
# POP: The Story So Far

Mike Dewar, NAG Ltd

EU H2020 Center of Excellence (CoE)

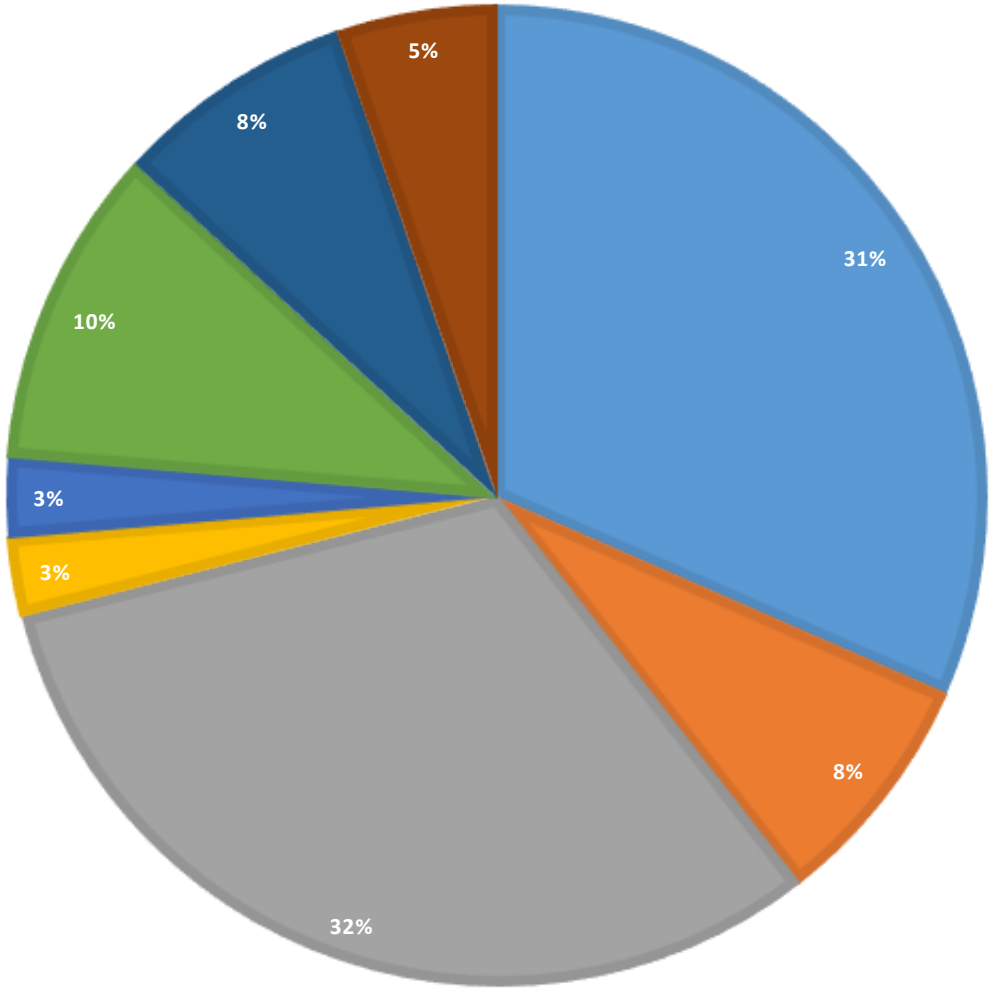
1 October 2015 – 31 March 2018

Grant Agreement No 676553

