POP: The Story So Far

Sally Bridgwater, NAG Ltd

EU H2020 Center of Excellence (CoE)

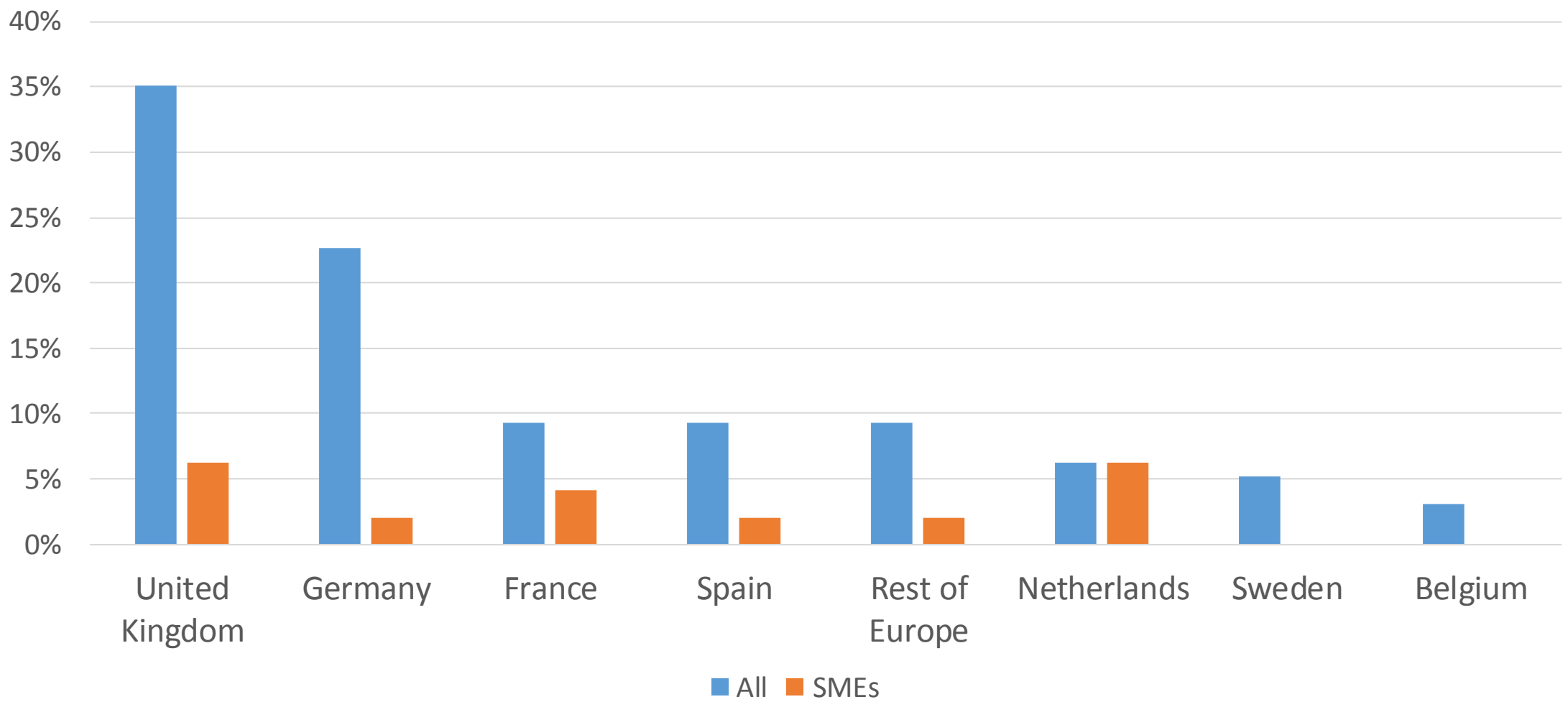
1 October 2015 – 31 March 2018

Grant Agreement No 676553

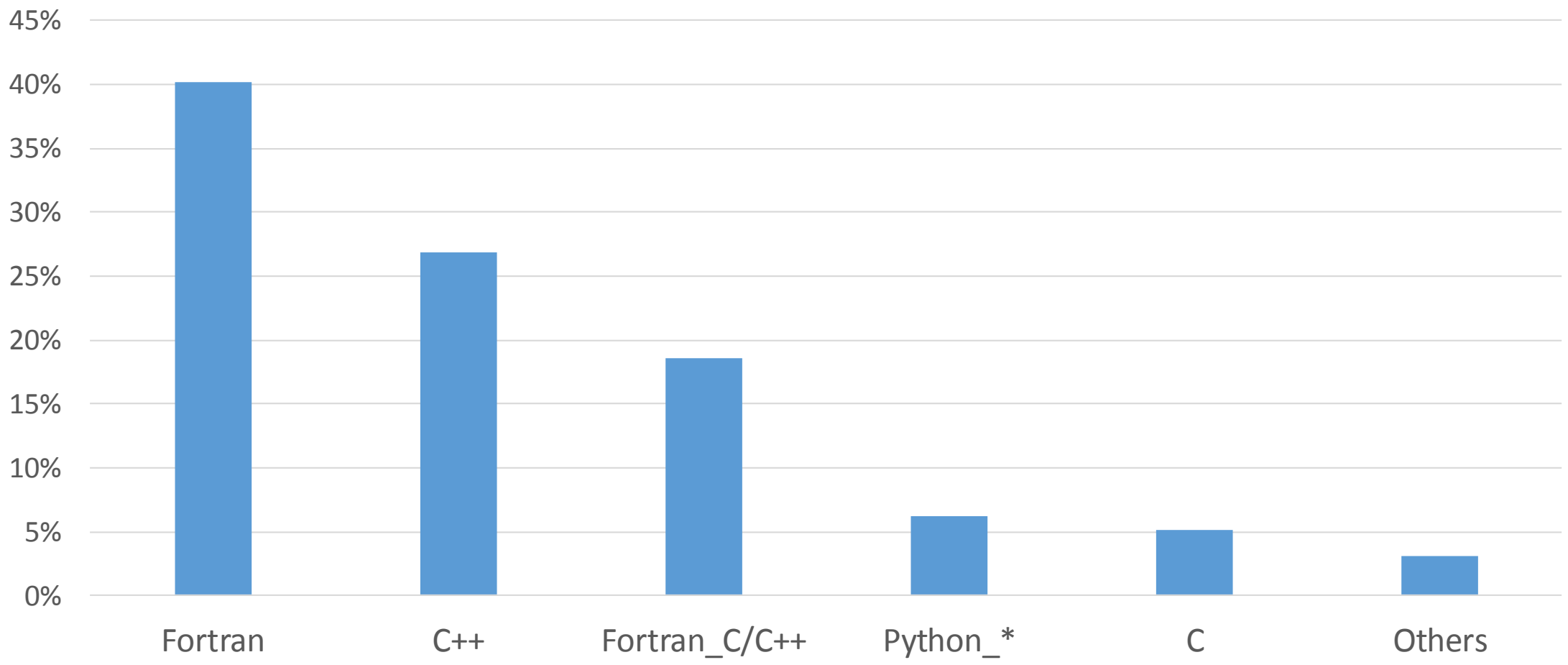
Outline

- Overview of codes investigated
- Code audit & plan examples
- Meta-analysis of inefficiencies
- Proof of concept projects
- Summary

