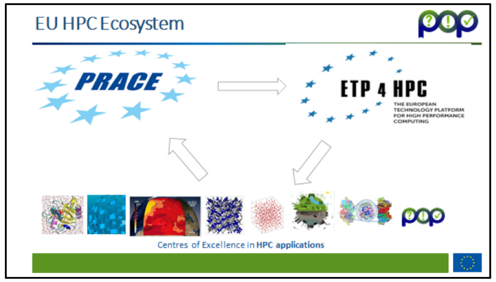
Parallel Performance Optimization and Productivity

EU H2020 Centre of Excellence (CoE)

1 December 2018 – 30 November 2021

Grant Agreement No 824080

POP CoE



- A **Centre of Excellence**
  - On **Performance Optimisation and Productivity**
  - Promoting **best practices in parallel programming**
- Providing **FREE Services**
  - Precise understanding of application and system behaviour
  - Suggestion/support on how to refactor code in the most productive way
- **Horizontal**
  - Transversal across application areas, platforms, scales
- **For (EU) academic AND industrial codes and users !**

Partners

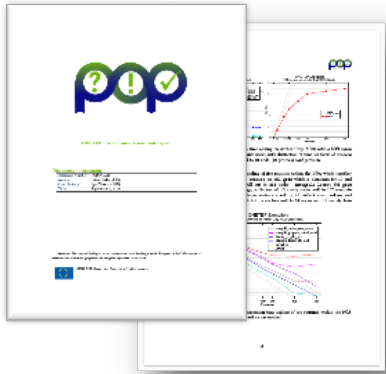
- **Who?**
  - BSC, ES (coordinator)
  - HLRS, DE
  - IT4I, CZ
  - JSC, DE
  - NAG, UK
  - RWTH Aachen, IT Center, DE
  - TERATEC, FR
  - UVSQ, FR

A team with

- Excellence in performance tools and tuning
- Excellence in programming models and practices
- Research and development background AND
  proven commitment in application to real academic and industrial use cases

FREE Services provided by the CoE

- **Parallel Application Performance Assessment**
  - Primary service
  - Identifies performance issues of customer code (at customer site)
  - If needed, identifies the root causes of the issues found and qualifies and quantifies approaches to address them (recommendations)
  - **Combines former Performance Audit (?) and Plan (!)**
  - Medium effort (1-3 months)

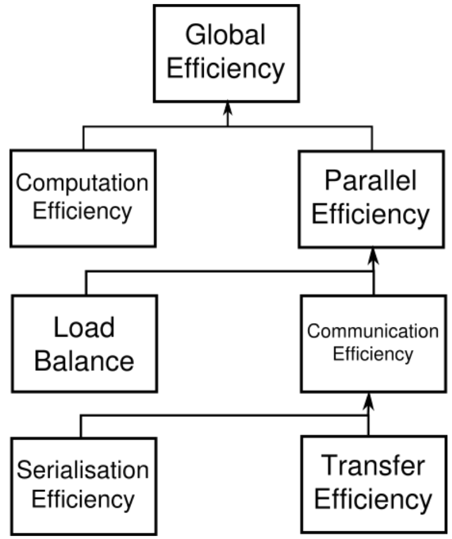


- **Proof-of-Concept (✓)**
  - Follow-up service
  - Experiments and mock-up tests for customer codes
  - Kernel extraction, parallelisation, mini-apps experiments to show effect of proposed optimisations
  - Larger effort (3-6 months)

Note: Effort shared between our experts and customer!

Efficiencies

- The following metrics are used in a POP Performance Audit:
- Global Efficiency (GE): GE = PE * CompE
  - Parallel Efficiency (PE): PE = LB * CommE
    - **Load Balance** Efficiency (LB): LB = avg(CT)/max(CT)
    - **Communication** Efficiency (CommE): CommE = SerE * TE
      - Serialization Efficiency (SerE):
        SerE = max (CT / TT on ideal network)
      - Transfer Efficiency (TE): TE = TT on ideal network / TT
  - (Serial) **Computation** Efficiency (CompE)
    - Computed out of IPC Scaling and Instruction Scaling
    - For strong scaling: ideal scaling -> efficiency of 1.0

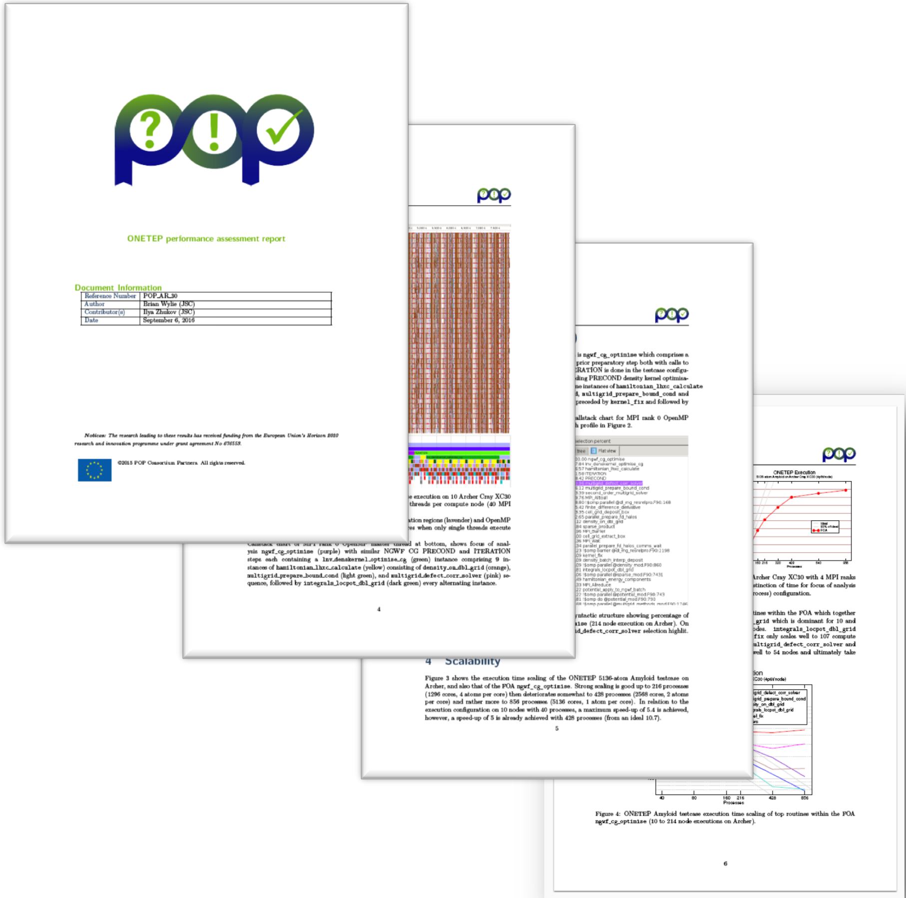


CT = Computational time
TT = Total time

- Details see https://sharepoint.ecampus.rwth-aachen.de/units/rz/HPC/public/Shared%20Documents/Metrics.pdf

