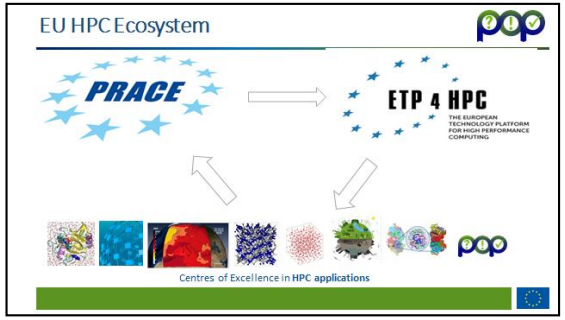
Performance Optimisation and Productivity

EU H2020 Centre of Excellence (CoE)

1 October 2015 – 31 March 2018

Grant Agreement No 676553

POP CoE



- A **Centre of Excellence**
  - On **Performance Optimisation and Productivity**
  - Promoting **best practices in parallel programming**
- Providing **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 (your?) academic AND industrial codes and users !**

Partners

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

Motivation

Why?

- Complexity of machines and codes
  → Frequent lack of quantified understanding of actual behaviour
  → Not clear most productive direction of code refactoring
- Important to maximize efficiency (performance, power) of compute intensive applications and productivity of the development efforts

What?

- Parallel programs, mainly MPI/OpenMP
  - Although also CUDA, OpenCL, OpenACC, Python, …

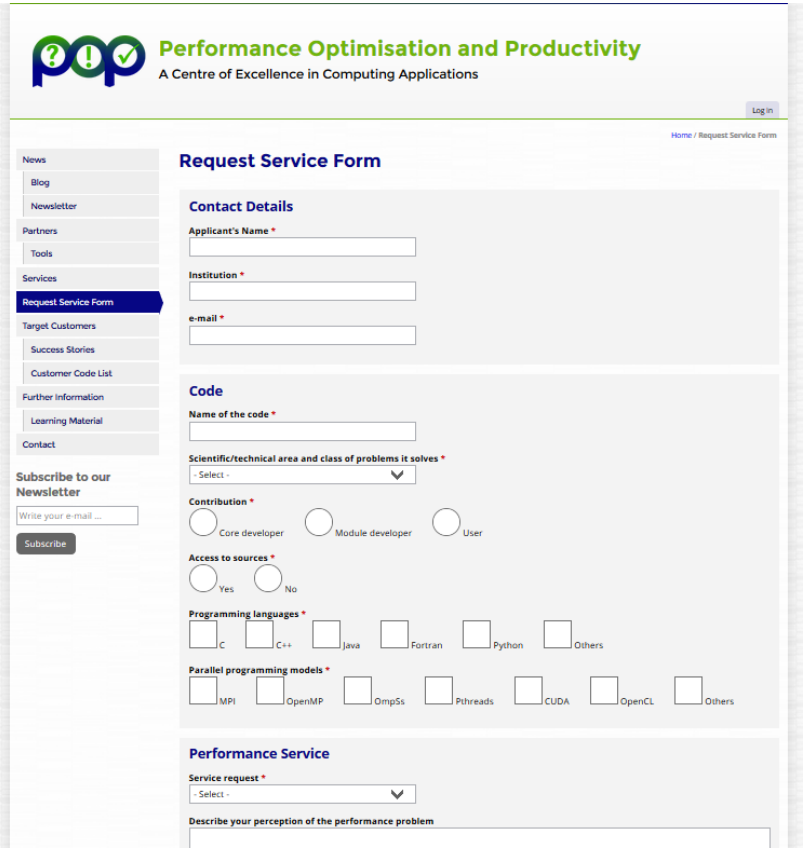
The process …

When?

October 2015 – March 2018

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



Services provided by the CoE

? Parallel Application Performance Audit ⇨ Report

- Primary service
- Identify performance issues of customer code (at customer site)
- Small effort (< 1 month)

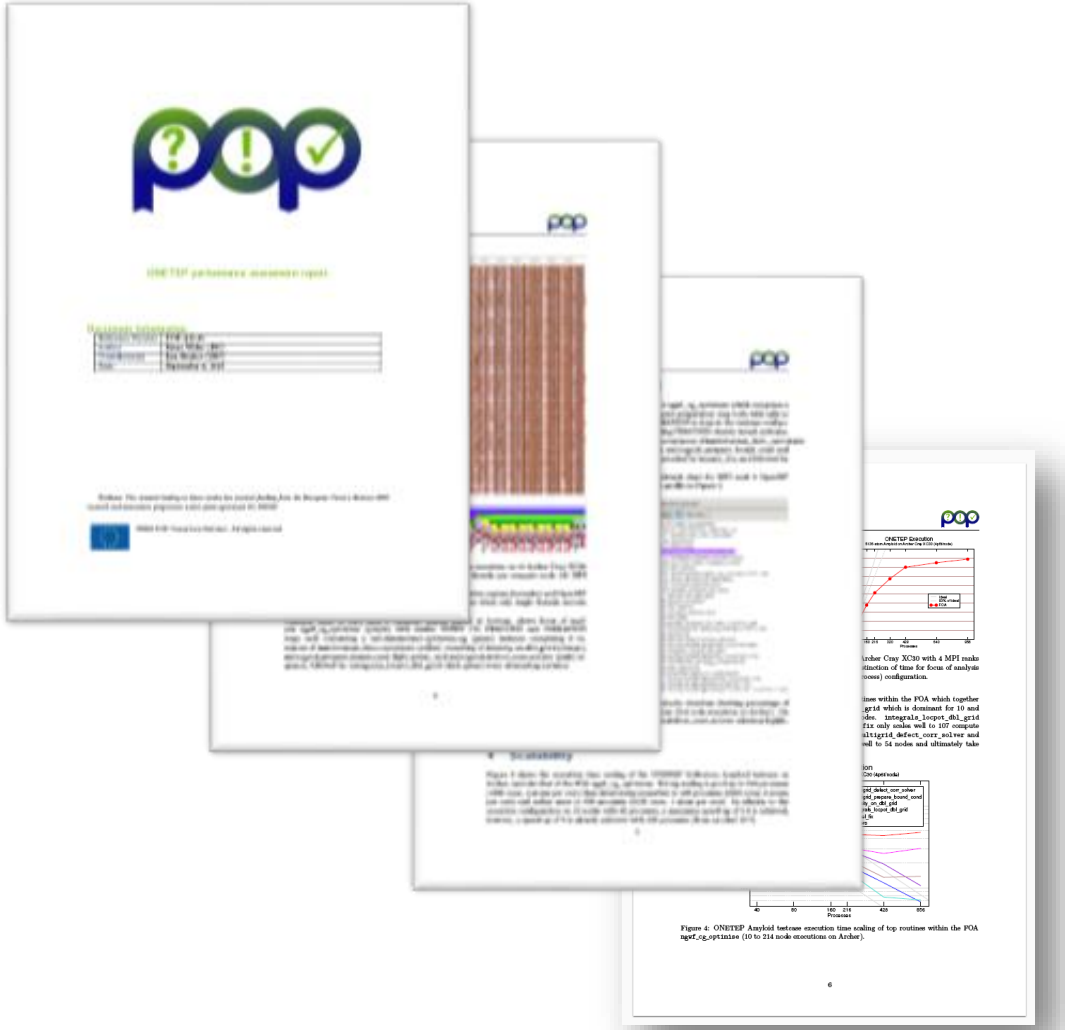
! Parallel Application Performance Plan ⇨ Report