# Outline

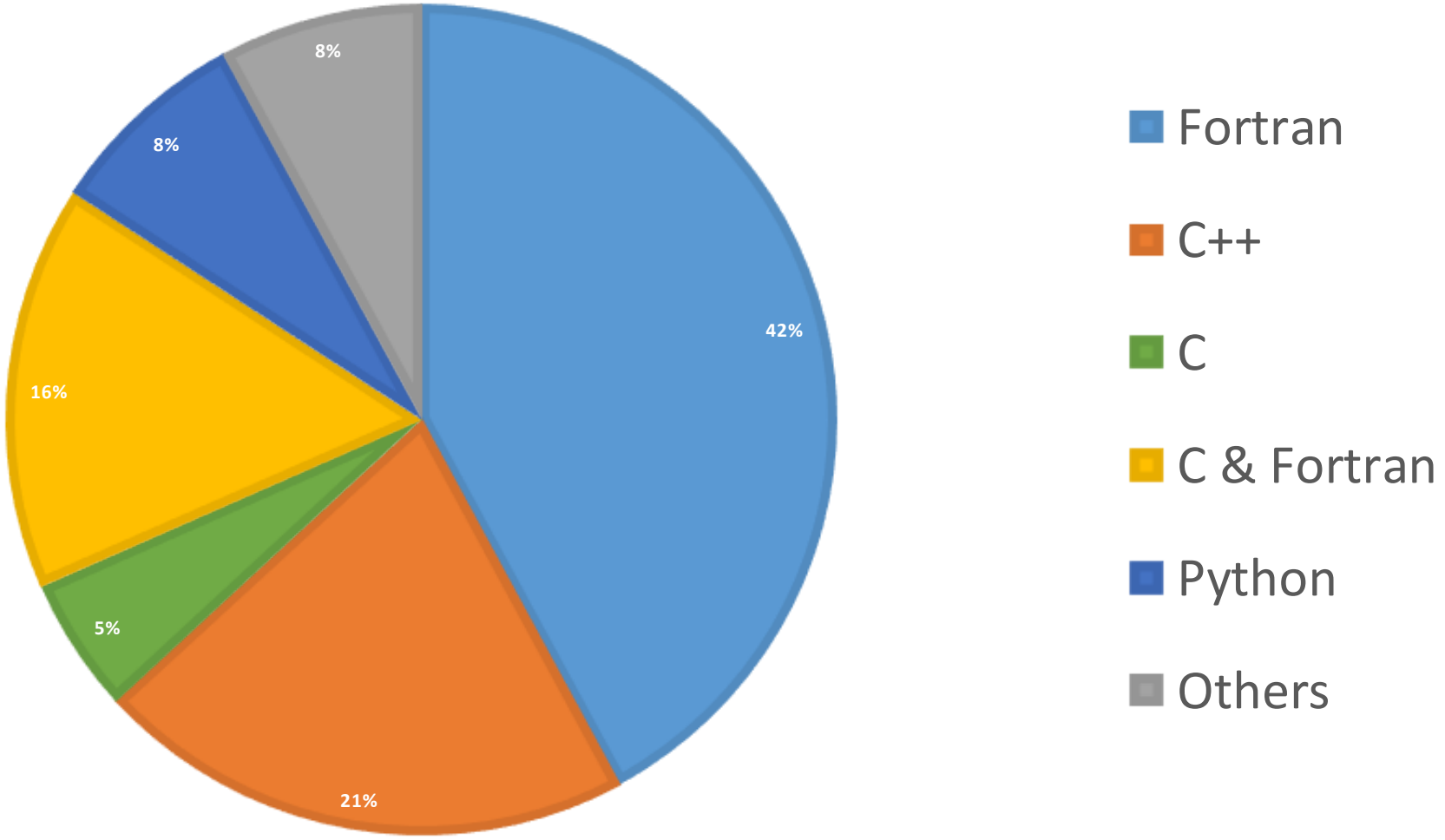
- Overview of codes investigated
- Outline of an audit
- Code audit examples
- Our first proof of concept projects
- Examples of impact
- Summary