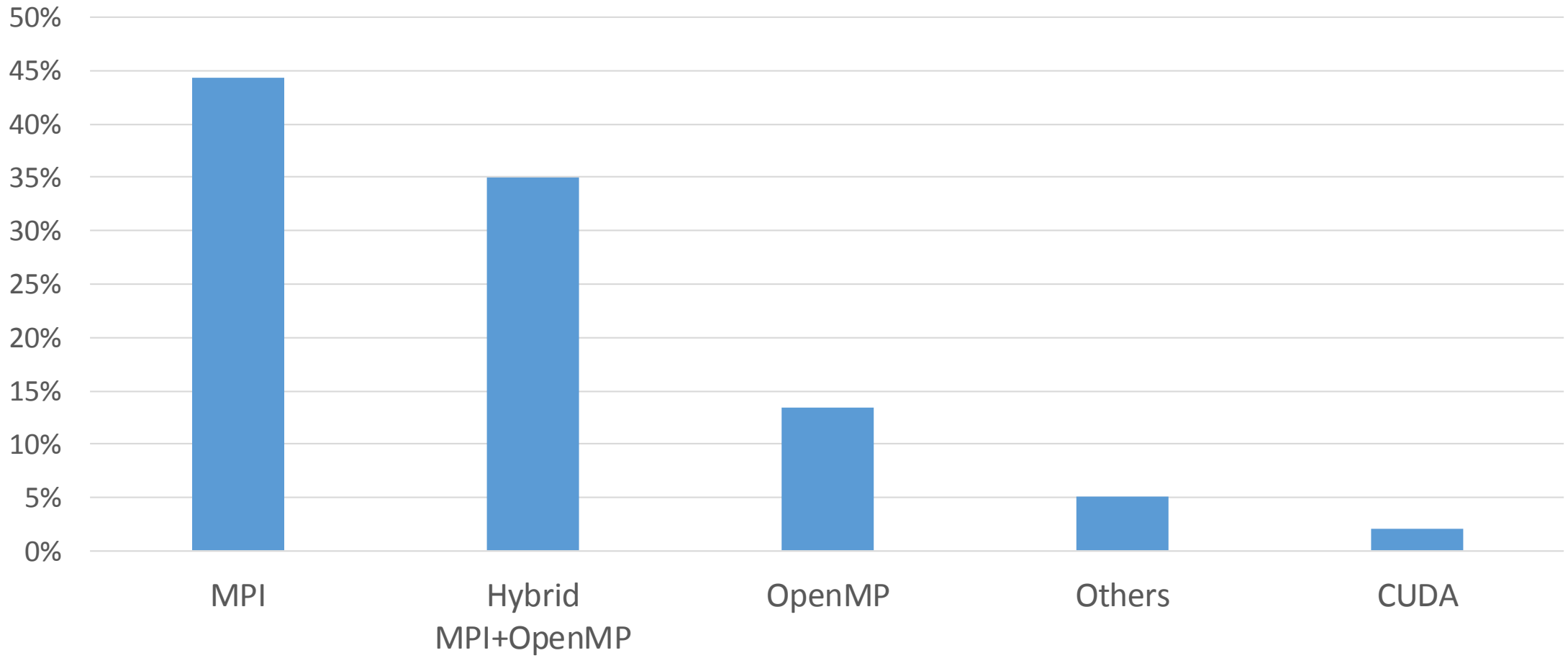
Customers by Country



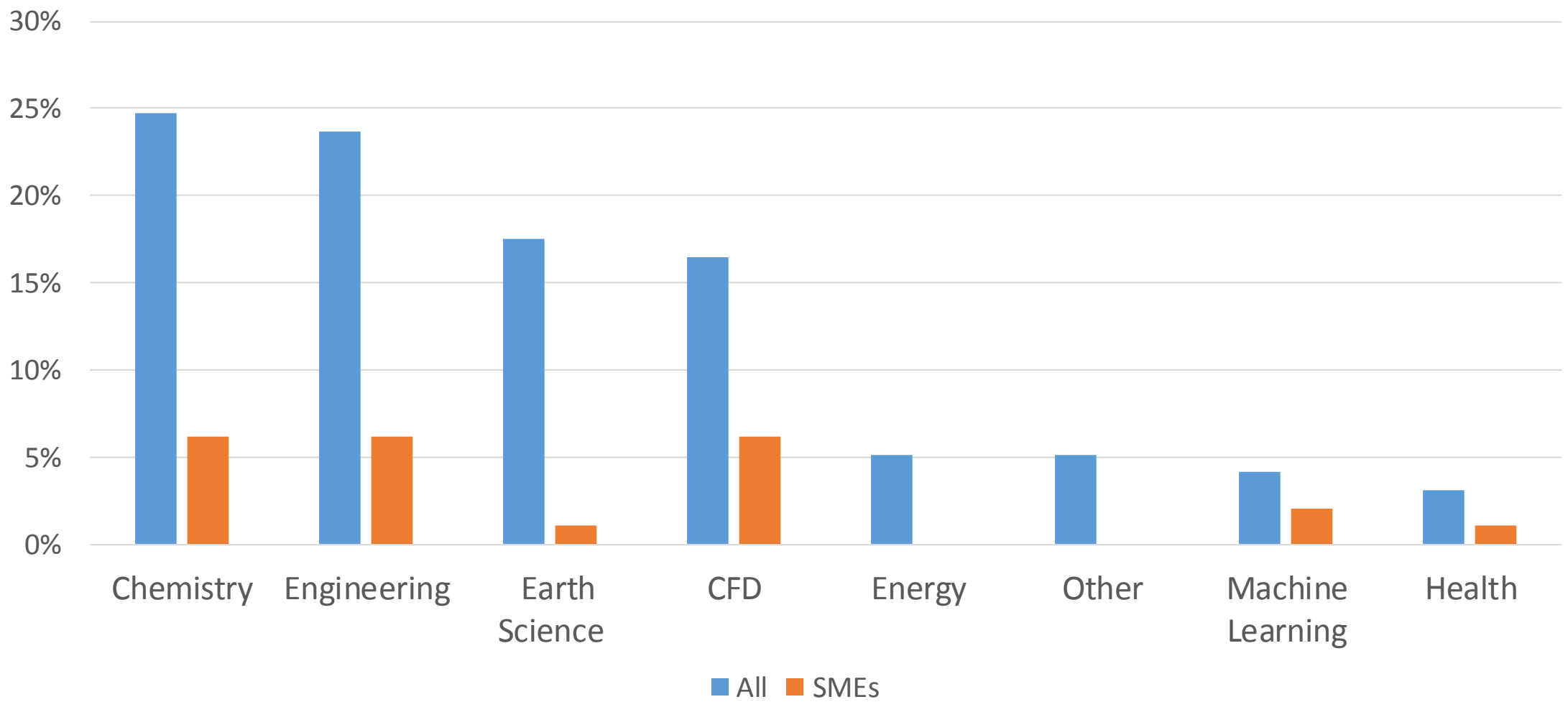
Programming Languages



Parallelisation Scheme



Application Sectors



So Far

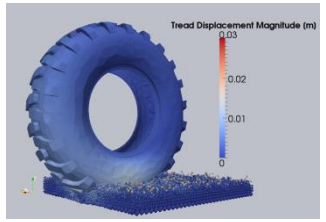
- 62 Audits and plans completed or reporting to customer
- 5 completed Proof of Concepts
- Working on a further 40 studies
- Close to 40% of Audits lead to a follow up Performance Plan