- Follow-up on the audit service
- Identifies the root causes of the issues found and qualifies and quantifies approaches to address them
- Longer effort (1-3 months)

✓ Proof-of-Concept ⇨ Software Demonstrator

- Experiments and mock-up tests for customer codes
- Kernel extraction, parallelisation, mini-apps experiments to show effect of proposed optimisations
- 6 months effort

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

Efficiencies

- The following metrics are used in a POP Performance Audit:

CT = Computational time
TT = Total time

- 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
  - Computation Efficiency (CompE)
    - Computed out of IPC Scaling and Instruction Scaling
    - For strong scaling: ideal scaling -> efficiency of 1.0
- Details see https://sharepoint.ecampus.rwth-aachen.de/units/rz/HPC/public/Shared%20Documents/Metrics.pdf

Target customers

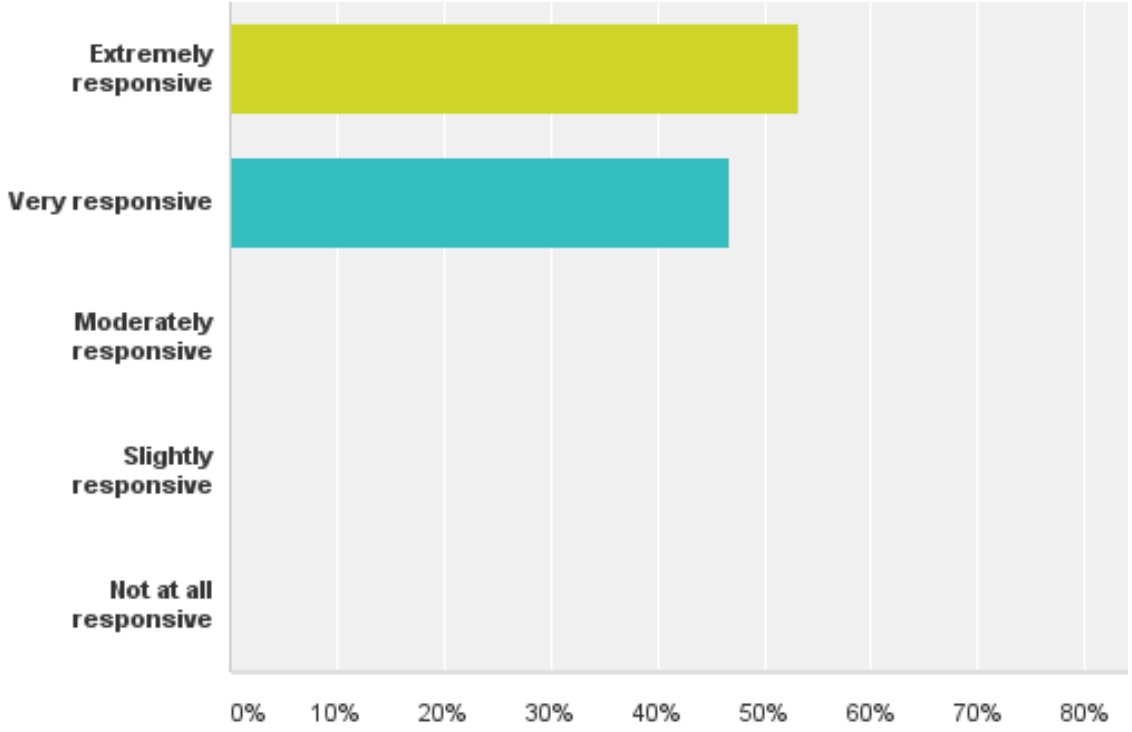
- **Code developers**
  - Assessment of detailed actual behaviour
  - Suggestion of most productive directions to refactor code
- **Users**
  - Assessment of achieved performance in specific production conditions
  - Possible improvements modifying environment setup
  - Evidence to interact with code 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

