Getting Insight into HPC Code Behaviour

Meet the POP CoE:
Fouzhan Hosseini, NAG Ltd

Who is NAG?

Renowned in Numerical Algorithms & HPC

Over the last 50 years, brands such as AMD, Arm, Intel, and Nvidia have become reliant on NAG IP

Global SME

With hubs in Asia, Europe and the US, NAG can support clients in their own time zone

NAG provides

Solutions

- NAG Library & custom algorithm development
- Fortran compiler
- Algorithmic differentiation

Consultancy

- Cloud HPC migration
- HPC technology evaluation & benchmarking
- Code porting & optimisation

nag®

A Centre of Excellence in HPC

Performance Optimisation and Productivity

- Better Parallel Code
- Boost scalability of HPC Applications
  - Faster results, novel solutions, reduced expenditure
  - More efficient use of HPC infrastructures
  - In prep for Exascale computing
- A Team with
  - Excellence in performance tools and tuning
  - Excellence in programming models and practices
  - R & D background in real academic and industrial use cases

What & who for

Methodology & Tools

- Quantitative picture of application behavior
- Hierarchical set of Performance metrics
- Open-source tools that support high-scale analysis

Communities

- All domains of science & technology
- SMEs or large industries
- Academia & Research institutes
- All other HPC CoEs
- HPC Centres

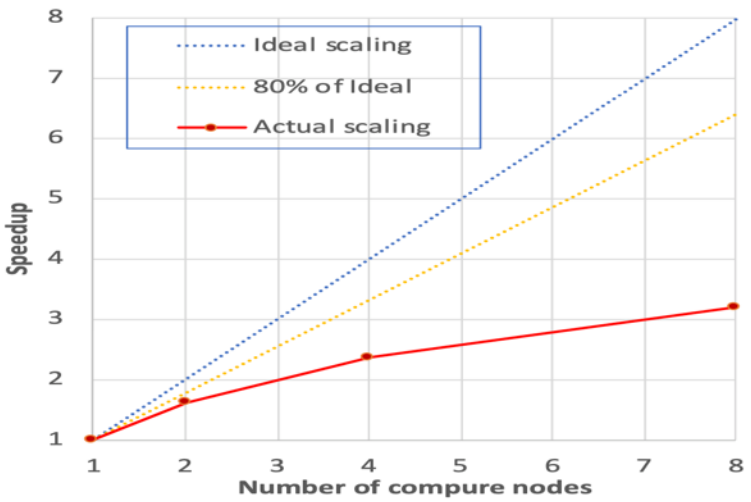
Services

- Performance assessment
- Code Optimization
- Transversal across all programing models, architectures & scale
- Education & training

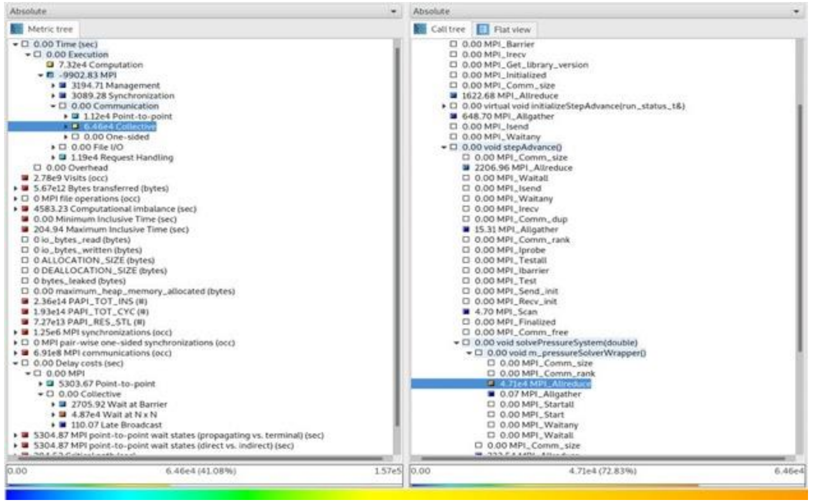
Parallel performance is hard to understand

How do we measure the performance of our parallel programs?