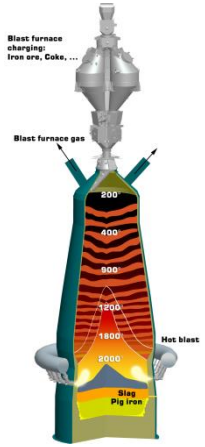
- Goal – 150 assessments

Code Audit & Plan Examples

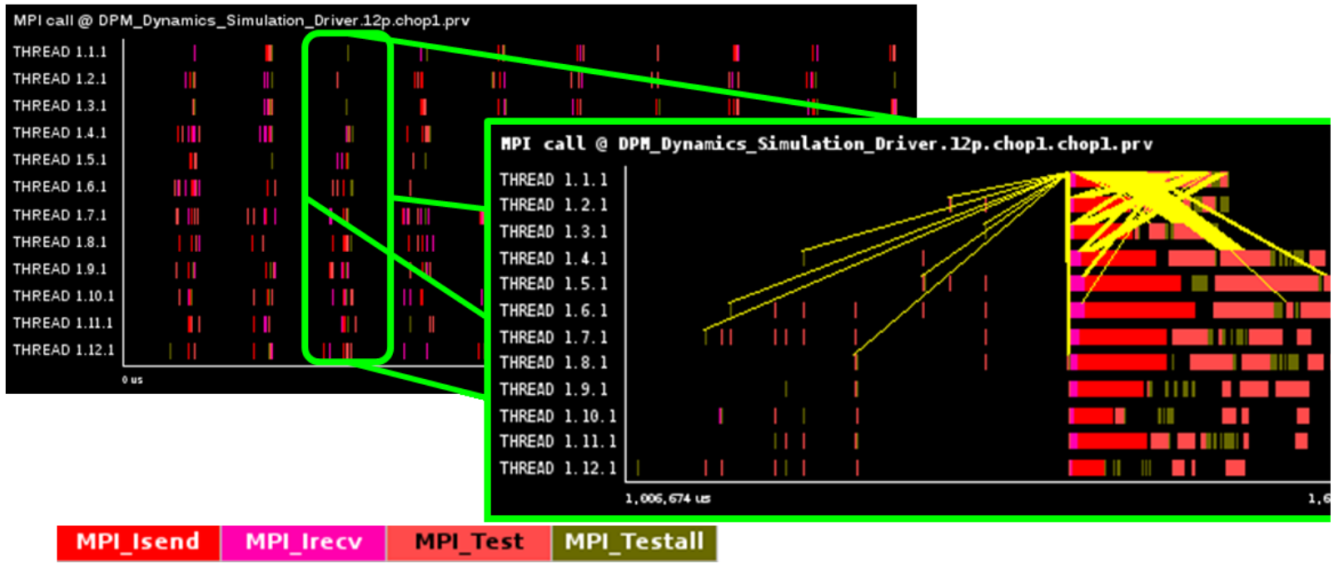
DPM – University of Luxembourg

- Numerical simulation tool using the Discrete Particle Method.
- Applied to blast or biomass incineration furnaces, snow-tire interaction …
- C++ code parallelised with MPI



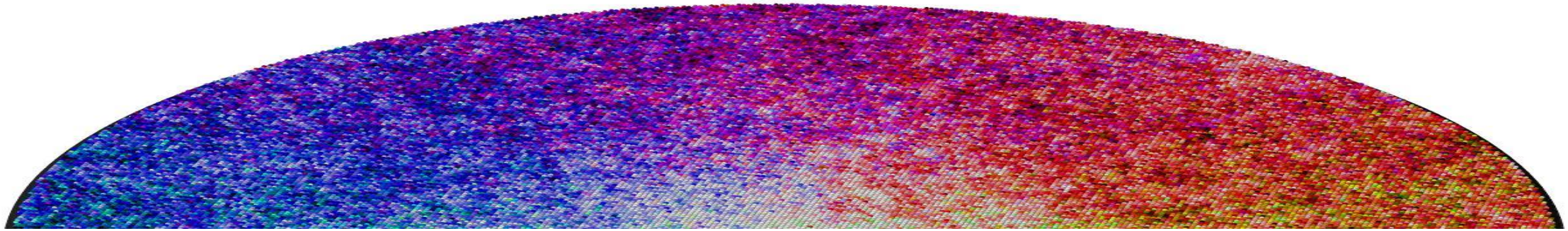


- Key audit results:
  - Performance problems were due to the way that the code had been parallelised
  - Scalability limited by end-point contention due to sending MPI messages in increasing-rank order



VAMPIRE – University of York

- Magnetic materials simulation code
- C++ code parallelised with MPI
- Key audit results:
  - Poor computational efficiency
  - 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



OpenNN - Artelnics

- Neural network open source application
- C++ code with OpenMP parallelisation

- Key audit results:
  - Poor computational efficiency
    - Main limit on performance is the unexpected variability on the number of times a parallel loop was executed.
  - Using hyper-threading was found to give a reasonable performance boost
  - Further work in a POP performance plan is investigating the unexpected extra computation