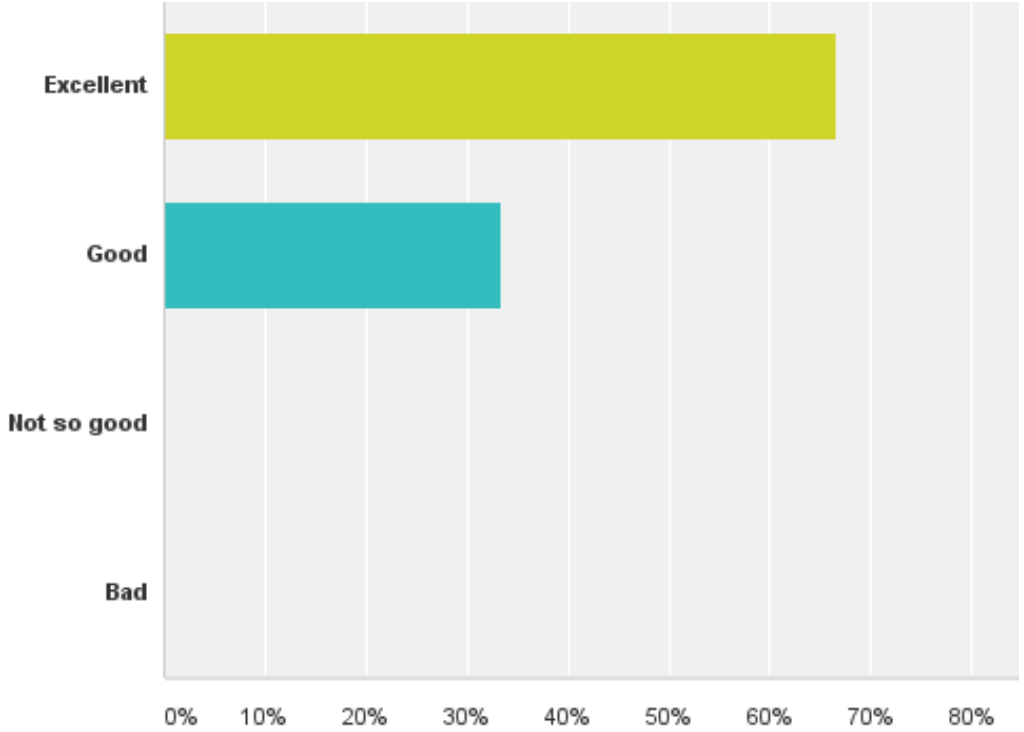
POP Users and Their Codes

Area	Codes
Computational Fluid Dynamics	DROPS (RWTH Aachen), Nek5000 (PDC KTH), SOWFA (CENER), ParFlow (FZ-Juelich), FDS (COAC) & others
Electronic Structure Calculations	ADF (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) & others
Finite Element Analysis	Ateles (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) & others
Neural Networks	OpenNN (Artelnics)

Costumer Feedback (Sep 2016)

- Results from 18 of 23 completed feedback surveys (~78%)





- How responsive have the POP experts been to your questions or concerns about the analysis and the report?
- What was the quality of their answers?

Best Practices in Performance Analysis

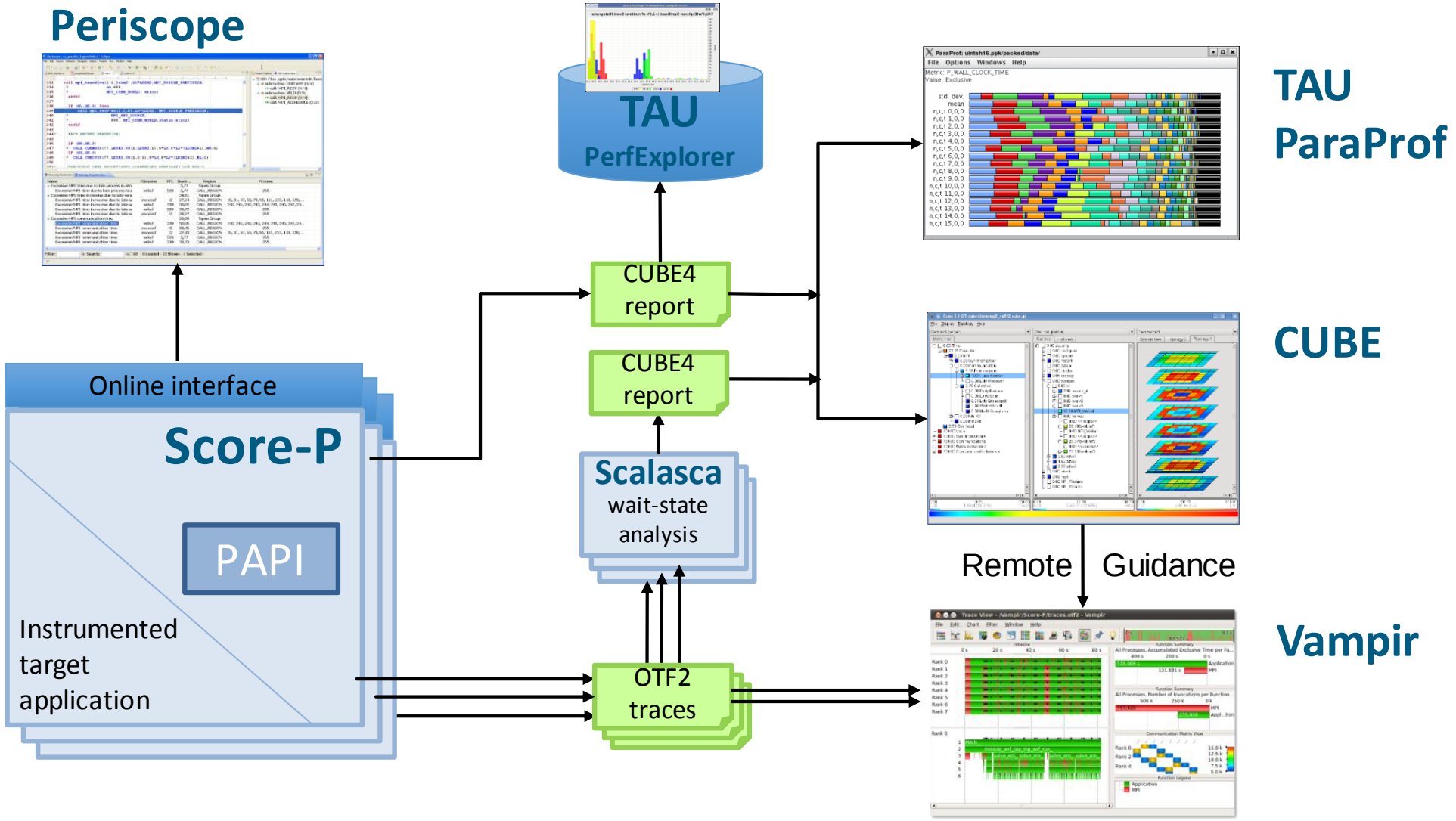
- **Powerful tools …**
  - Extrae + Paraver
  - Score-P + Scalasca/TAU/Vampir + Cube
  - Dimemas, Extra-P
  - Other commercial tools
- **… and techniques**
  - Clustering, modeling, projection, extrapolation, memory access patterns,
  - … with extreme detail …
  - … and up to extreme scale
- **Unify methodologies**
  - Structure
    - Spatio temporal / syntactic
  - Metrics
    - Parallel fundamental factors: Efficiency, Load balance, Serialization
    - Programming model related metrics
    - User level code sequential performance
  - Hierarchical search
    - From high level fundamental behavior to its causes
- **To deliver insight**
- **To estimate potentials**