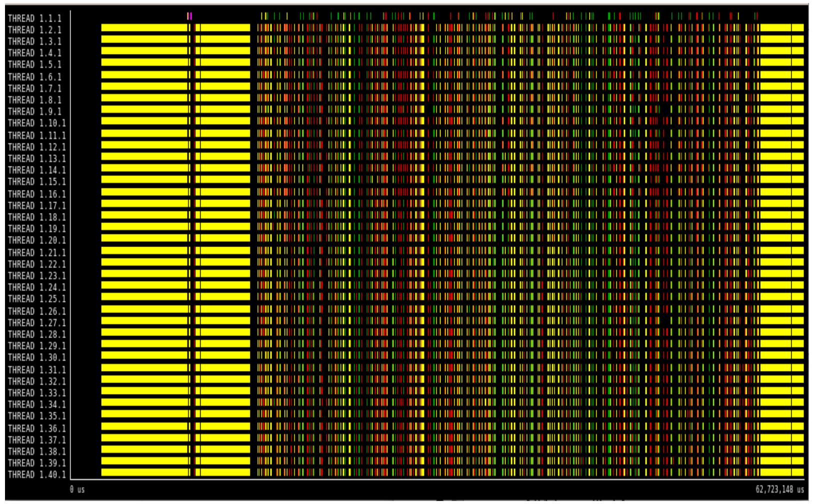
speed-up and efficiency plots?



collect data using performance monitoring tools?



Cube, perf. metrics per routines/call stack, data collected by Scalasca/Score-P



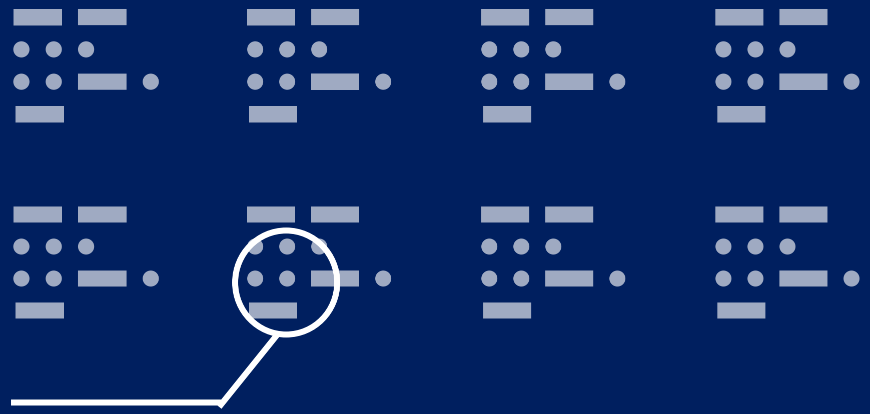
Paraver, timeline view of program execution, data collected by Extrae

Tracing is powerful

But can generates overwhelming amount of data

Main Problem

Lack of quantitative understanding of the actual behavior of a parallel application



Difficult to know where to start and what to look for

The POP metrics:

A simple but powerful solution

Devise a simple set of performance metrics...

- using values easily obtained from the trace data
- where low values indicate specific causes of poor parallel performance

These metrics then are used to understand...

- what are the causes of poor performance
- what to look for in the trace data

POP MPI Parallel Efficiency Metrics

CT = Computation Time | **TT** = Total Time | Ideal network data transfer takes zero time

Global Efficiency (GE)

Describes how well your parallel application scales.

GE = PE × Comps

Parallel Efficiency (PE)

Describe how well the execution of your code is working in parallel

PE = LB × CommE

Computational Scaling (CS)

Describe how well the computational load of an application scales with the number of processors/threads.

CS=CT_O/CT_n

Instruction Scaling

Compares the total number of instructions executed for different numbers of threads/processes.

Communication Efficiency (CE)

Reflects the loss of efficiency by communication.

CE = max (CT/TT) = SE × TE

Load Balance (LB)

Measures how well work is distributed to threads/processes in the application.

LB = average (CT) / max (CT)

IPC Scaling

Compares how many instructions per cycle (IPC) are executed for different numbers of threads/processes.

Transfer Efficiency (TE)

Describes the loss of efficiency due to actual data transfer time.

TE = TT on ideal network / TT

Serialisation Efficiency (SE)

Describes efficiency loss due to dependencies, waiting in MPI calls, ets.

SE = max(CT / TT on ideal network)

Frequency Scaling

Compares the processor frequency for different numbers of threads/processes.