Outline of a Typical Audit Report

- Application Structure
- (If appropriate) Region of Interest
- Scalability Information
- Application Efficiency
  - E.g. time spent outside MPI
- Load Balance
  - Whether due to internal or external factors
- Serial Performance
  - Identification of poor code quality
- Communications
  - E.g. sensitivity to network performance
- Summary and Recommendations



Tools

- **Install and use already available monitoring and analysis technology**
  - Analysis and predictive capabilities
  - Delivering insight
    - With extreme detail
    - Up to extreme scale

- **Open-source toolsets**
  - Extrae + Paraver
  - Score-P + Cube + Scalasca/TAU/Vampir
  - Dimemas, Extra-P
  - MAQAO

- **Commercial toolsets**

  (if available at customer site)
  - Intel tools
  - Cray tools
  - ARM tools

Target customers

- **Code developers**
  - Assessment of detailed actual behaviour
  - Suggestion of most productive directions to refactor code
- **Users**
  - Assessment of achieved performance in specific production conditions
  - Possible improvements modifying environment setup
  - Evidence to interact with code provider
- **Infrastructure operators**
  - Assessment of achieved performance
    in production conditions
  - Possible improvements from modifying environment setup
  - Information for time computer time allocation processes
  - Training of support staff
- **Vendors**
  - Benchmarking
  - Customer support
  - System dimensioning/design

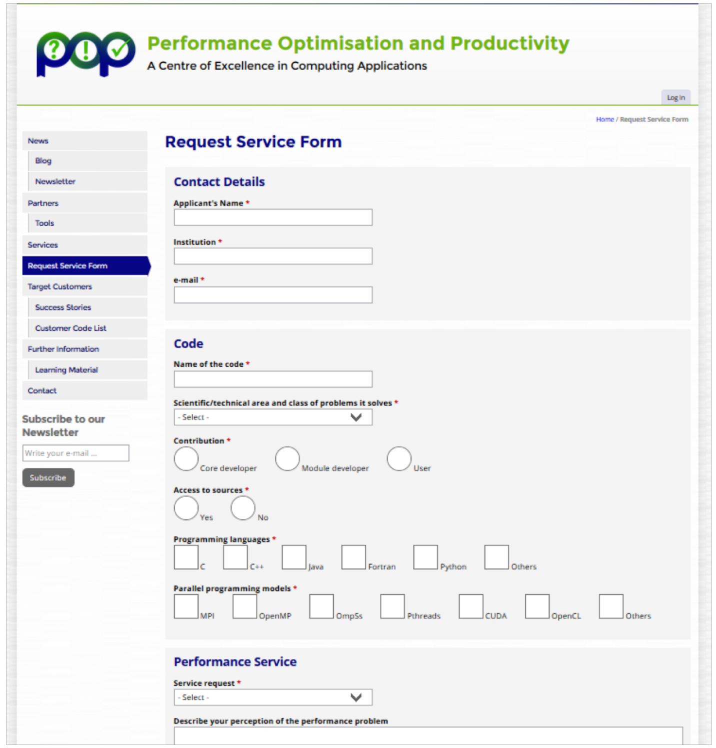
The Process …

When?

December 2018 – November 2021

How?

- Apply
  - Fill in small questionnaire describing application and needs https://pop-coe.eu/request-service-form
  - Questions? Ask pop@bsc.es
- Selection/assignment process
- Install tools @ your production machine (local, PRACE, …)
- Interactively: Gather data → Analysis → Report



Overview of Codes Investigated

Status after 2½ Years (End of Phase1)