Performance Tools

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
  - SimGrid
- **Commercial toolsets**
  (if available at customer site)
  - Intel tools
  - Cray tools
  - Allinea tools

Score-P Tool Ecosystem -- Overview



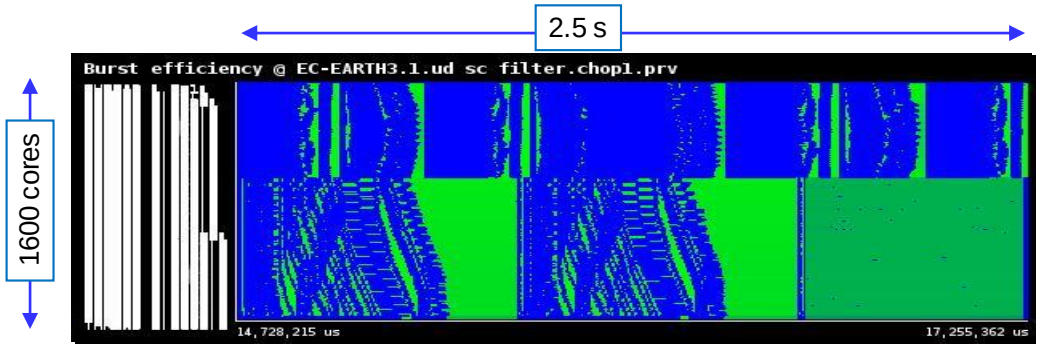
Score-P Tool Ecosystem -- Status

- **Score-P** (www.score-p.org)
  - Parallel Program Instrumentation and Profile/Trace Measurement
    - MPI, OpenMP, SHMEM, CUDA, OpenCL, OmpSs support
  - Latest version: 3.0
    - New: User function sampling + MPI measurement, OpenACC support
- **Scalasca** (www.scalasca.org)
  - Scalable Profile and Trace analysis
  - Latest version: 2.3.1
    - New: More platforms (Xeon Phi, K computer, ARM64, …), Score-P 2.X and 3.x support
- **Cube** (www.scalasca.org)
  - Profile browser
  - Latest version: 4.3.4
    - Soon: Client/server architecture, more analysis plugins, performance improvements

BSC Performance Tools (www.bsc/es/paraver)