ADF – SCM

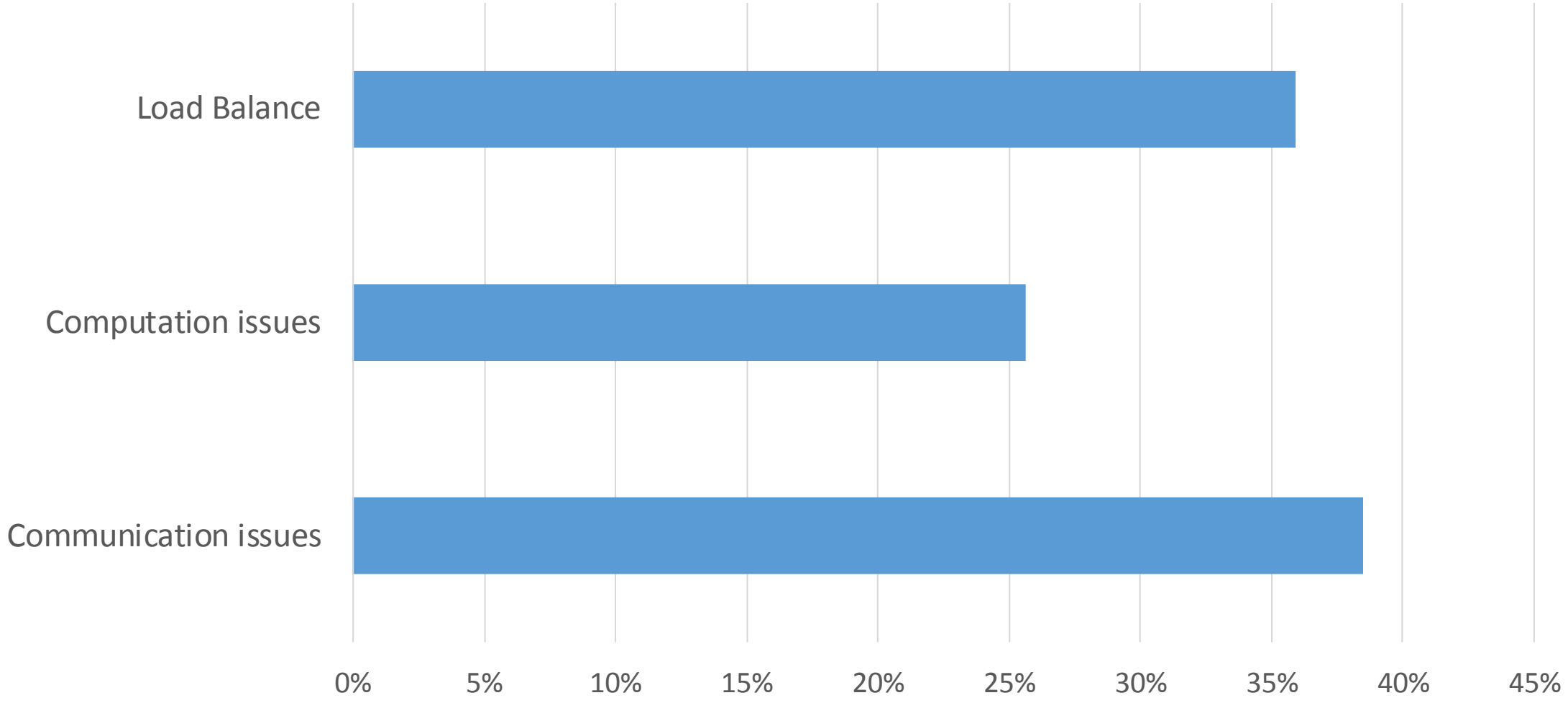
- Computational chemistry application
- Fortran with MPI and low-level shared arrays
- Key Plan results of Load Balance invesitgation:
  - Located unequal division of work
  - Work sharing amongst ranks was not frequent enough -> time spent waiting
  - Potential for up to a factor of two performance improvement