# Customers by Country

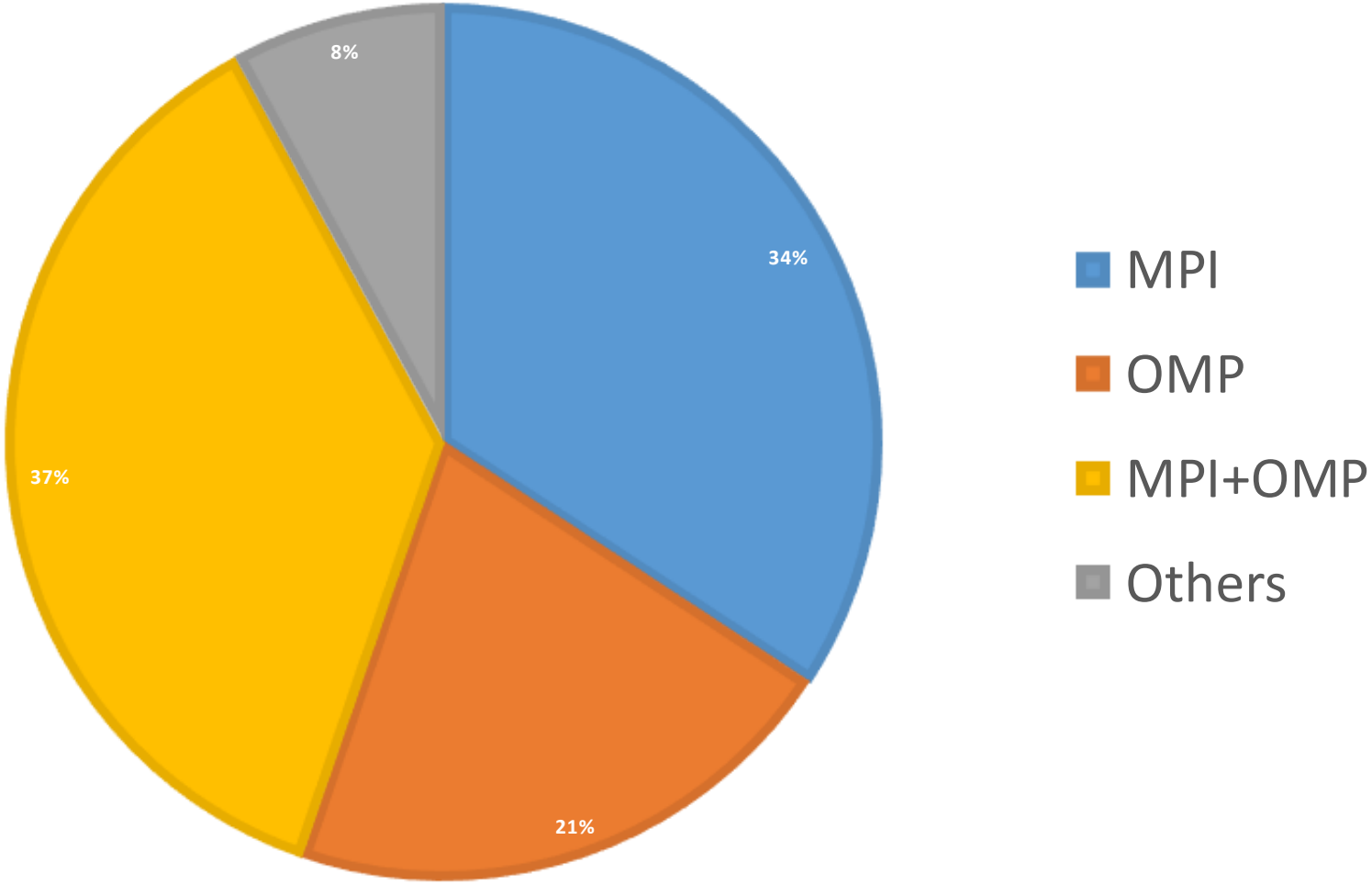



- UK
- Sweden
- Germany
- Belgium
- Luxembourg
- Spain
- France
- Italy

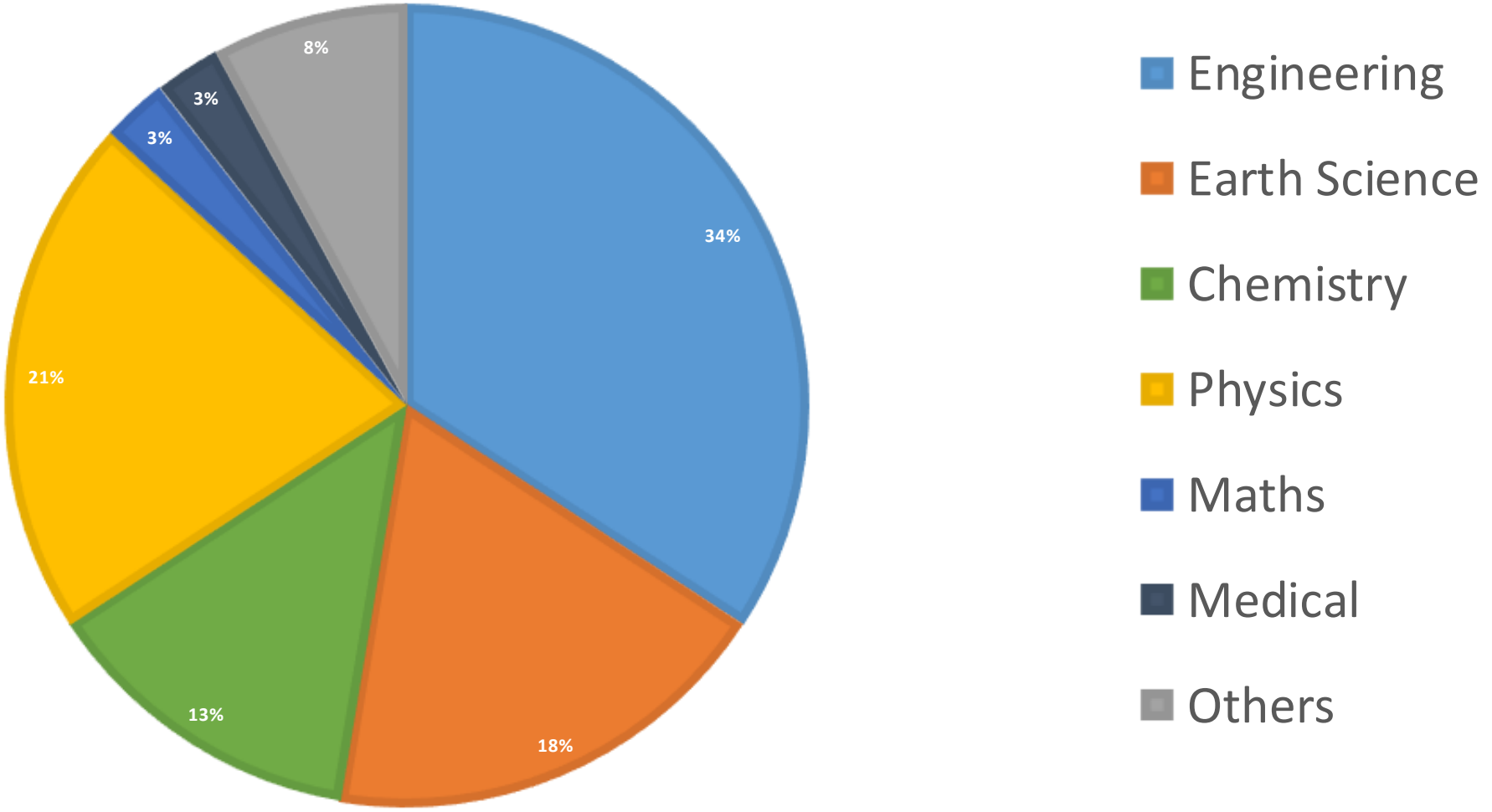
# Programming Languages



# Parallelisation Scheme



# Subject Areas



# 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) |

# The POP Audit

- POP users help us to prepare for analysis runs (e.g. identify representative inputs, help to compile code)
- Either POP staff do analysis runs if they have access to machines and codes, or users can do runs themselves and supply traces to POP
- Traces are analysed to produce efficiency metrics and to help identify underlying causes of any inefficiencies
- Result is a written report to the user presenting the results of this analysis, including the calculated metrics

