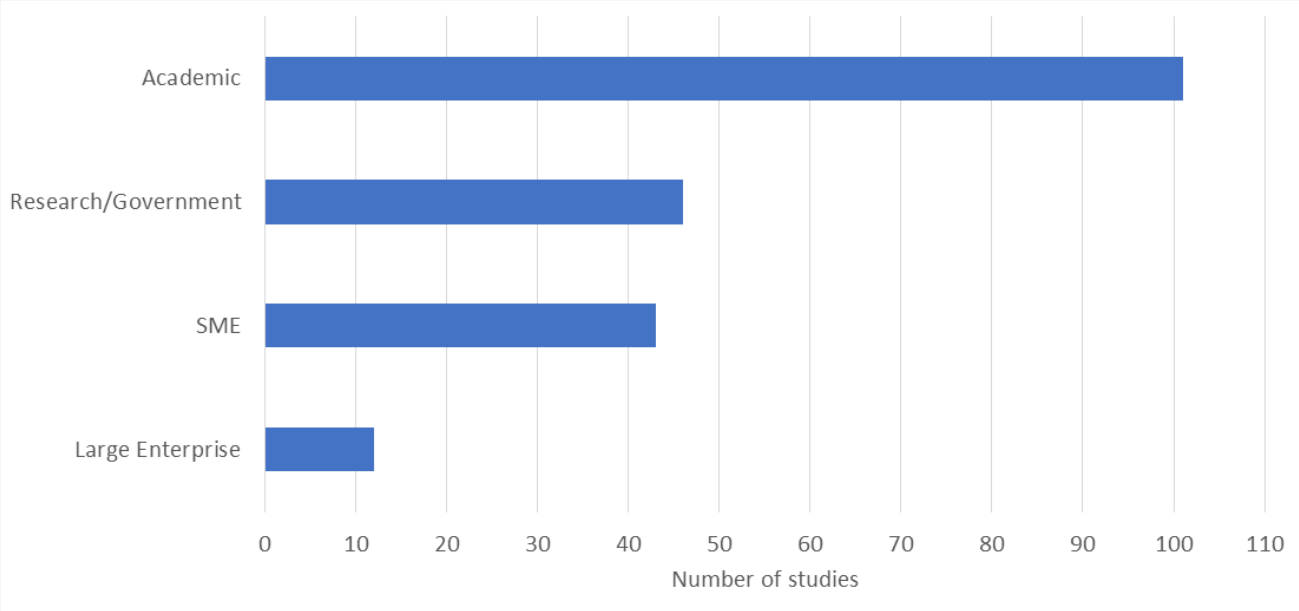
The POP CoE will continue its activities during the next months with limited resources. We expect to be able to continue them in the future under different programmes, supporting the improvement of productivity and overall efficiency in how research and industry sustainably use the available computing resources. To keep up to date with POP activities, please check our website:

https://pop-coe.eu/

Whilst the project has come to a completion, its impact in different dimensions will continue.

Sectors That Benefited From POP

The bar chart below shows the sectors that have benefited from the POP service.

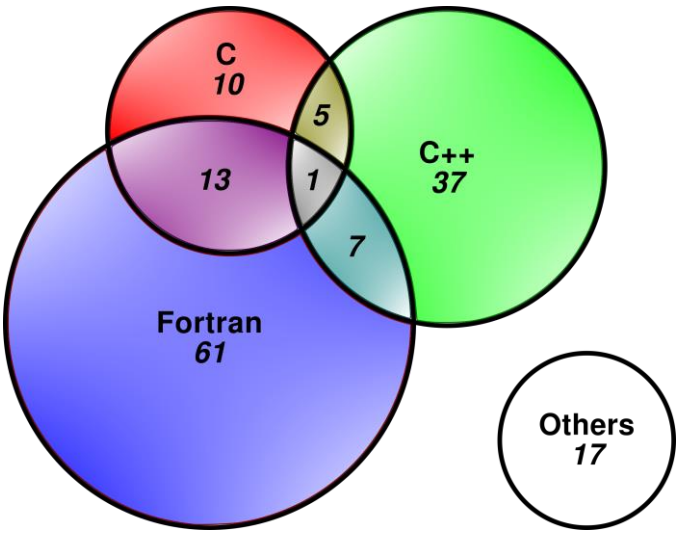


As many of the project partners had wide connections with academia, academics were able to take advantage of the service. The project embarked on a marketing campaign with a dedicated business development manager to target commercial organisations which resulted in attracting a number of customers from this sector. The project also had a sizeable number of customers from research and government organisations.

Some commercial organisations, quite rightly, worry about intellectual property (IP) and their IP not being shared with their competitors. To allay their concerns, the project partners signed non-disclosure agreements with customers when required. We aim to attract more commercial organisations and to build on the success and experience in the future of POP. In addition, the project also aims for wider collaboration with other EC funded HPC projects to share experiences and best practices.