SCM
Software for Chemistry & Materials
www.scm.com

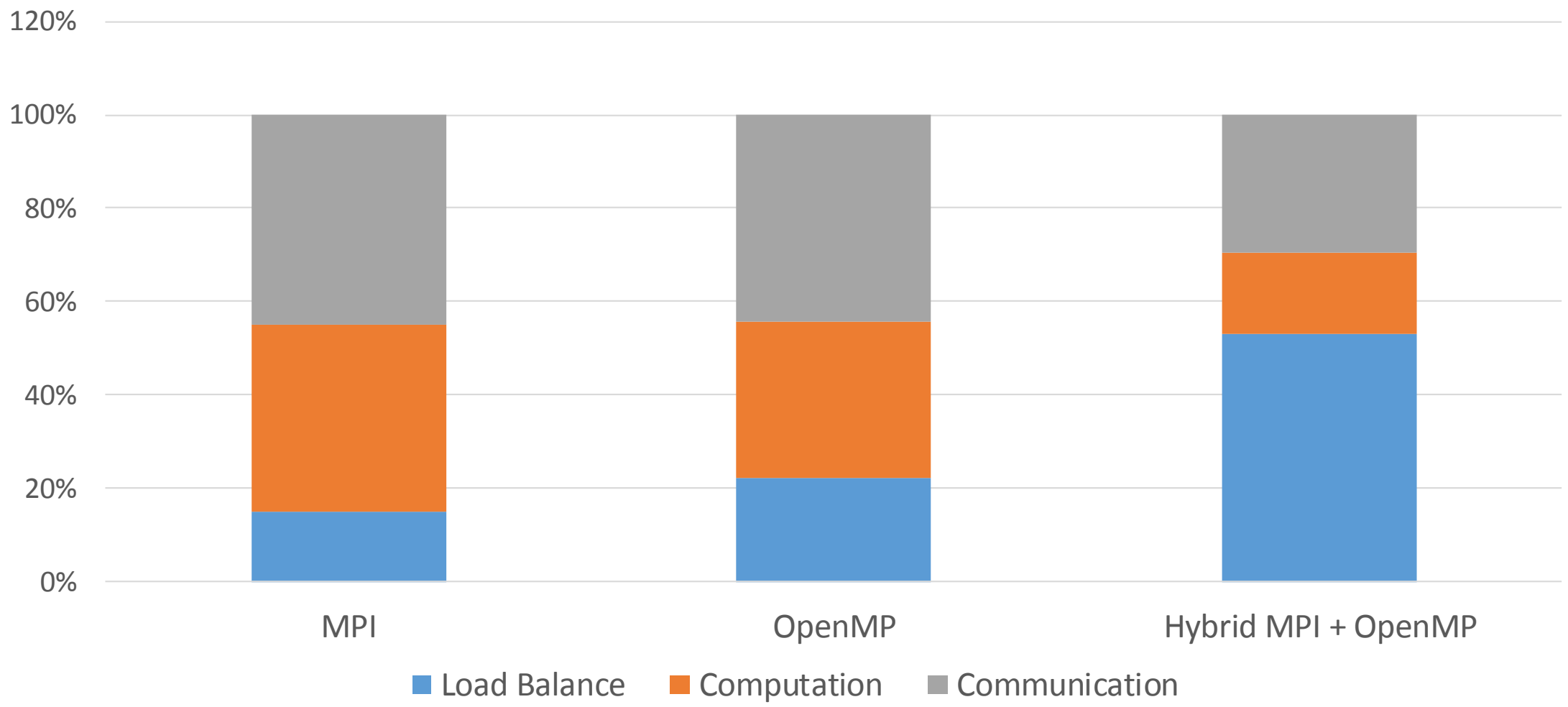
- Code changes implemented by the developers and released in their most recent update