Performance Audits and Plans

- 139 completed or reporting to customer
  - 13 more in progress

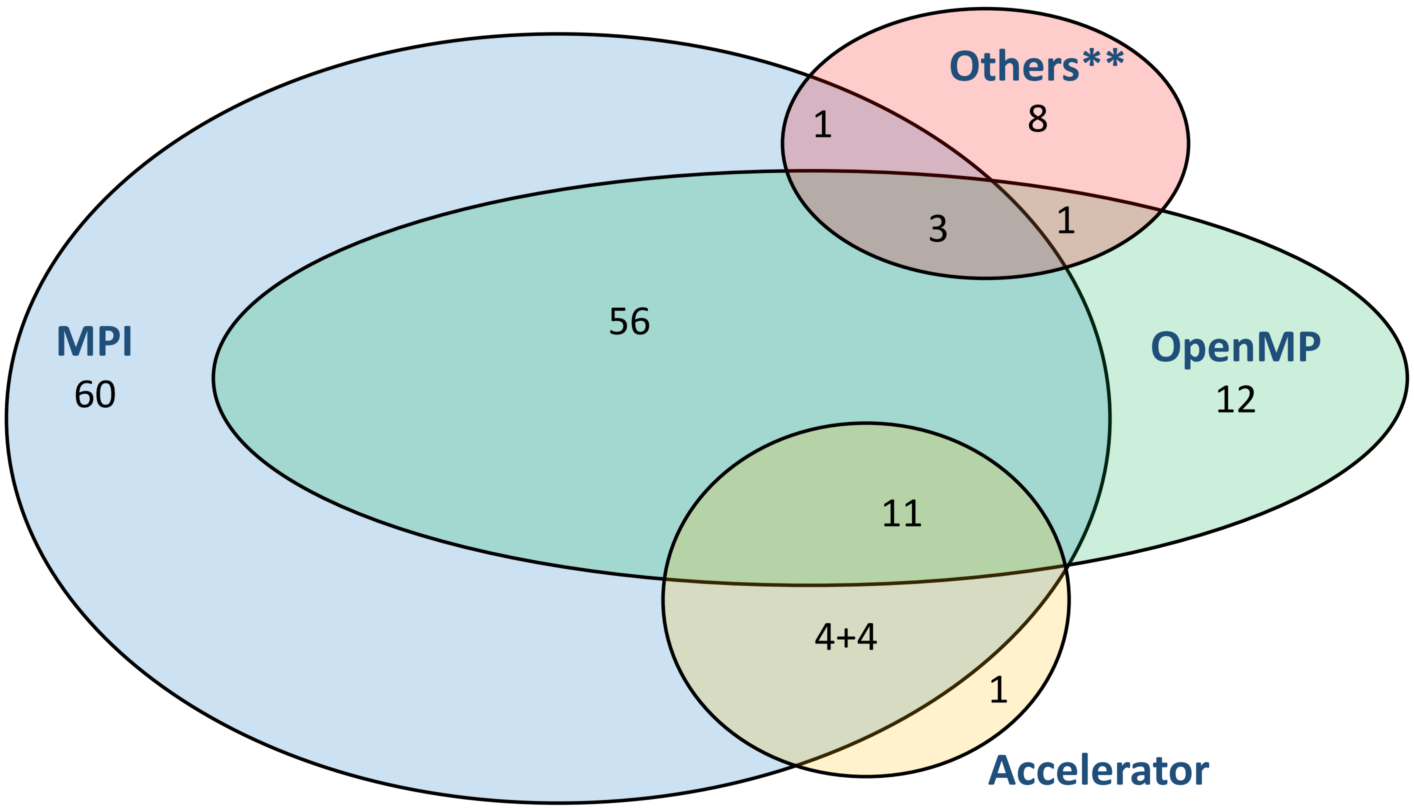
Proof-of-Concept

- 19 completed Proofs of Concept
  - 3 more in progress

Example POP Users and Their Codes

Area	Codes
Computational Fluid Dynamics	DROPS (RWTH Aachen), Nek5000 (PDC KTH), SOWFA (CENER), ParFlow (FZ-Juelich), FDS (COAC) & others
Electronic Structure Calculations	ADF, BAND, DFTB (SCM), Quantum Expresso (Cineca), FHI-AIMS (University of Barcelona), SIESTA (BSC), ONETEP (University of Warwick)
Earth Sciences	NEMO (BULL), UKCA (University of Cambridge), SHEMAT-Suite (RWTH Aachen), GITM (Cefas) & others
Finite Element Analysis	Ateles, Musubi (University of Siegen) & others
Gyrokinetic Plasma Turbulence	GYSELA (CEA), GS2 (STFC)
Materials Modelling	VAMPIRE (University of York), GraGLeS2D (RWTH Aachen), DPM (University of Luxembourg), QUIP (University of Warwick), FIDIMAG (University of Southampton), GBmolDD (University of Durham), k-Wave (Brno University), EPW (University of Oxford) & others
Neural Networks	OpenNN (Artelnics)