# Outline of a Typical Audit Report

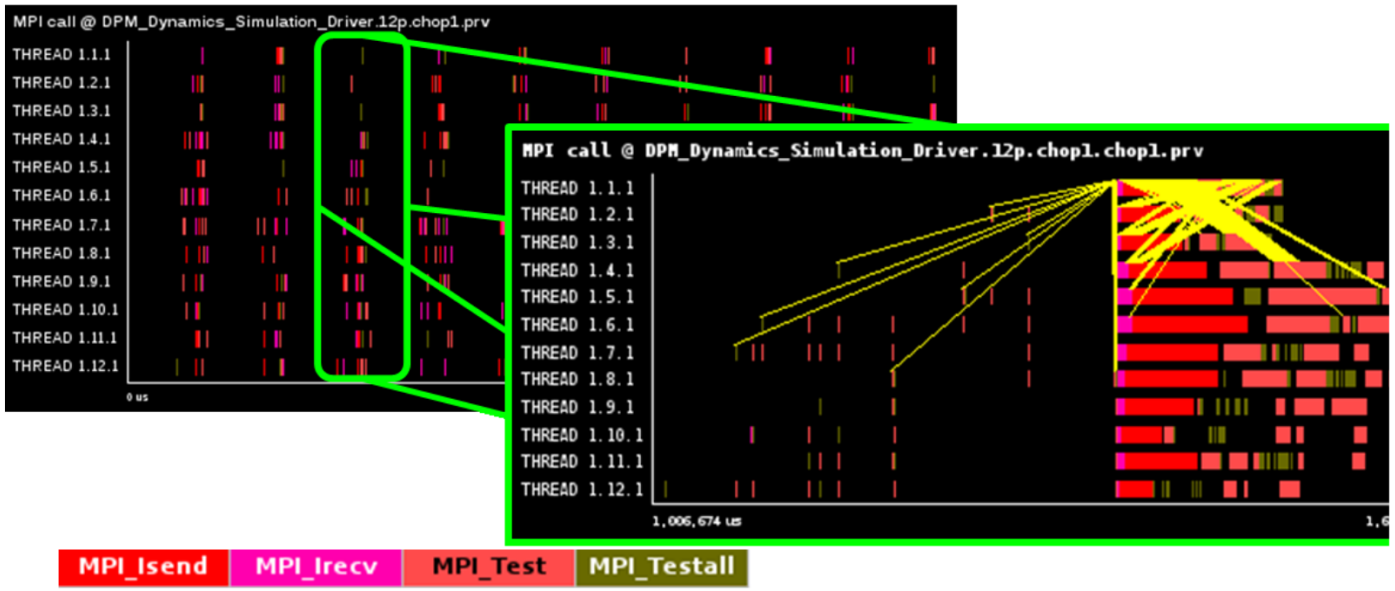
- Application Structure
- (if appropriate) Region of Interest
- Scalability Information
- Application Efficiency
  - e.g. time spent outside MPI
- Load Balance
  - Whether due to Source code or external factors
- Serial Performance
  - Identification of poor code quality
- Communications
  - e.g. sensitivity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