For more details visit https://pop-coe.eu

POP performance monitoring tools

- Developing open-source tools
  - Extrae (tracing), Paraver (visualisation) & Dimemas
    https://tools.bsc.es
  - Score-P (profiling and tracing), Scalasca (Post Processing) & Cube (visualisation) https://tools.bsc.es/
  - MAQAO: synthetic reports and hints with a focus on core performance http://www.maqao.org
  - PyPOP: automated generation of POP metrics from Extrae traces
    https://github.com/numericalalgorithmsgroup/pypop

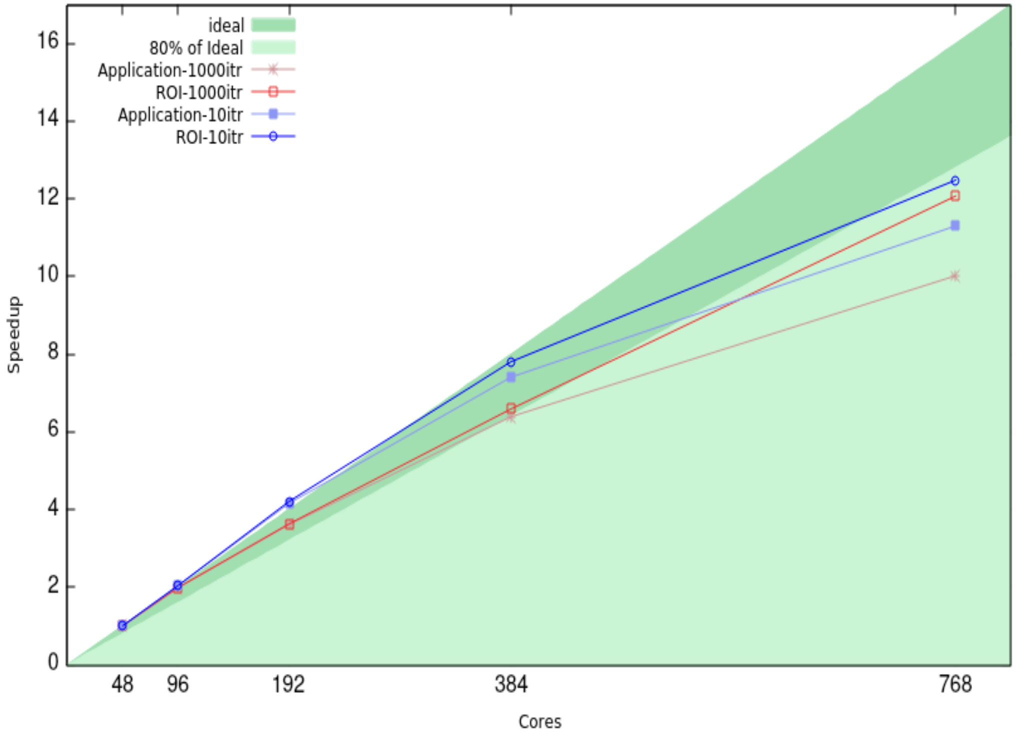
For more help on how to use these tools and calculate the POP metrics see the POP website learning material & online training

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

Other tools can also be used

Example 1

A Computational Fluid Dynamics Code



- Code: C++, MPI
- Platform: MareNostrum-IV
  - Dual Intel Xeon Platinum 8160 Skylake 48-core nodes
- Performance data collection:
  - *Score-P/Scalasca* using compiler instrumentation filter and hardware counters
- Scale: 48-768 cores (1-16 nodes)

Example 1

POP Metrics

Number of cores	48	96	192	384	768
Global Efficiency	0.93	0.94	0.93	0.84	0.76
Parallel Efficiency	0.93	0.91	0.87	0.77	0.68
Load balance	0.99	0.98	0.98	0.97	0.95
Communication Efficiency	0.94	0.92	0.89	0.79	0.72
Serialisation	0.95	0.94	0.92	0.85	0.81
Transfer efficiency	0.99	0.99	0.97	0.94	0.89
Computational Scaling	1.00	1.03	1.07	1.09	1.12
Instruction Scaling	1.00	0.99	0.97	0.95	0.92
IPC Scaling	1.00	1.05	1.10	1.18	1.27
Frequency Scaling	1.00	1.00	1.00	0.98	0.96