Flexible trace visualization and analysis

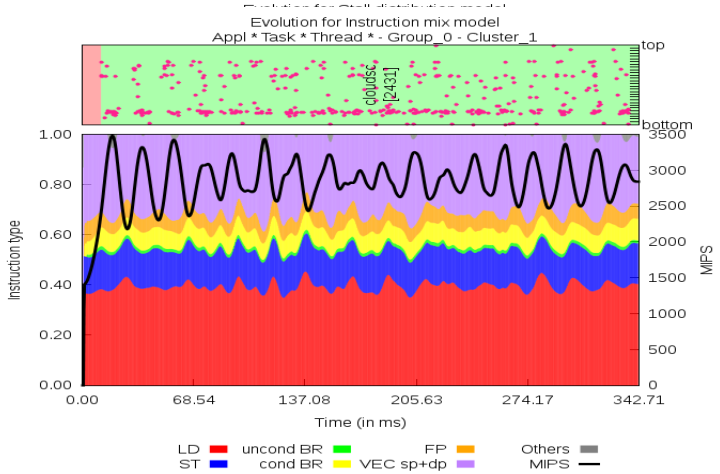
Adaptive burst mode tracing



26.7MB trace
Eff: 0.43; LB: 0.52; Comm:0.81

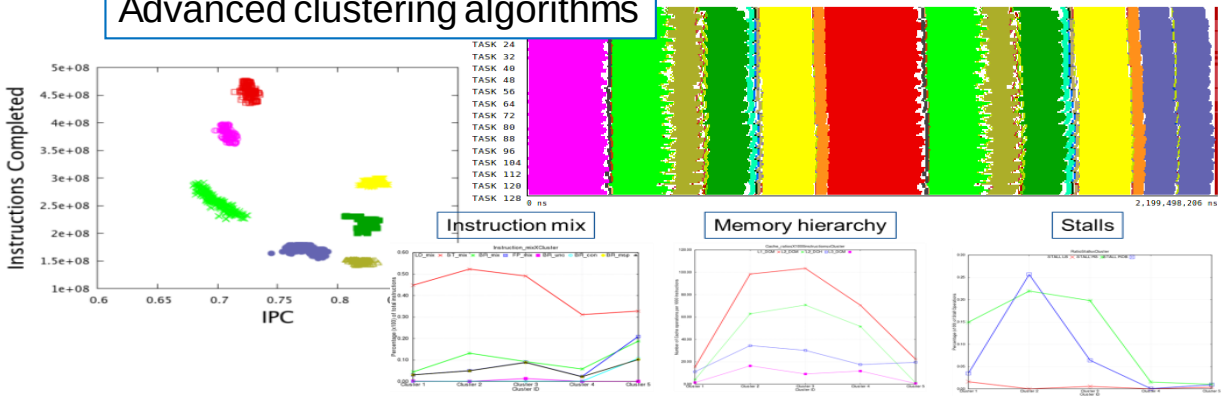
BSC-ES – EC-EARTH

Instantaneous metrics for ALL hardware counters at "no" cost

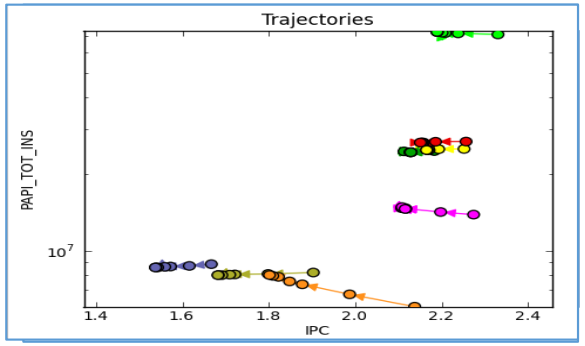


BSC-ES – EC-EARTH

Advanced clustering algorithms

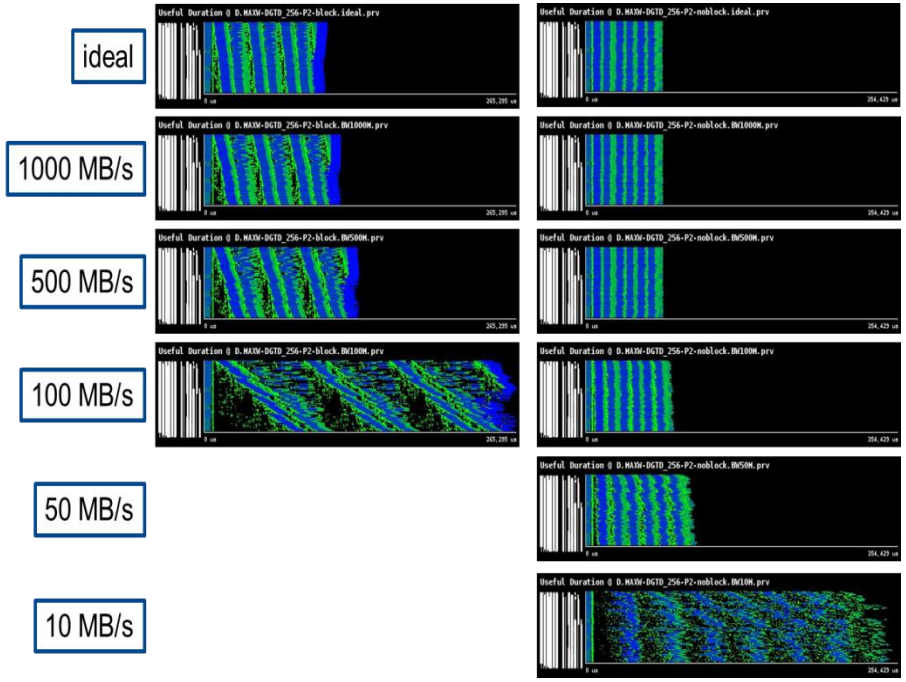


Tracking performance evolution

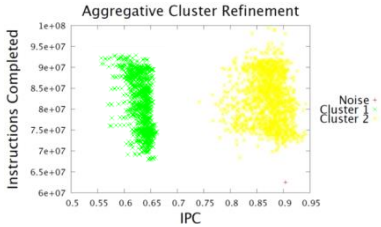


AMG2013

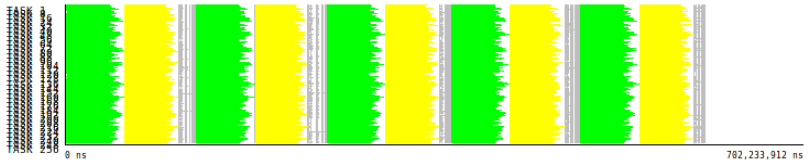
BSC Performance Tools (www.bsc/es/paraver)

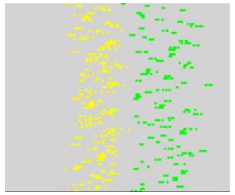


What if …

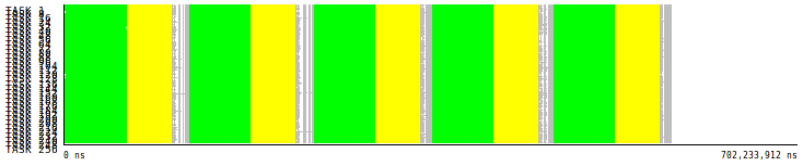


… we increase the IPC of Cluster1?



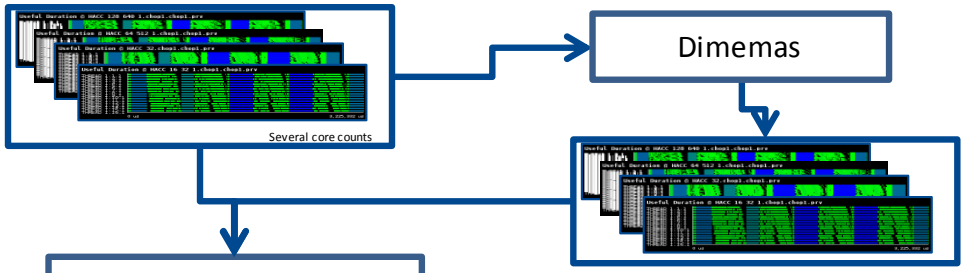


… we balance Clusters 1 & 2?