# 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

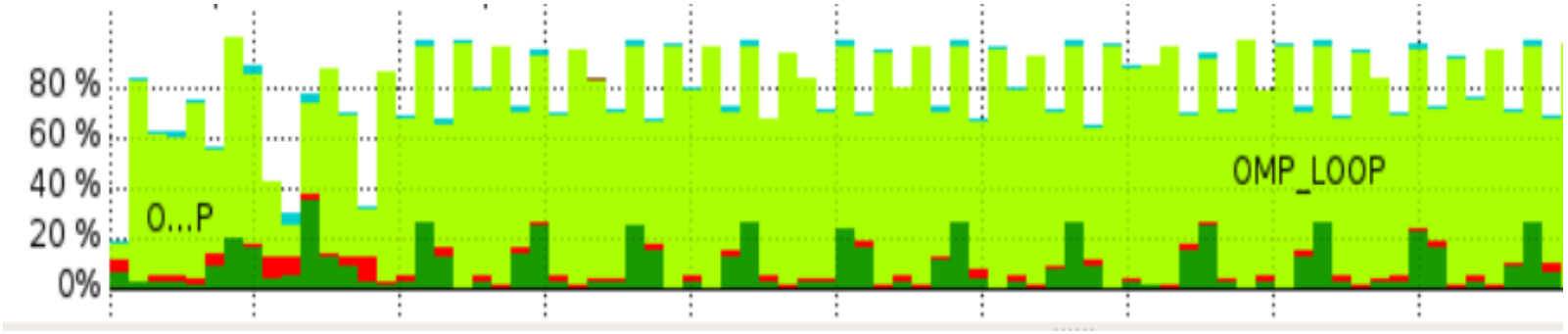


# VAMPIRE – University of York

- 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

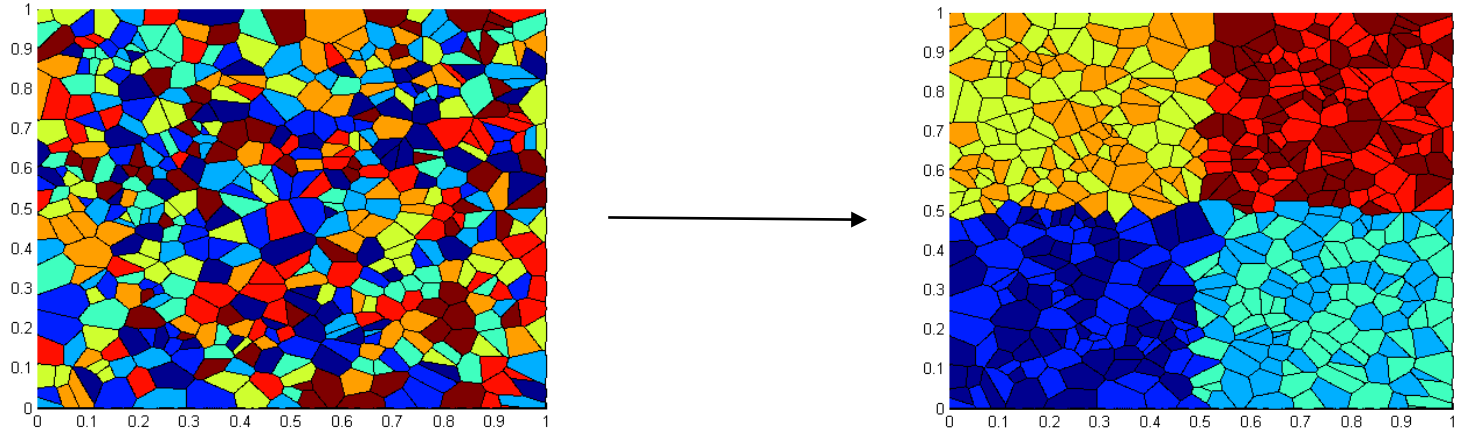
# GraGLeS2D – RWTH Aachen

- 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

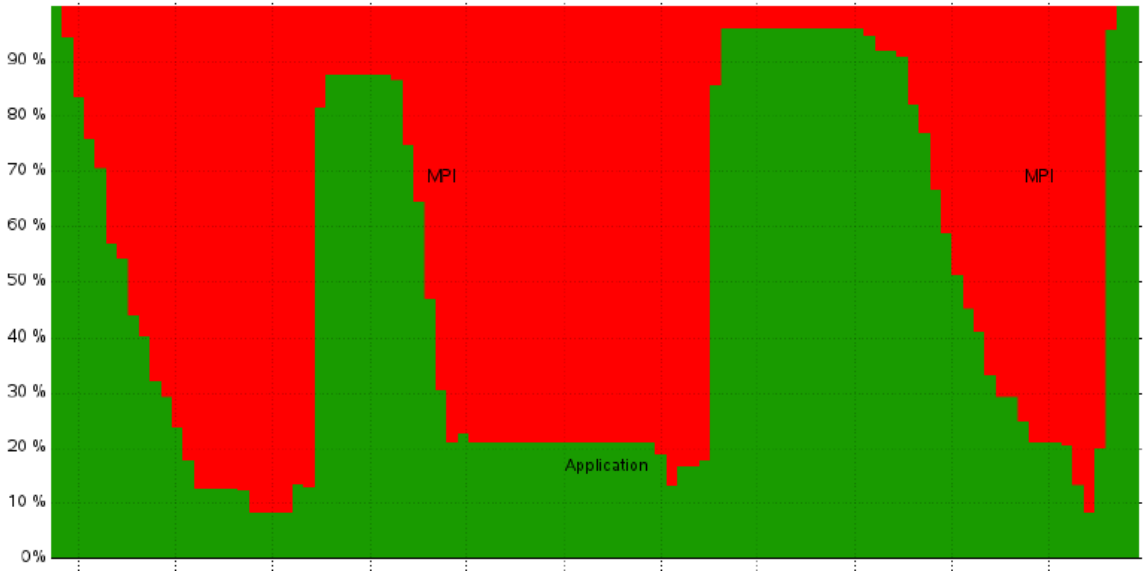
- Improvements:
  - Restructured code to enable vectorisation
  - Used memory allocation library optimised for NUMA machines
  - Reordered work distribution to optimise for data locality