We immediately see that Serialisation is the main factor that limits the scalability

Example 1

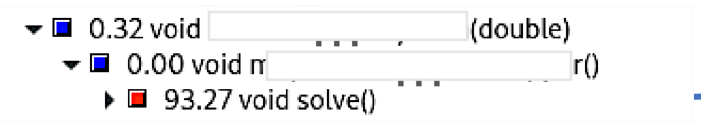
Cause of Low Serialisation Efficiency

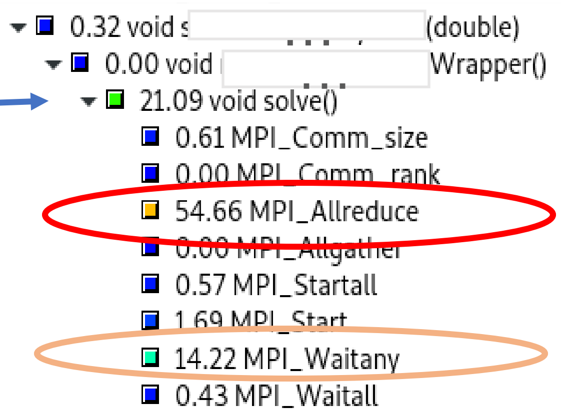
Serialisation

- typically happens due to at least one process arriving early/late at synchronization point

Scalasca calculates a delay cost metric

- This metric highlights the root causes of serialization





Numbers report percentage of total delay cost for Example 2 - ROI on 768 cores

Inclusive values (▶)
Exclusive values (▼)

The MPI collective calls and imbalanced computation regions within a Library call were the main causes of the serialisation on 768 cores

Example 2

Methodology & Tools

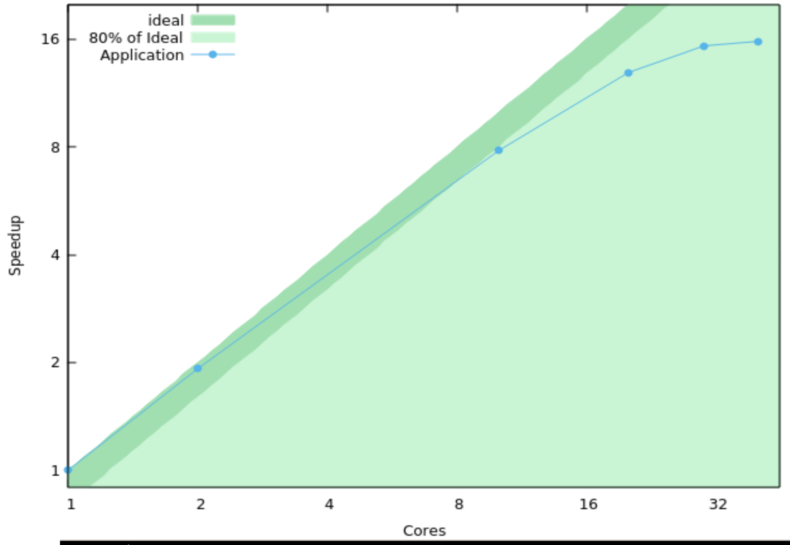
- Code: C++, Fortran, MPI
- No access to the source code

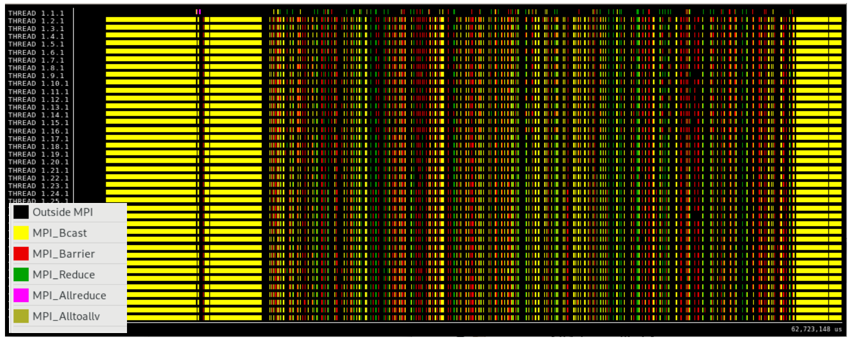
Platform:

- Dual Intel Xeon Gold 6248 CPU @ 2.50GHz – 40 cores
- Intel Fortran and C++ compiler with MKL and MPI Library (2019 version

Performance data collection: Extrae

- Scale: 2- 40 cores





Timeline of the program execution on 40 cores

Example 2

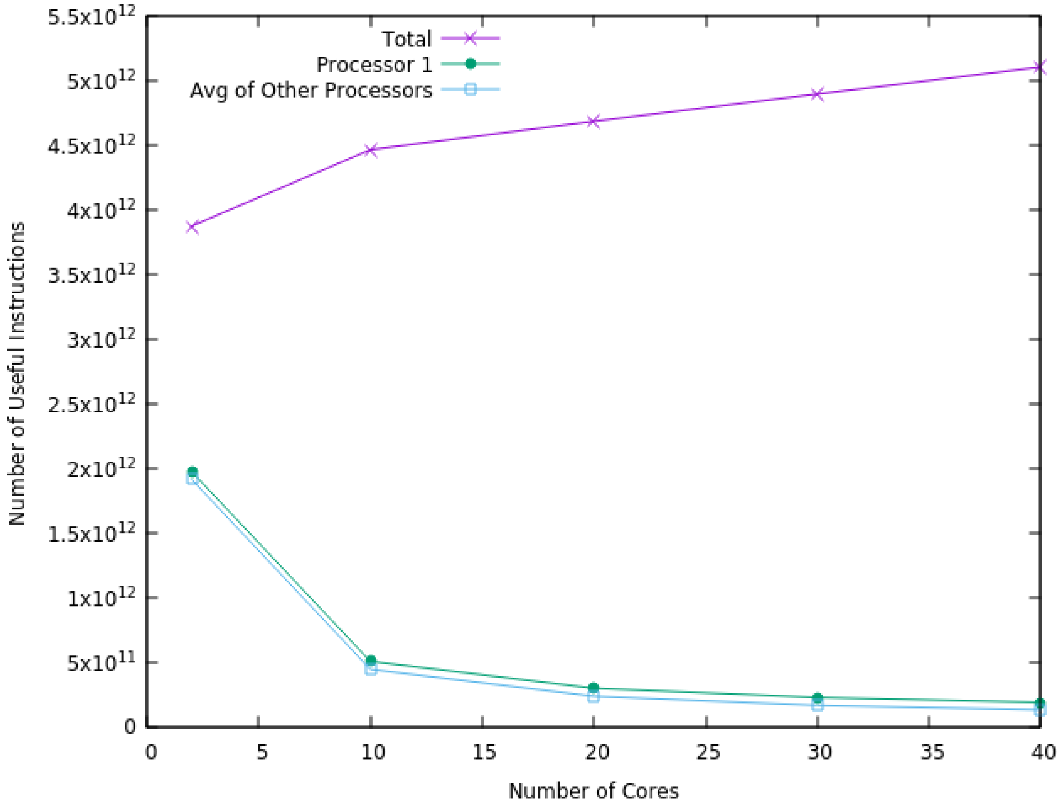
POP Metrics

Number of cores	2	10	20	30	40
Global Efficiency	0.95	0.73	0.60	0.47	0.36
Parallel Efficiency	0.95	0.89	0.81	0.75	0.68
Load balance	0.95	0.92	0.85	0.81	0.80
Communication Efficiency	0.99	0.97	0.95	0.92	0.85
Serialisation	1.00	0.99	0.99	0.98	0.94
Transfer efficiency	0.99	0.98	0.96	0.94	0.91
Computational Scaling	1.00	0.82	0.74	0.63	0.53
Instruction Scaling	1.00	0.87	0.83	0.79	0.76
IPC Scaling	1.00	0.99	0.95	0.90	0.83
Frequency Scaling	1.00	0.95	0.94	0.88	0.84

Load imbalance & increasing instruction count are major factors that limit the scalability

Example 2

Useful Instructions



Total number of useful instructions increases with increasing number of processes

- Low Instruction scaling

Process 1 always executes more instructions compared with other processes

- Load imbalance

With 40 processes, Processor 1 executes 46% more instructions with respect to average number of instruction per process