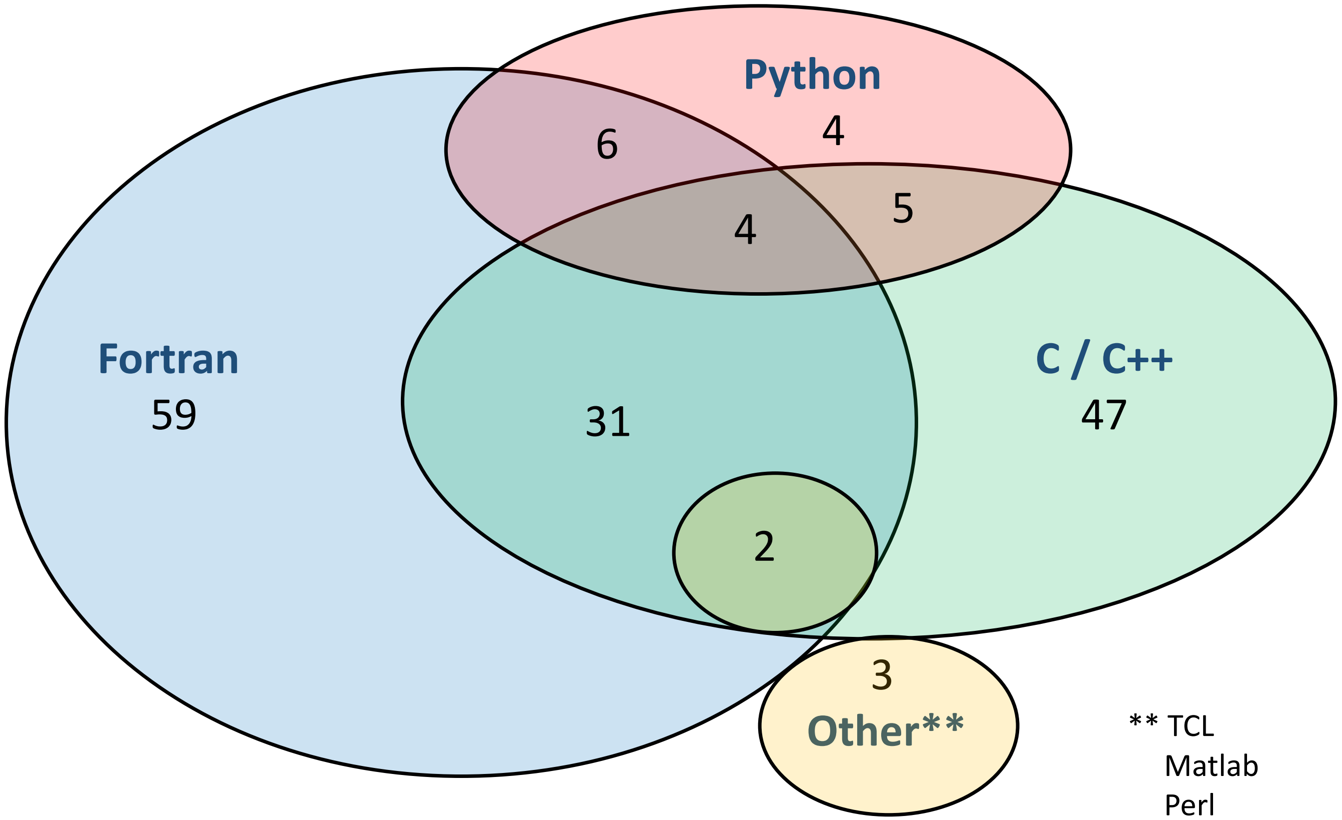
Programming Models Used



** MAGMA
Celery
TBB
GASPI
C++ threads
MATLAB PT
StarPU
GlobalArrays
Charm++
Fortran Coarray

* Based on data collected for 161 POP Performance Audits

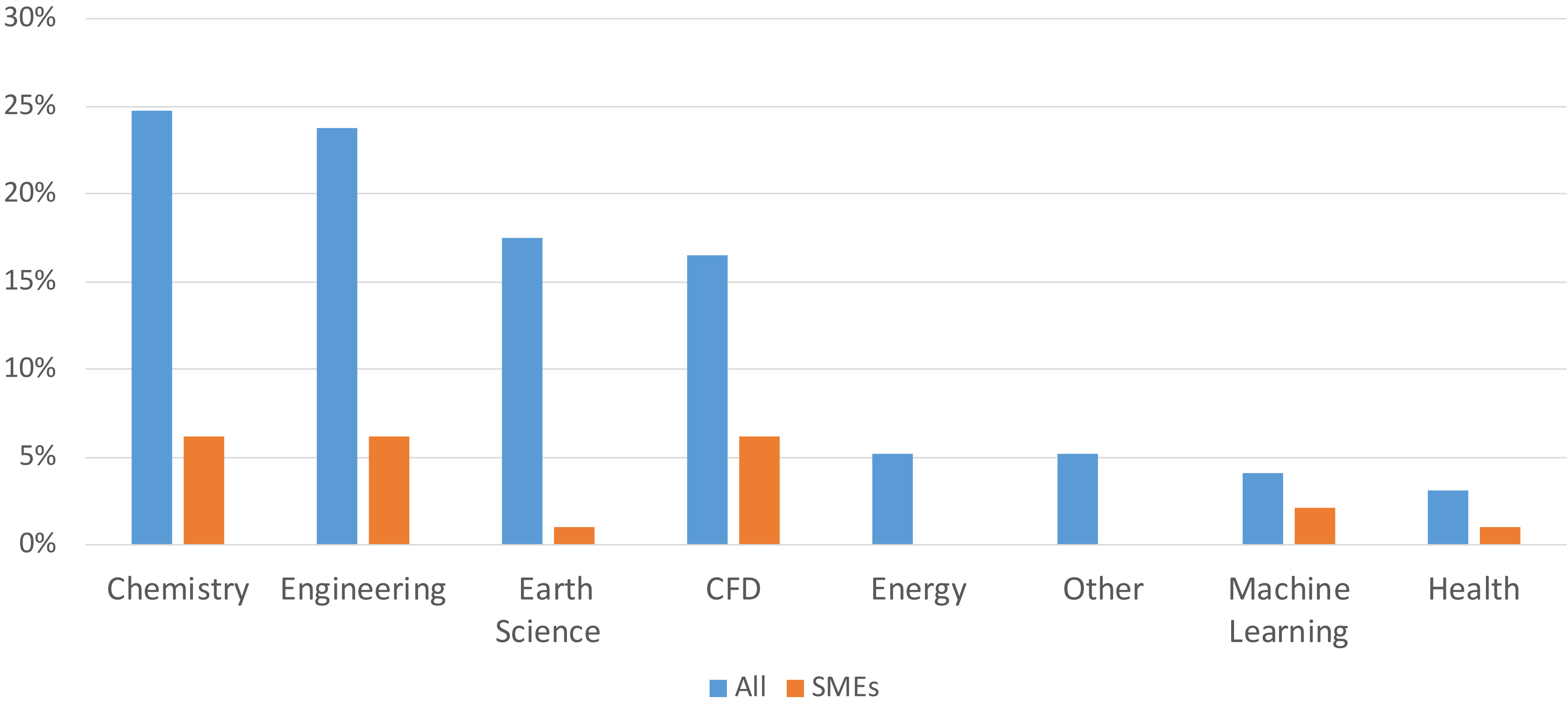
Programming Languages Used



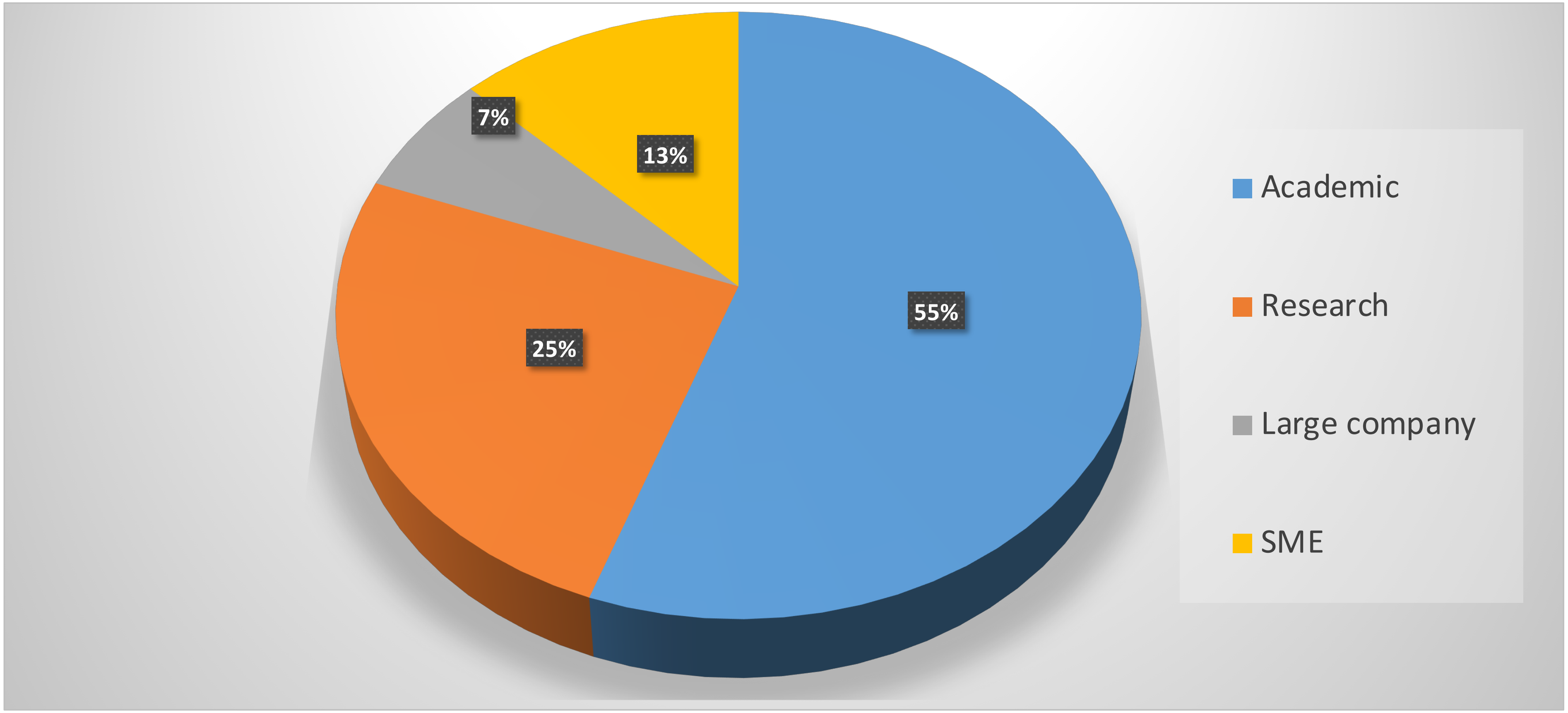
** TCL
Matlab
Perl
Octave
Java

* Based on data collected for 161 POP Performance Audits

Application Sectors

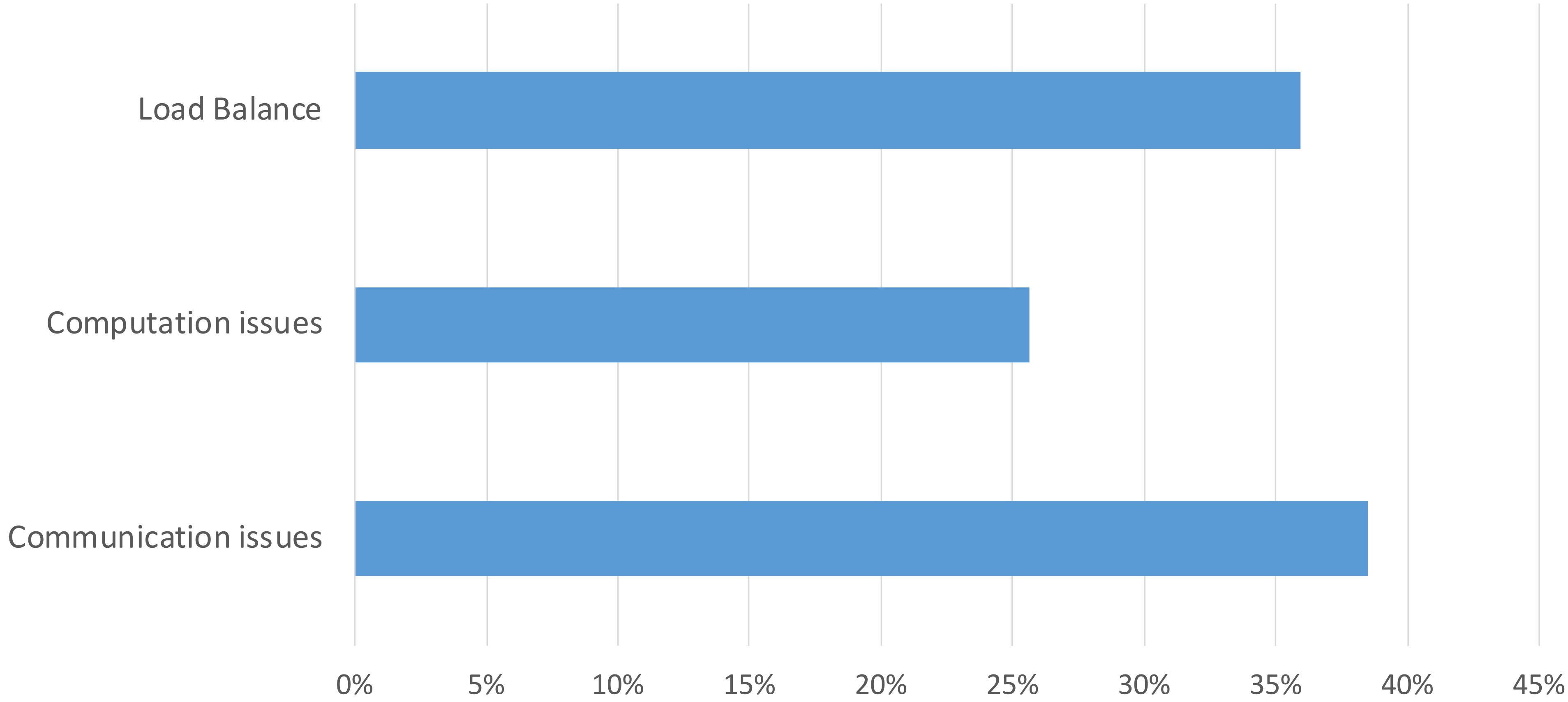


Customer Types



Analysis of Inefficiencies

Leading Cause of Inefficiency



Success Stories

Some PoC Success Stories

- See ⇨ https://pop-coe.eu/blog/tags/success-stories
  - Performance Improvements for SCM's ADF Modeling Suite
  - **3x Speed Improvement** for zCFD Computational Fluid Dynamics Solver
  - **25% Faster time-to-solution** for Urban Microclimate Simulations
  - **2x performance improvement** for SCM ADF code
  - Proof of Concept for BPMF leads to around **40% runtime reduction**
  - POP audit helps developers **double their code performance**
  - **10-fold scalability improvement** from POP services
  - POP performance study improves performance **up to a factor 6**
  - POP Proof-of-Concept study leads to **nearly 50% higher performance**
  - POP Proof-of-Concept study leads to **10X performance improvement** for customer

Improvements

Reductions

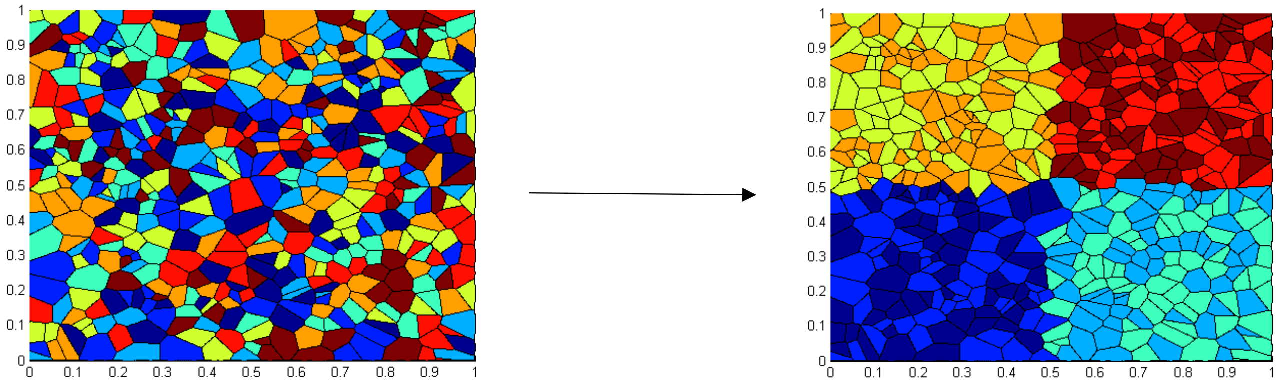
GraGLeS2D – RWTH Aachen

- Simulates grain growth phenomena in polycrystalline materials
- C++ parallelized with OpenMP