- 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

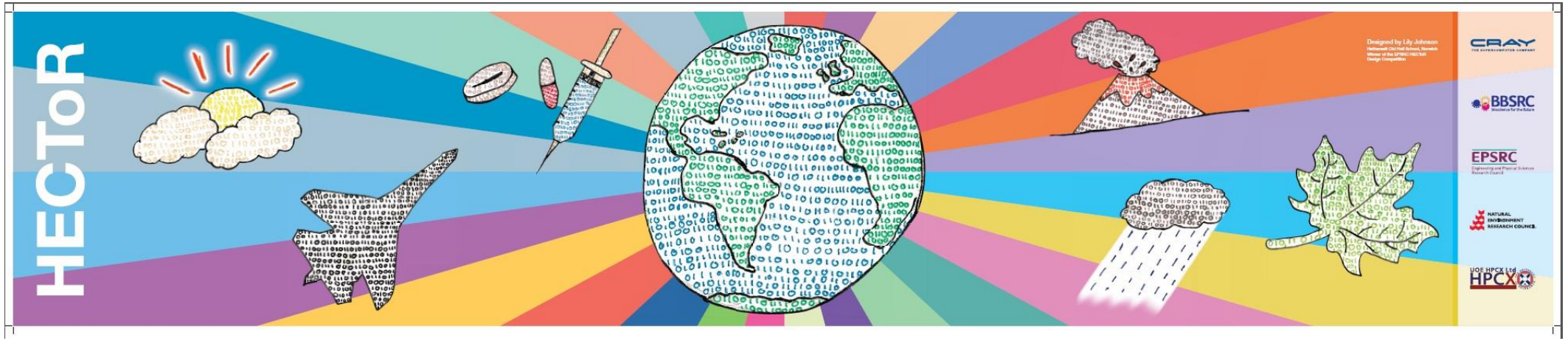


# Ateles – Proof-of-concept

- 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

# Examples of impact of code improvement




- Projects undertaken as part of the HECToR service in the UK 2007-2014

# Financial impacts

- CASINO: Molecular simulation code, used to simulate thermodynamic effects at the earth's core
  - Improved memory footprint per node using shared memory
  - Introduced hybrid parallelism with OpenMP
  - Improved parallel I/O

- More efficient use of resources led to £2m saving in compute costs.

# Scientific Impacts

- CASTEP & ONETEP: Calculate properties of materials from first principles using Density Function Theory
  - Improvements to memory scaling through new algorithm
  - Made better use of shared memory

- Original code limited to 3000 atoms
- Could now deal with 100,000 atoms, enabling
  - CASTEP: study of larger structures such as grain boundaries
  - ONETEP: study of larger molecules such as proteins and DNA segments

# Summary

- POP seeks to not only describe the performance of an application, but to identify the root causes of poor performance.
- Better performance leads to both resource savings and improved science.
- POP is a free service for people and organisations in the European Union.

https://pop-coe.eu

"The audit of the VAMPIRE code has been **extremely helpful** in identifying the hot spots and specific areas to focus on performance improvements. Preliminary results suggest this may give a **factor of 2 performance improvement** on modern CPUs. I would **highly recommend** the service for the speed and usefulness of the audit."

- **Richard Evans**, VAMPIRE developer