Analysis of Inefficiencies

Leading cause of inefficiency



Inefficiency by Parallelisation



Proof of concepts

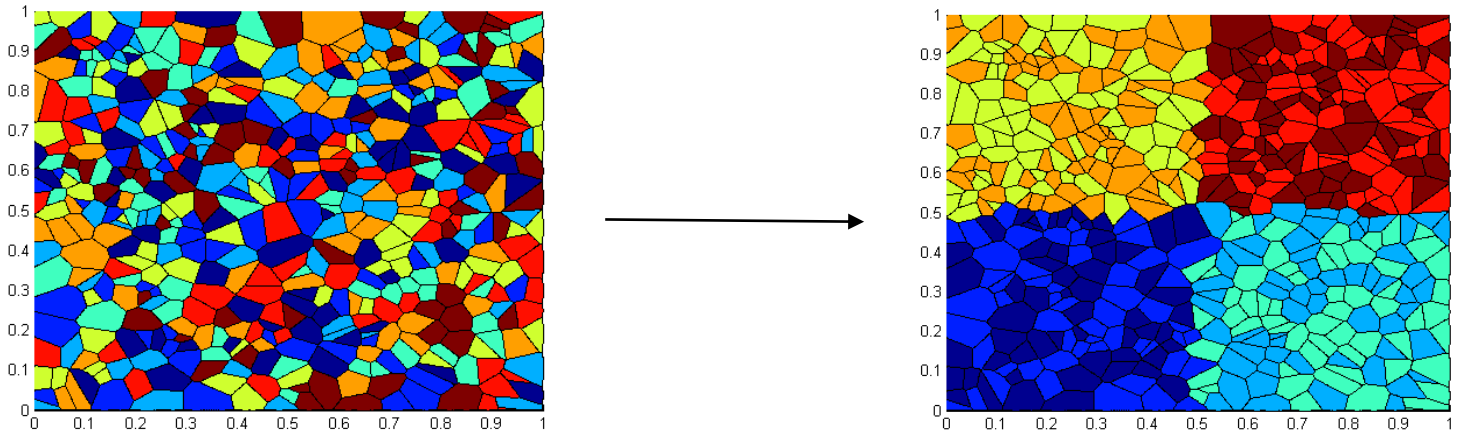
GraGLeS2D – RWTH Aachen

- 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

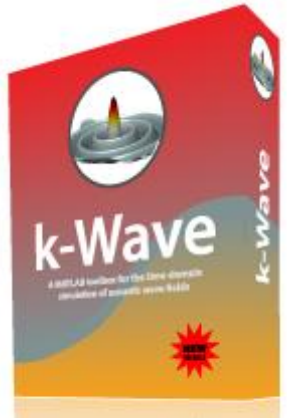
- Improvements:
  - Restructured code to enable vectorisation
  - Used memory allocation library optimised for NUMA machines
  - Reordered work distribution to optimise for data locality



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

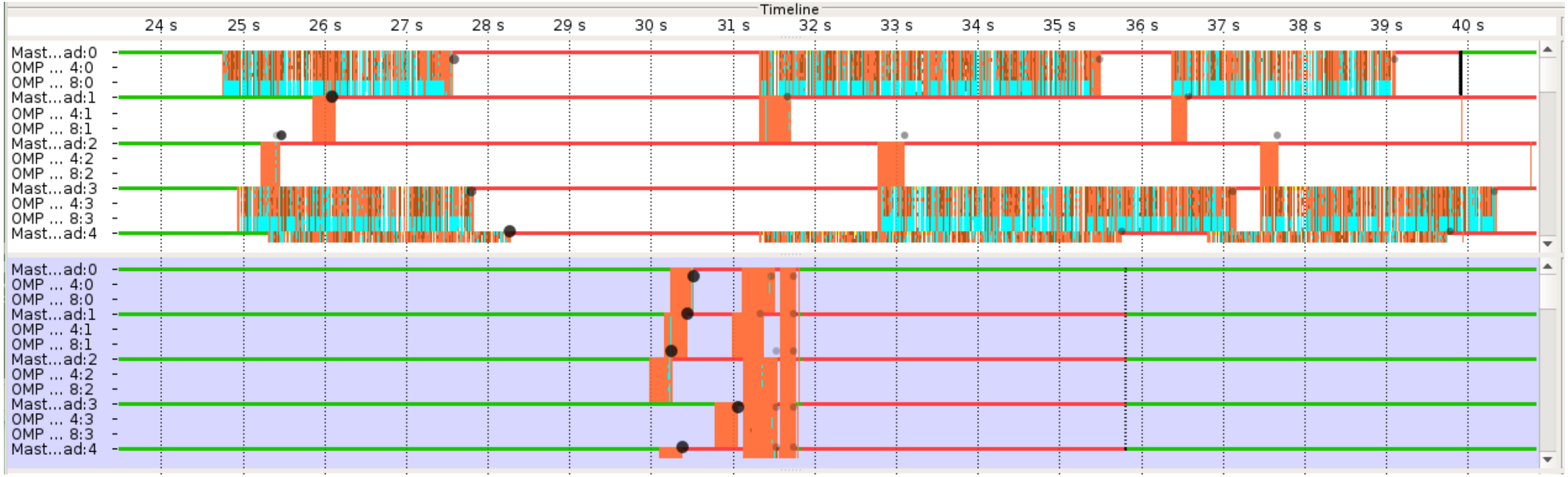
k-Wave – Brno Uni. of Technology

- Toolbox for time domain acoustic and ultrasound simulations in complex and tissue-realistic media
- C++ code parallelised with Hybrid MPI and OpenMP (+ CUDA)
- Executed on Salomon Intel Xeon compute nodes
- Key audit findings:
  - 3D domain decomposition suffered from major load imbalance : exterior MPI processes with fewer grid cells took much longer than interior
  - OpenMP-parallelised FFTs were much less efficient for grid sizes of exterior, requiring many more small and poorly-balanced parallel loops
- Using a periodic domain with identical halo zones for each MPI rank reduced overall runtime by a factor of 2