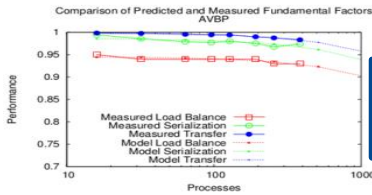


BSC Performance Tools (www.bsc/es/paraver)

Models and Projection



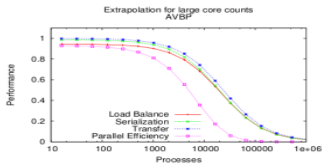
eff_factors.py

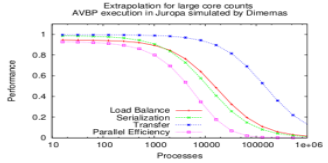


eff.csv

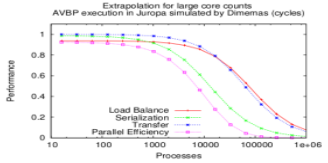
$$\eta_{\parallel} = LB * Ser * Trf$$

extrapolation.py





No MPI noise

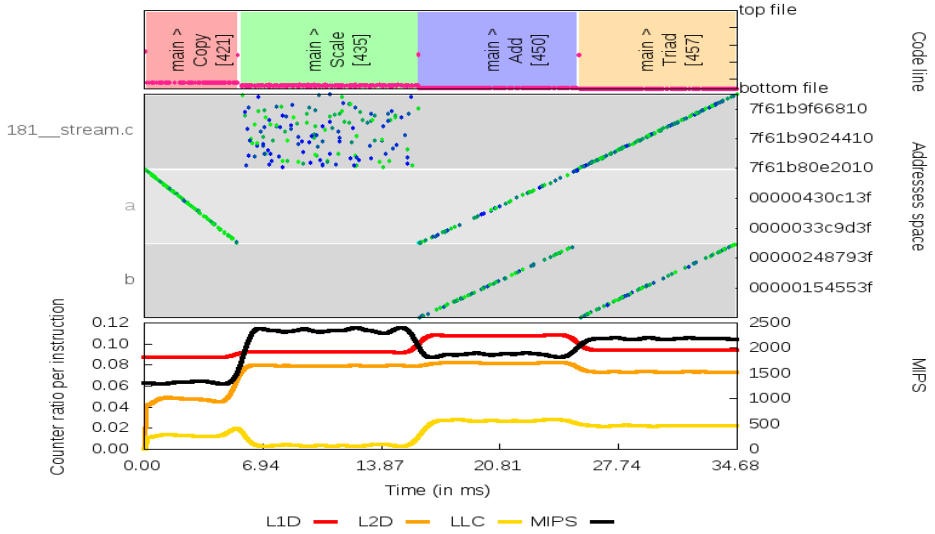


+ No OS noise

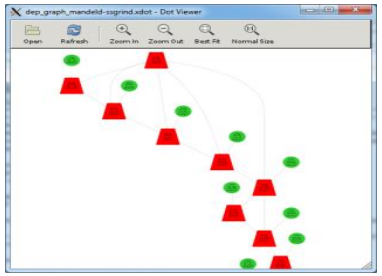
"Scalability prediction for fundamental performance factors" J. Labarta et al. SuperFRI 2014