Parallel Models and Programming Languages of Codes that Benefited from POP

POP analysed a wide variety of parallel codes, from commercial CFD to academic molecular dynamics codes. The codes were also implemented in different programming languages and employing several parallel models. The Venn diagram below shows the breakdown of programming languages used:

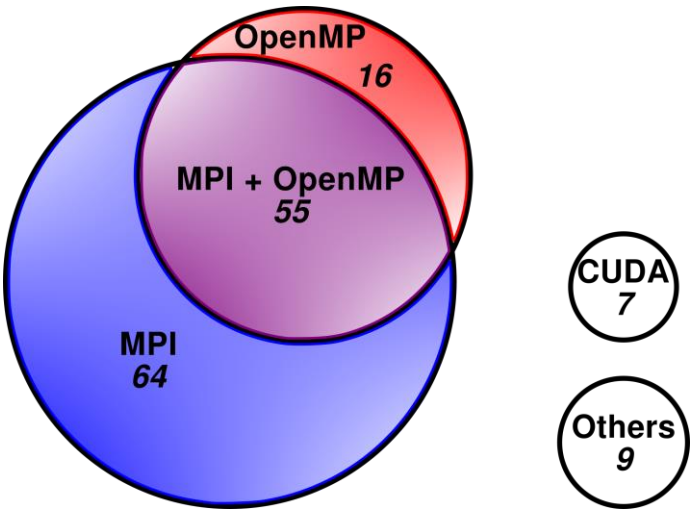


The overlap of the languages indicates codes using more than one language, e.g. Fortran & C++. As can be seen, a large majority of codes are implemented in Fortran, followed by C++ & C. The other languages are C & Fortran & Octave (1), C & C++ & Fortran & Python (1), C & Python (1), C++ & Python (5), Fortran & Python (3), Java (1), Matlab (1), Perl (1), and Python (3). As POP analysed codes mainly from computational science and engineering, it is not a surprise that Fortran is still a widely used language. The diagram also shows an increased uptake in the C++ language amongst the computational community.

In computational codes, Python is generally used in conjunction with compiled languages such as C and Fortran to give it the performance boost. Although there are newer languages such as Julia, the traditional compiled languages are here to stay for some time.

The profiling tools' support for compiled languages such as C, C++ and Fortran are very mature, and support for Python is now available in Paraver. However, limited profiling information can be gained from other languages such as Java, Matlab, Julia, and Perl, and detailed hardware counter metrics for these languages are not available. Hardware metrics includes cache usage statistics, branch misprediction, and vector instructions. Therefore, it is important to carefully choose a suitable language for parallel code development.

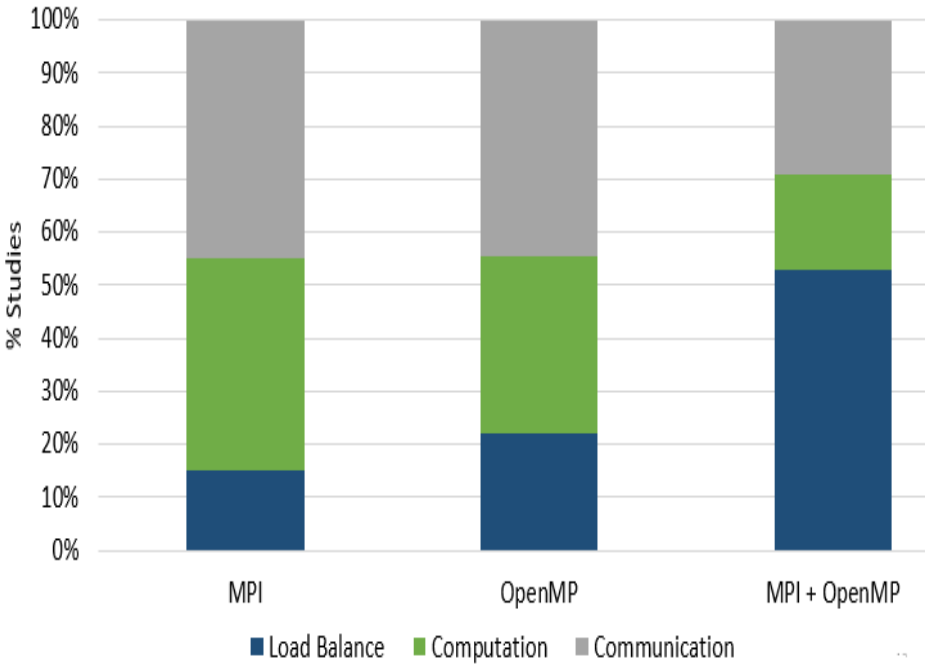
The parallel programming model used is shown below.



Not surprisingly, MPI is the dominant parallel model and there are a large number of hybrid parallel codes, namely MPI & OpenMP. The other models are MPI & Pthreads (1), Pthreads (1), Coarray Fortran (1), TBB (1), C++ threads (1), Java (1), Matlab (1), Perl (1), and Python (1).