- Designed for very large SMP machines (e.g. 16 sockets and 2 TB memory)

- **Key audit results:**
  - **Good load balance**
  - **Costly use of division and square root inside loops**
  - **Not fully utilising vectorisation in key loops**
  - **NUMA data sharing issues lead to long times for memory access**

GraGLeS2D – RWTH Aachen

- Improvements:
  - Restructured code to enable vectorisation
  - Used memory allocation library optimised for NUMA machines
  - Reordered work distribution to optimise for data locality



- **Speed up in region of interest is more than 10x**
- **Overall application speed up is 2.5x**

Ateles – University of Siegen

- Finite element code
- C and Fortran code with hybrid MPI+OpenMP parallelisation
- **Key audit results:**
  - **High number of function calls**
  - **Costly divisions inside inner loops**
  - **Poor load balance**
- Performance plan:
  - Improve function inlining
  - Improve vectorisation
  - Reduce duplicate computation

Ateles – University of Siegen

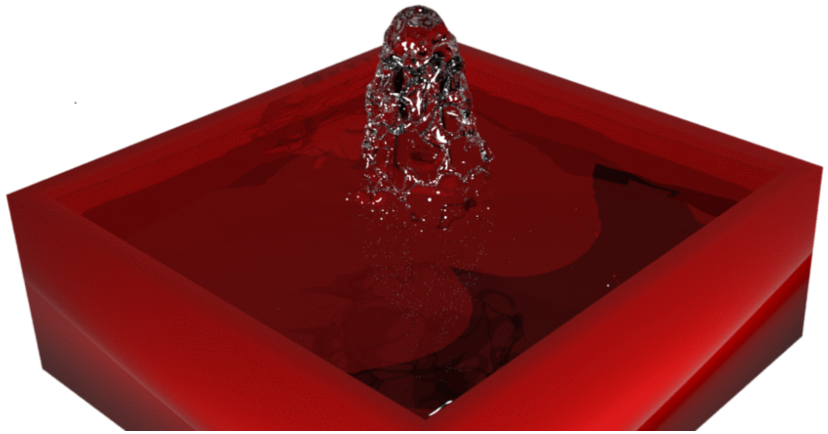
- Inlined key functions → **6% reduction in execution time**
- Improved mathematical operations in loops → **28% reduction in execution time**
- Vectorisation: found bug in gnu compiler, confirmed Intel compiler worked as expected

- **6 weeks software engineering effort**
- **Customer has confirmed “substantial” performance increase on production runs**

sphFluids – Stuttgart Media University

- Simulates fluids for computer graphics applications
- C++ parallelised with OpenMP