www.k-wave.org

k-Wave – Brno Uni. of Technology

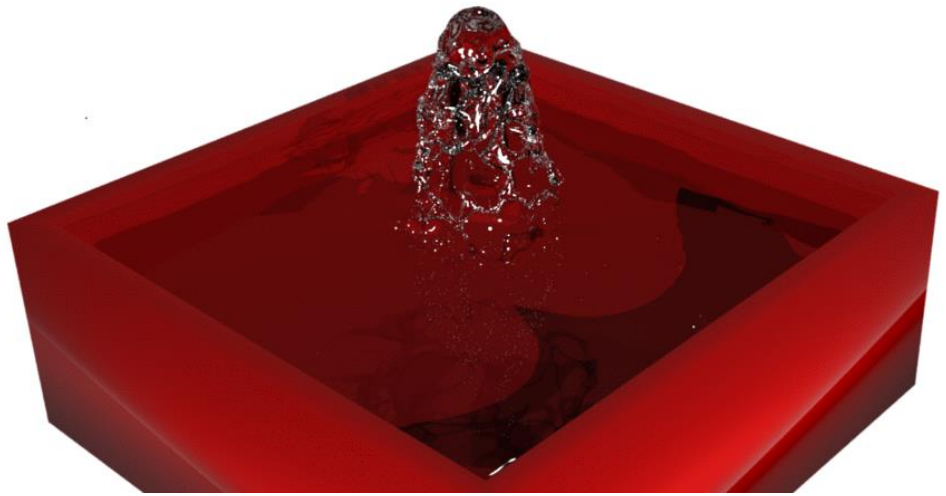


- Comparison time-line before (white) and after (lilac) balancing, showing exterior MPI ranks (0,3) and interior MPI ranks (1,2)
  - MPI synchronization in red, OpenMP synchronization in cyan

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

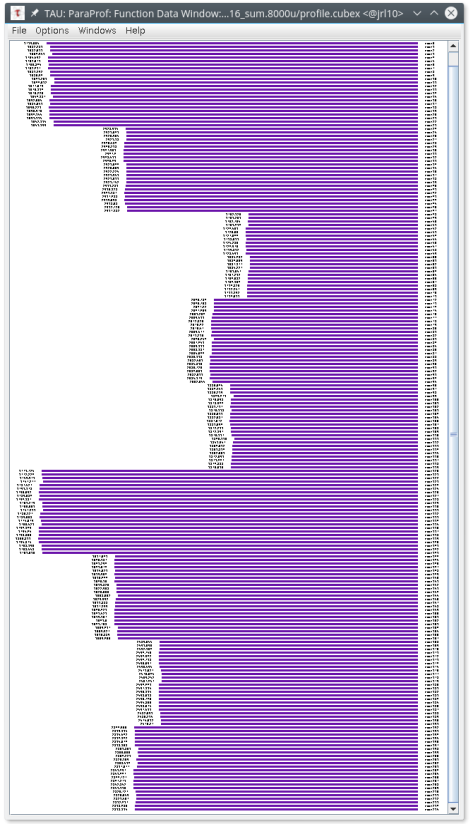
- Implemented by the code developers:
  - Review of overall code design from issues identified in POP audit
  - Inlining short functions
  - Reordering the particle processing order to reduce cache misses
  - Removal of unnecessary operations and costly inner loop definitions

- 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)

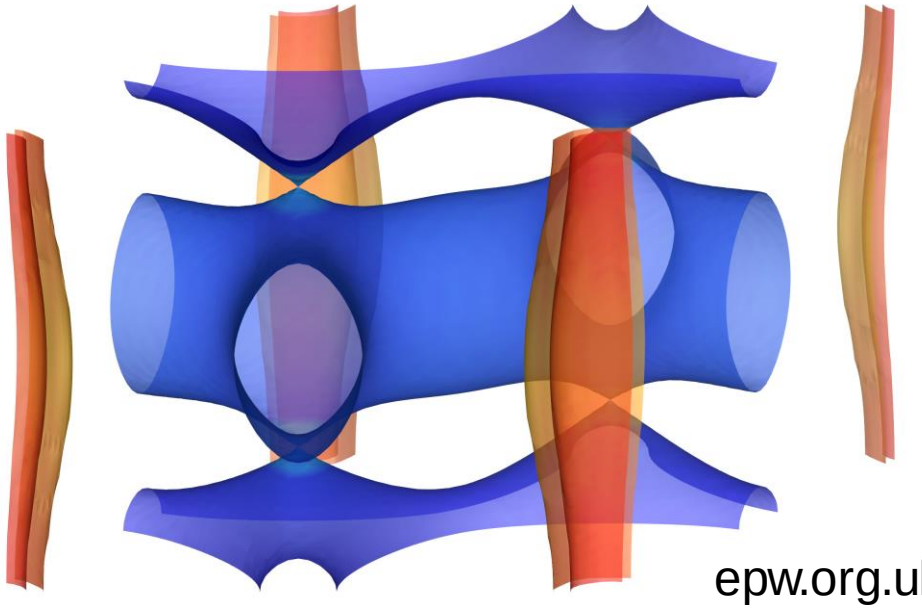
- Key audit findings:
  - Poor load balance from excessive computation identified
  - (addressed in separate POP Performance Plan)
  - Large variations in runtime, likely caused by IO
  - Final stage spends a great deal of time writing output to disk
- Report used for successful PRACE resource allocation



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



Future plans

- Webinars –
  - Advantages of Profiling
    - 15th June
    - Sign up available soon on website
    - Why you want to know what your code actually does
    - Motivated with examples of unexpected insights gained via profiling
  - Performance from OpenMP programs on NUMA architectures
    - Date tbd
    - Understanding the characteristics of cc-NUMA architectures
    - practical examples illustrating best practices
  - More to be added

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.
- Current funding secured until March 18 – apply now for full range of services

https://pop-coe.eu