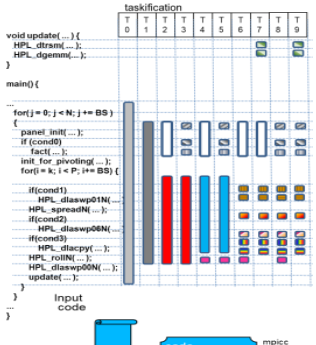
Intel–BSC Exascale Lab

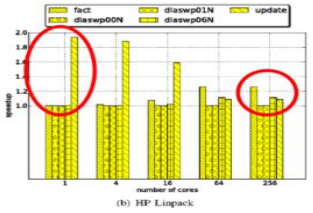
Data access patterns

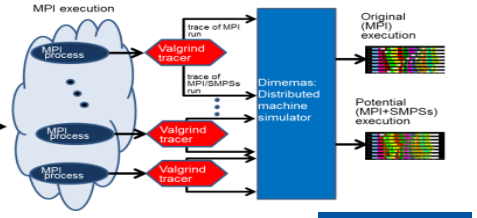


Tareador





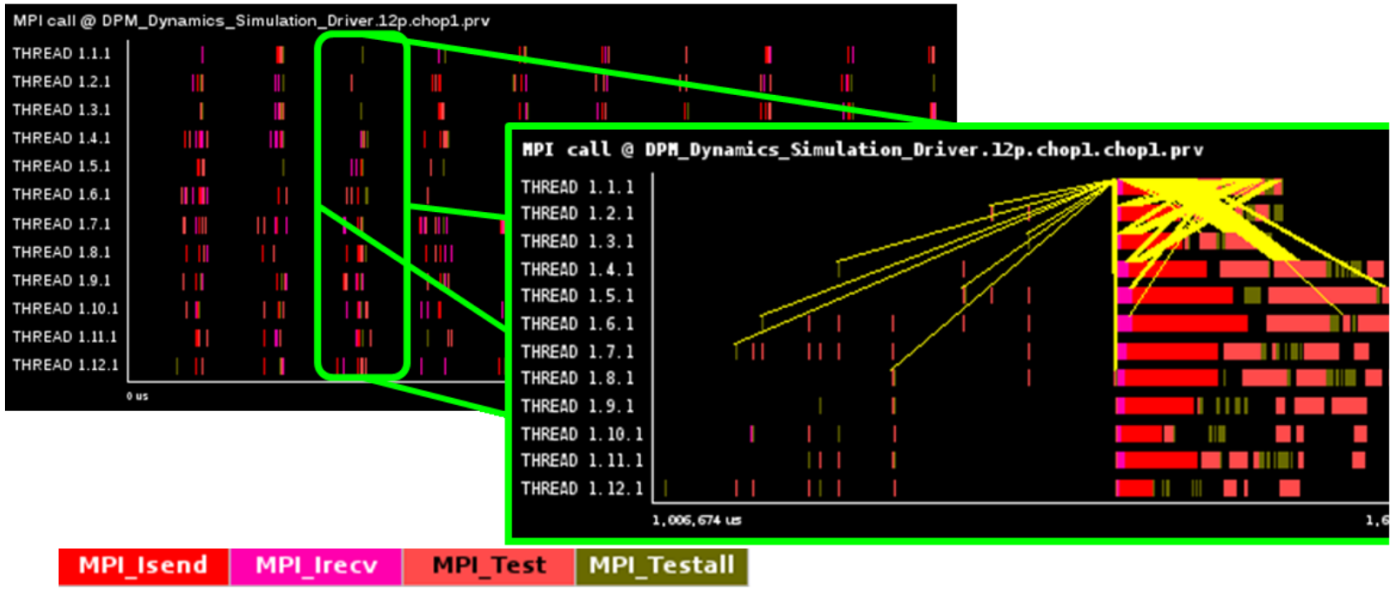




Code Audit Examples

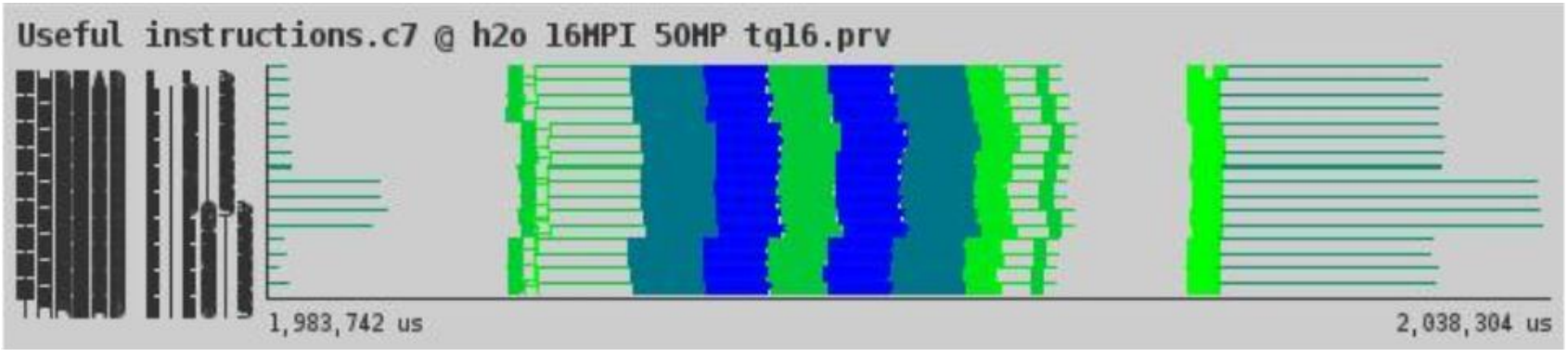
DPM – University of Luxembourg

- Numerical simulation tool for studying the motion and chemical conversion of particulate material in furnaces
- C++ code parallelised with MPI
- Key audit results:
  - Performance problems were due to the way that the code had been parallelised
  - Scalability limited by end-point contention due to sending MPI messages in increasing-rank order



Quantum Espresso – Cineca/MaX CoE

- An integrated suite of codes for nanoscale electronic structure calculations and materials modelling
- Very widely used
- Fortran code with hybrid MPI+OpenMP
- Key audit result:
  - For a significant portion of time only 1 out of 5 OpenMP threads per MPI process does useful computation (1.77x speedup over 1 thread)

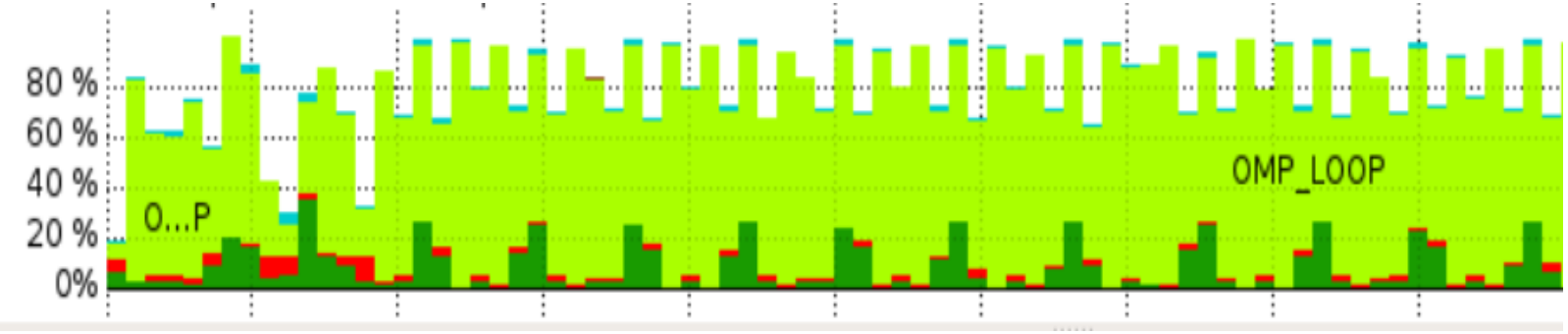


VAMPIRE – University of York

- Magnetic materials simulation code
- C++ code parallelised with MPI
- Key audit results:
  - Best enhancements would be to vectorise main loops, improve cache reuse and replace multiple calls to the random number generator with a single call that returns a vector of numbers
  - Initial implementation of these points by the user suggests that they could lead to 2x speedup