- Key audit results:
  - Several issues relating to the sequential computational performance
  - Located critical parts of the application with specific recommended improvements



sphFluids – Stuttgart Media University

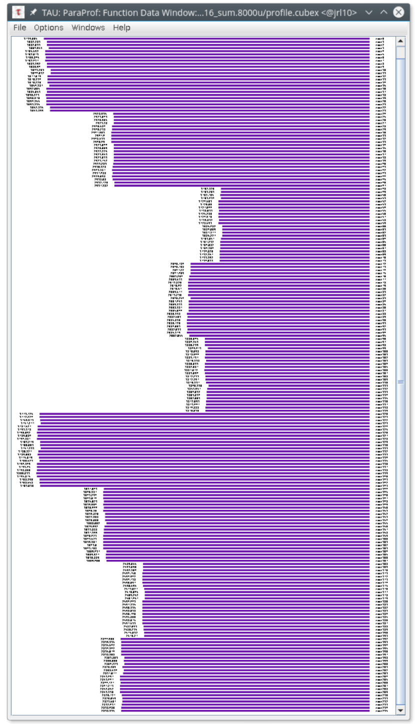
- Implemented by the code developers:
  - Review of overall code design from issues identified in POP audit
  - Inlining short functions
  - Reordering the particle processing order to reduce cache misses
  - Removal of unnecessary operations and costly inner loop definitions

- **Confirmed performance improvement up to 5x – 6x depending on scenario and pressure model used**
- Used insights provided by the POP experts and the good information exchange during the work

EPW – University of Oxford

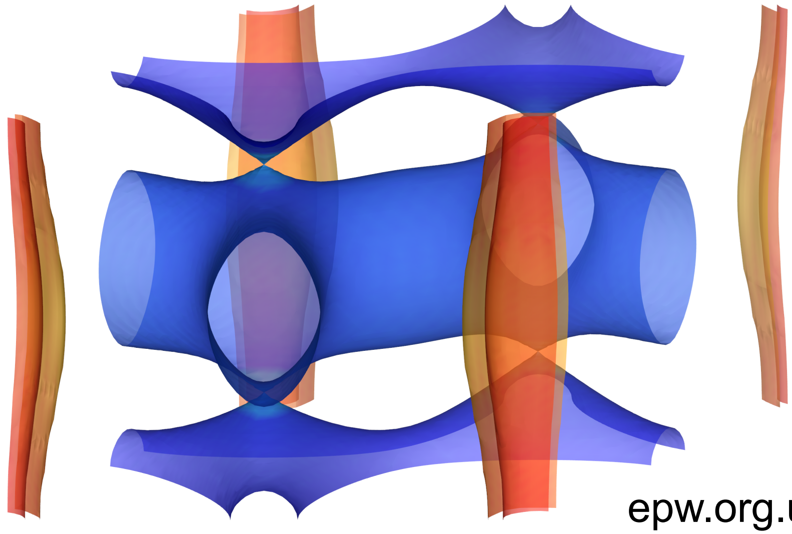
- Electron-Phonon Wannier (EPW) materials science DFT code;
- part of the Quantum ESPRESSO suite
- Fortran code parallelised with MPI
- Audit of unreleased development version of code
- Executed on ARCHER Cray XC30 (24 MPI ranks per node)

- Key audit findings:
  - Poor load balance from excessive computation identified
  - (addressed in separate POP Performance Plan)
  - Large variations in runtime, likely caused by IO
  - Final stage spends a great deal of time writing output to disk
- Report used for successful PRACE resource allocation



EPW – University of Oxford

- Original code had all MPI ranks writing the result to disk at the end
- POP PoC modified this to have only one rank do output
- **On 480 MPI ranks, time taken to write results fell from over 7 hours to 56 seconds: 450-fold speed-up!**
- **Combined with previous improvements, enabled EPW simulations to scale to previously impractical 1920 MPI ranks**
- **86% global efficiency with 960 MPI ranks**



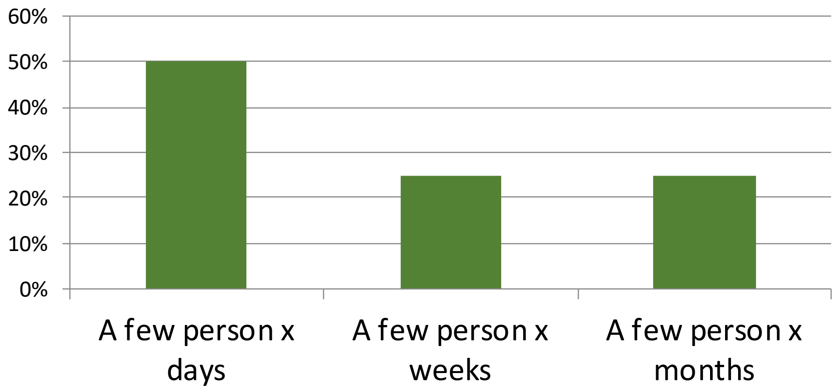
epw.org.uk

(Eight) Customers Success Feedback

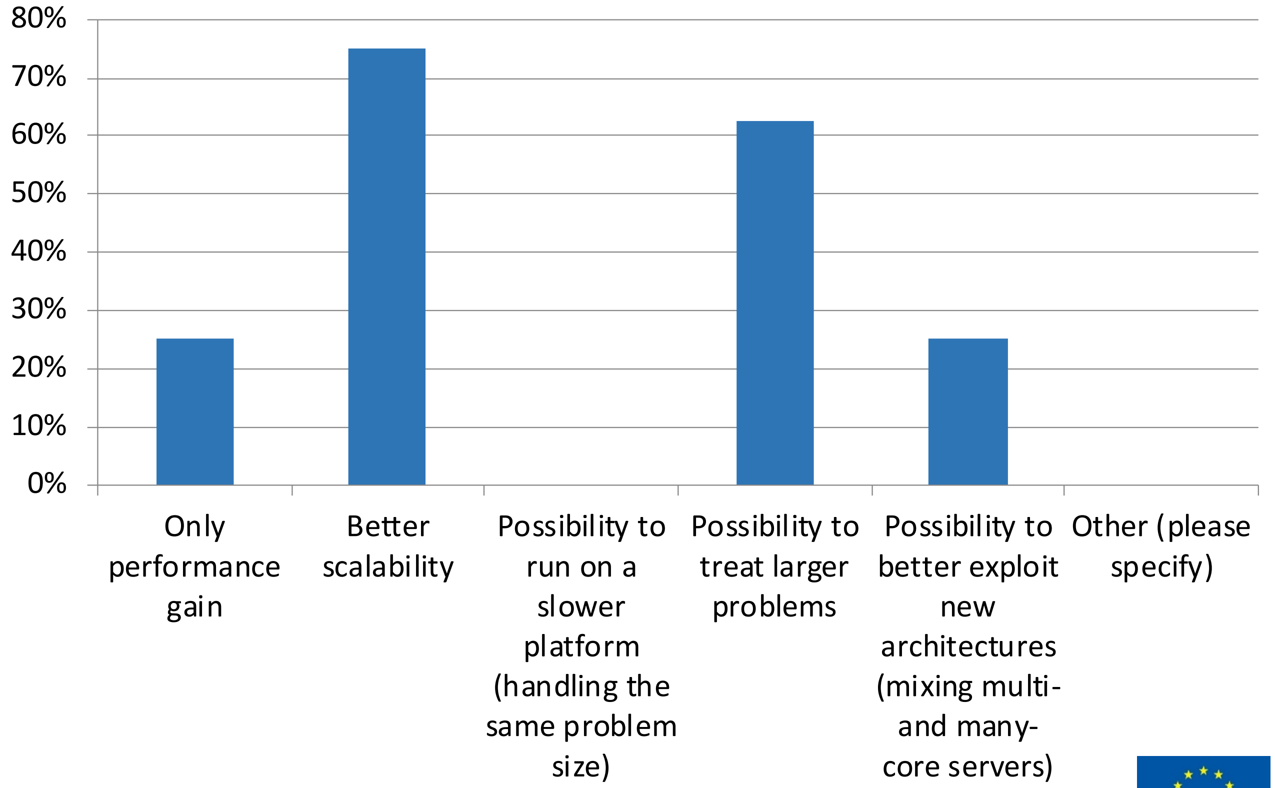
What is the observed performance gain after implementing recommendations?