Although PGAS models such as coarray Fortran are being promoted, MPI and OpenMP are probably here to stay for some time as both standards are constantly being modernised for larger scale parallelism which include asynchronous communication.

The chart below shows the performance issues identified for the MPI, OpenMP and MPI & OpenMP codes.



The communication category of OpenMP is serialized execution due to synchronisation. For MPI codes, communication transfer is an additional performance issue. Load balance is a major cause of inefficiency for hybrid MPI + OpenMP codes. The load imbalance in OpenMP and the load imbalance in MPI seem to accentuate the hybrid load imbalance. Or, the load imbalance in hybrid codes can be caused by large serial execution in the MPI process.

There was no discernible correlation between the programming language, parallel model and performance issues of the codes. The performance issues discovered are based on the code, and the knowledge and experience of the code developers.

Performance Profiling Education

Another deliverable of the POP project was to promote education and knowledge on the topic of performance analysis and optimisation. To achieve this, the POP project has run five webinars, six workshops at HPC centers and six tutorials at conferences. The webinars attracted over 70 attendees per session with engaging questions from the attendees. The workshops attracted attendees from a wide range of computational disciplines which started with the basics of performance profiling and introduced the various profiling tools, including Paraver and Scalasca. Attendees were encouraged to bring their own code to the workshops for profiling with help provided by experts in performance analysis, including profiling tool developers.

If you have missed any of the webinars or need a refresher, they can be obtained from the POP blog where the slides and a video recording of the webinars are made available:

https://pop-coe.eu/blog/tags/webinar

The training material used in the workshops can also be obtained:

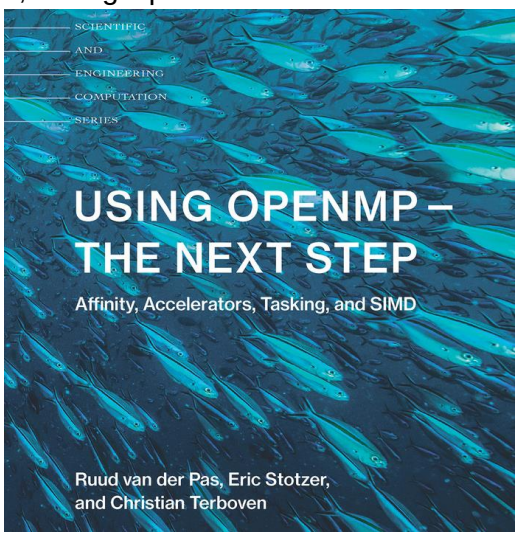
https://pop-coe.eu/further-information/learning-material

Whilst customers have greatly benefited from the skills, experience and expertise of POP experts, ultimately, we would like code developers to also gain these skills, so they can write efficient parallel codes themselves. We encourage all parallel code developers to go through the learning materials and the webinars to further their coding knowledge and skills, and ultimately, include code profiling and analysis in their code development workflow.

POP member co-authors new book “Using OpenMP - The Next Step”

Christian Terboven is the HPC group Manager at RWTH Aachen University (a POP consortium member) and is an active member of the OpenMP standards committee. He has also worked on the POP project and has been an invaluable member of the project with his expertise on OpenMP and performance profiling. Christian has co-authored the book “Using OpenMP - The Next Step”.

This book offers an up-to-date, practical tutorial on advanced features in the widely used OpenMP parallel programming model. Building on the previous volume, “Using OpenMP: Portable Shared Memory Parallel Programming”, this book goes beyond the fundamentals to focus on what has been changed and added to OpenMP since the 2.5 specifications. It emphasizes four major and advanced areas: thread affinity (keeping threads close to their data), accelerators (special hardware to speed up certain operations), tasking (to parallelize algorithms with a less regular execution flow), and SIMD (hardware assisted operations on vectors). The example codes are written in C and C++, and can be easy enough to translate to Fortran.



Further information on the book can be found by clicking on the image or at http://bit.ly/2pvO6BE

Free Code Optimisation Help

The POP project funding will end on 31 March 2018 but the service will continue with limited resources, and new requests will be put on a waiting list. If you have a request and would like to know about timelines (which will depend on workload and available resources), please email us to discuss this further. The POP webinar series will continue and further information on the webinars can be found at https://pop-coe.eu/news/events

We offer a range of free services designed to help EU organisations improve the performance of parallel software. If you’re not getting the performance you need from parallel software, please apply for help via the short Service Request Form, or email us to discuss further.

The POP Helpdesk

Past and present POP users are eligible to use our email helpdesk (pop-helpdesk@bsc.es). Please contact our team of experts for help analysing code changes, to discuss your next steps, and to ask questions about your parallel performance optimisation.

https://pop-coe.eu

This project received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 676553