GYSELA – CEA

- 5D gyrokinetic code for studying flux-driven plasma turbulence in tokamaks
- Fortran code with hybrid MPI+OpenMP
- Key audit results:
  - Not fully utilising OpenMP threads: idle for 17.24% of execution time (only 1.4% due to MPI)
  - Imbalance due to unequal distribution of threads on nodes



Proof-of-Concept Examples

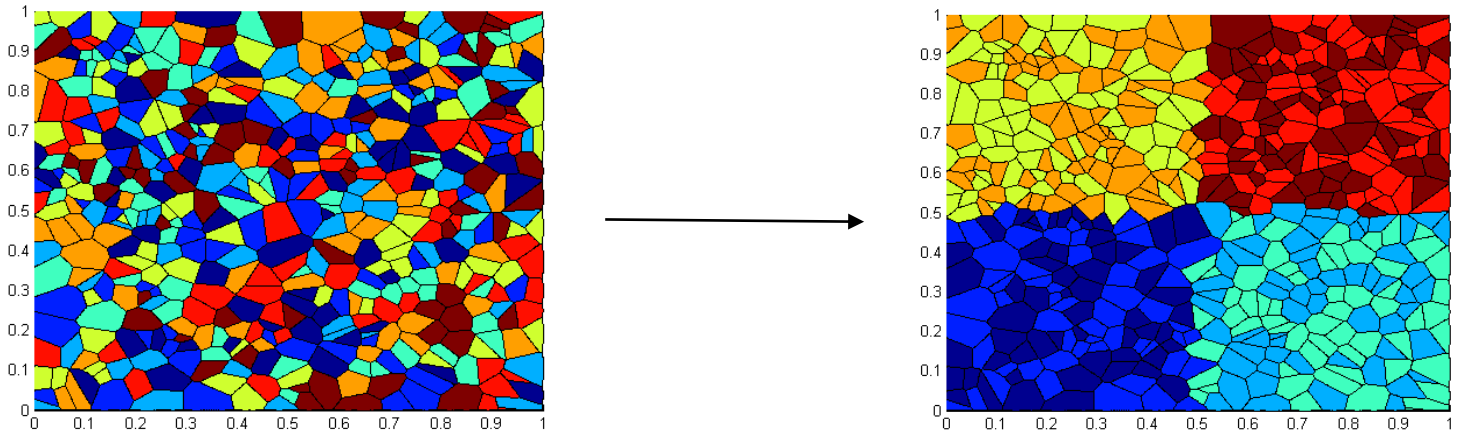
GraGLeS2D – RWTH Aachen

- 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 specific data sharing issues lead to long times for memory access

GraGLeS2D – RWTH Aachen

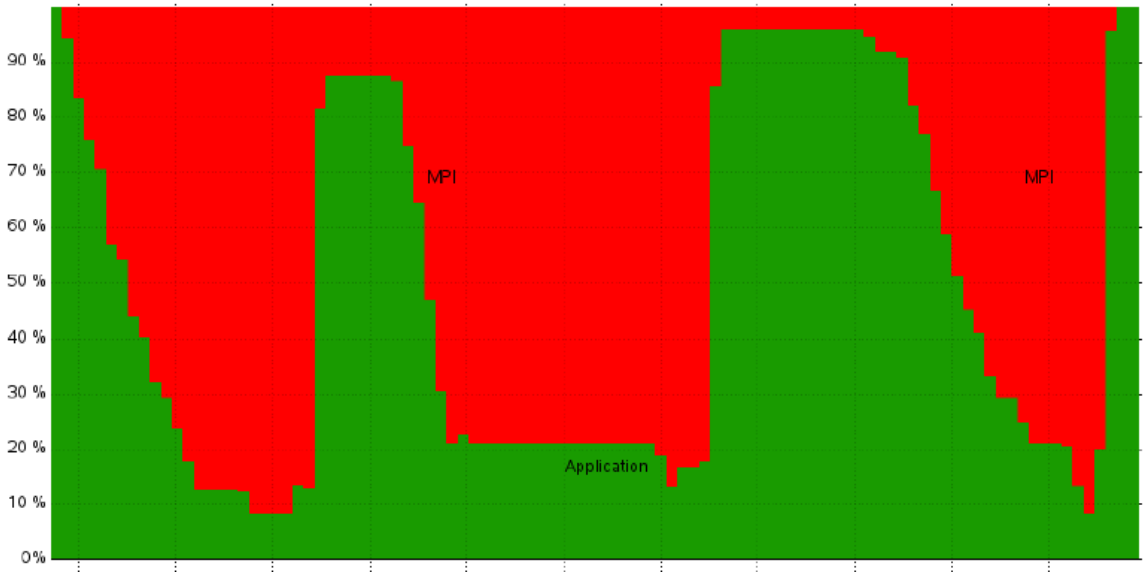
- Improvements:
  - Restructured code to enable vectorisation
  - Used memory allocation library optimised for NUMA machines
  - Reordered work distribution to optimise for data locality



- 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 – Proof-of-concept

- 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

Contact us !!

- If you have the feeling you are not getting the performance you expected
- If you are not sure whether it is a problem of your application, the system, …
- If you want an external view and recommendations on suggested refactoring efforts
- If you would like some help on how to best restructure your code

POP Coordination

Prof. Jesus Labarta, Judit Gimenez

Barcelona Supercomputing Center (BSC)

Email: pop@bsc.es

URL: http://www.pop-coe.eu

Other activities

- **Customer advocacy**
  - Gather customers feedback, ensure satisfaction, steer activities
- **Sustainability**
  - Explore business models
- **Training**
  - Best practices on the use of the tools and programming models (MPI + OpenMP)

Performance Optimisation and Productivity

A Centre of Excellence in Computing Applications

Contact:

https://www.pop-coe.eu

mailto:pop@bsc.es

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