25%
25%
20% overall, 50% for the given module
50-75% (case dependent)
12%
Up to 62 %, depending on the use case.
6 - 47 % depending on the test case.
15%

How much effort was necessary?



What are the main results?



Summary & Conclusion

Costumer Feedback

Performance Audits (73 customers)	• About 90% very satisfied or satisfied with service • About half of the customers signed-up for a follow-up service
Performance Plans (11 customers)	• About 90% very satisfied or satisfied with service • All customers thought suggestions were precise and clear and 70% plan to implement the suggested code modifications • About 2/3 plan to do use the POP services again
Proof-of-Concepts (8 customers)	• All customers very satisfied or satisfied with this service • About 80% plan to implement further code modifications or complete the work of the POP experts

* Based on data collected in 92 customer satisfaction questionnaires and 52 phone interviews with customers

ROI Examples

Application Savings after POP Proof-of-Concept

- POP PoC resulted in 72% faster-time-to-solution
- Production runs on ARCHER (UK national academic supercomputer)
- Improved code saves €15.58 per run
- Yearly savings of around €56,000 (from monthly usage data)

Application Savings after POP Performance Plan

- Cost for customer implementing POP recommendations: €2,000
- Achieved improvement of 62%
- €20,000 yearly operating cost
- Resulted in yearly saving of €12,400 in compute costs ⇨ ROI of 620%

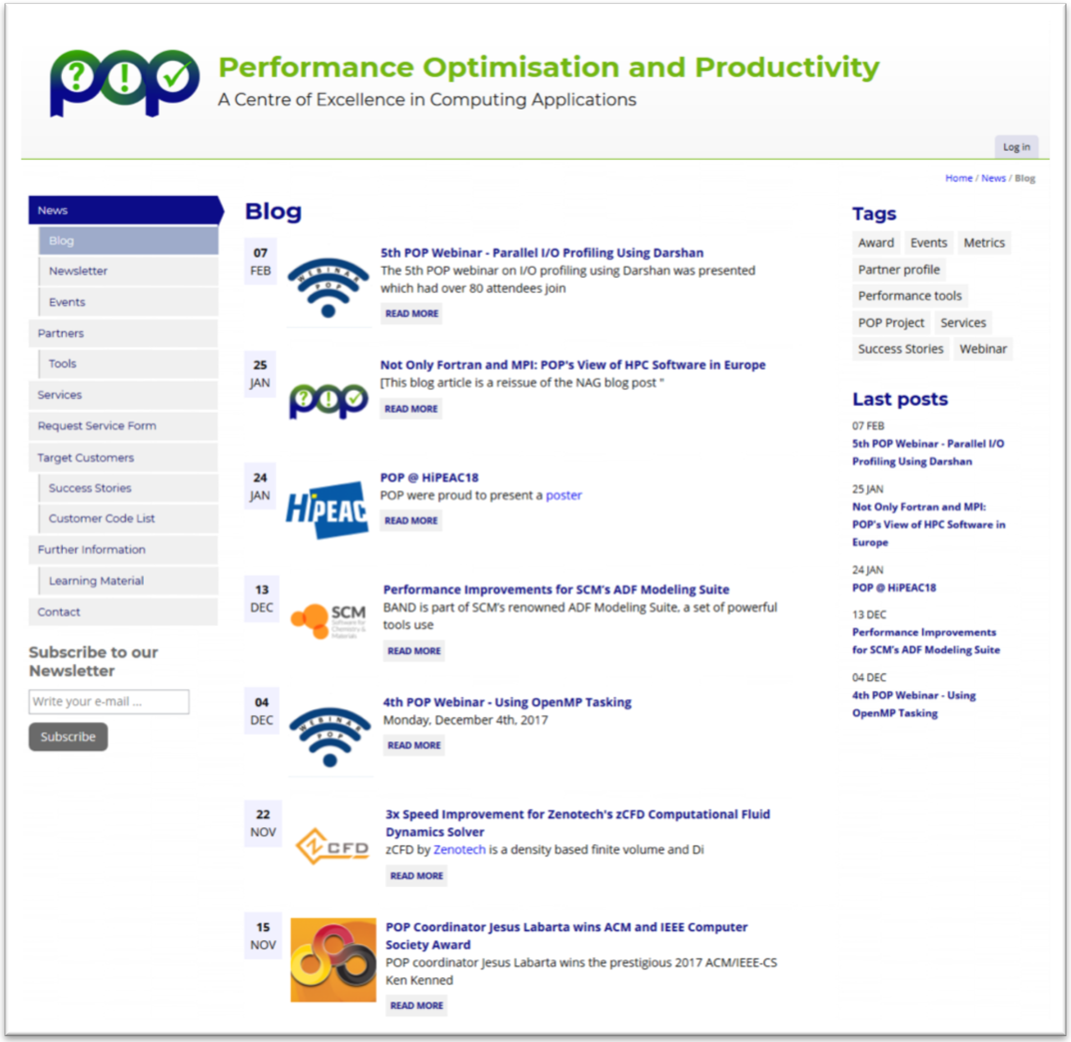
Dissemination and Contact

Website – www.pop-coe.eu

- POP User Portal
- Access to all public information and servcies

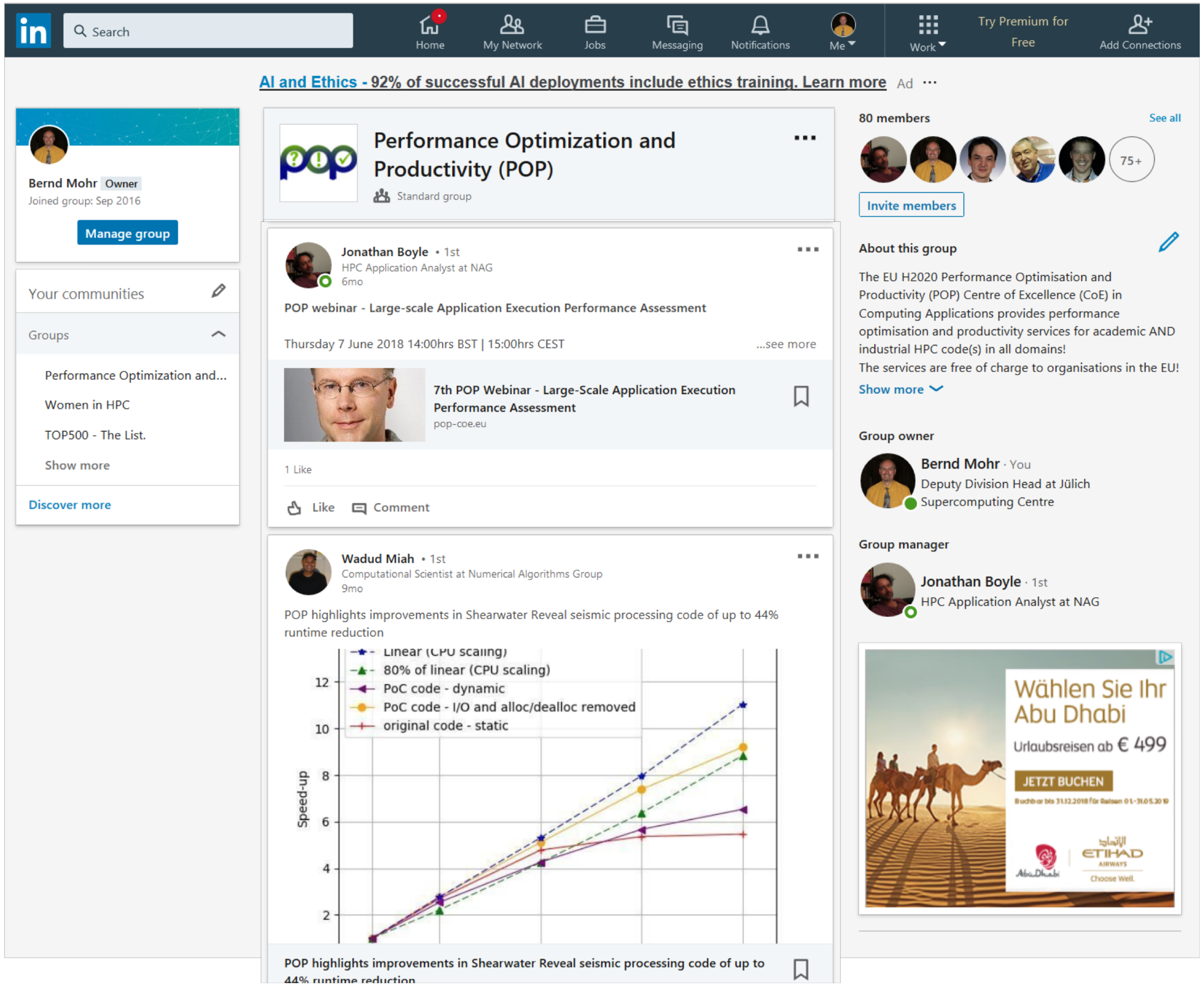
Blog – https://pop-coe.eu/blog

- Typically 2 new articles per month
- Easy filtering via Tags, e.g
  - Success Stories
  - Events
  - Webinars
  - …



Follow us on Twitter @POP_HPC

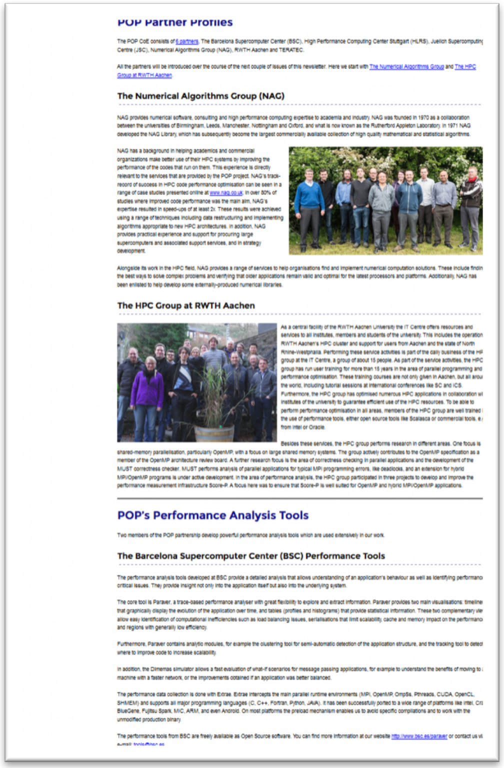
LinkedIn Group

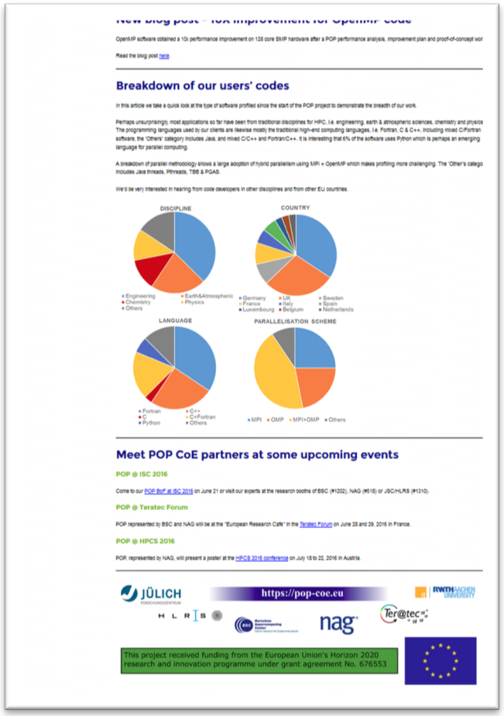


- Important announcements
- Serves also as user forum

Quarterly Email Newsletter

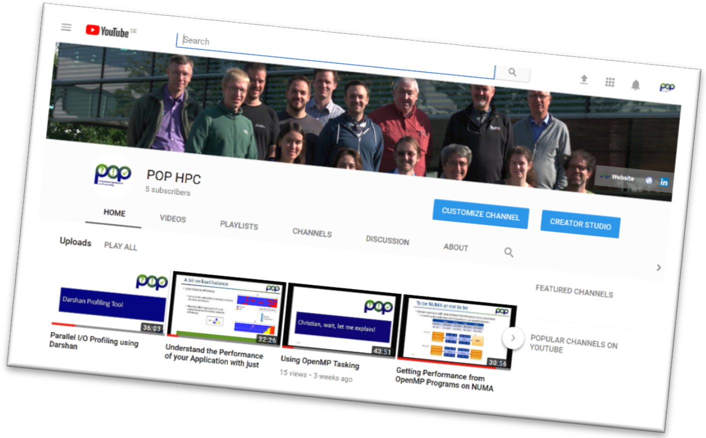
- Subscribe on POP website
- Newsletter archive: https://pop-coe.eu/news/newsletter





Webinars / YouTube

- See ⇨ https://pop-coe.eu/blog/tags/webinar
- Or see our YouTube Channel



- Already available:
  - How to Improve the Performance of Parallel Codes
  - Getting Performance from OpenMP Programs on NUMA Architectures
  - Understand the Performance of your Application with just Three Numbers
  - Using OpenMP Tasking
  - Parallel I/O Profiling Using Darshan
  - The impact of sequential performance on parallel codes
  - Large scale Application Execution Performance Assessment

Performance Optimisation and Productivity

A Centre of Excellence in HPC

Contact:

https://www.pop-coe.eu

mailto:pop@bsc.es

@POP_HPC

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676553 and 824080.