- Amdahl's law

Example 3

A Computational Fluid Dynamics Code

POP metrics from the *Performance Assessment*

# threads	1	10	30	45
Global Efficiency	1.00	0.80	0.36	0.26
↳ Parallel Efficiency	1.00	0.86	0.60	0.55
↳ OpenMP Region Efficiency	1.00	0.95	0.74	0.70
↳ Serial Region Efficiency	1.00	0.91	0.86	0.85
↳ Computational Scaling	1.00	0.94	0.60	0.48
↳ Instruction Scaling	1.00	1.01	1.00	1.00
↳ IPC Scaling	1.00	0.92	0.61	0.50
↳ Frequency Scaling	1.00	1.00	0.98	0.95

Code: Fortran, OpenMP

Platform: MareNostrum-IV

Dual Intel Xeon Platinum 8160 Skylake 48-core nodes

Scale: 1-45 threads

Tools: Extrae & Paraver, Vtune, MAQAO

Poor scalability of the code is due to multiple factors:

- OpenMP Region Efficiency and reducing IPC are major limiting factors,
- Resulting in, respectively, poor Parallel Efficiency and poor Computational scaling

nag

Example 3

Improving the Performance

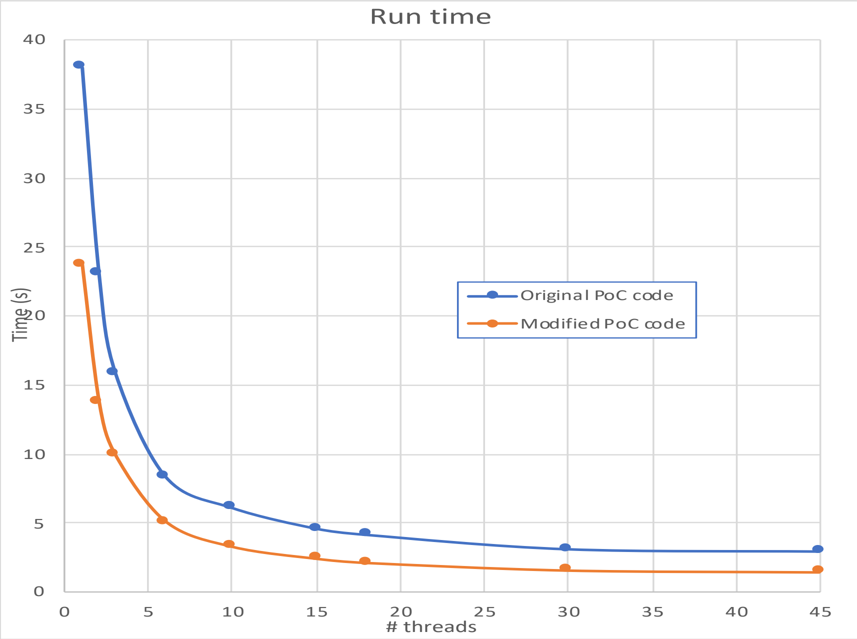
	Original code for *Proof of Concept*						Modified code					
# threads	1	2	10	18	30	45	1	2	10	18	30	45
Global Efficiency	1.00	0.86	0.65	0.41	0.31	0.15	1.00	0.86	0.72	0.62	0.51	0.37
↳ Parallel Efficiency	1.00	0.97	0.80	0.69	0.62	0.59	1.00	0.97	0.90	0.83	0.78	0.75
↳ OpenMP Region Efficiency	1.00	0.97	0.81	0.69	0.63	0.60	1.00	0.97	0.91	0.85	0.80	0.78
↳ Serial Region Efficiency	1.00	1.00	0.99	0.99	0.99	0.99	1.00	1.00	0.99	0.98	0.98	0.98
↳ Computational Scaling	1.00	0.89	0.81	0.60	0.49	0.26	1.00	0.88	0.81	0.75	0.65	0.49
↳ Instruction Scaling	1.00	1.00	1.00	1.00	0.99	0.97	1.00	1.00	1.00	0.99	0.99	0.98
↳ IPC Scaling	1.00	0.87	0.80	0.60	0.51	0.36	1.00	0.89	0.82	0.77	0.67	0.56
↳ Frequency Scaling	1.00	1.02	1.02	1.00	0.97	0.74	1.00	1.00	0.98	0.98	0.98	0.89

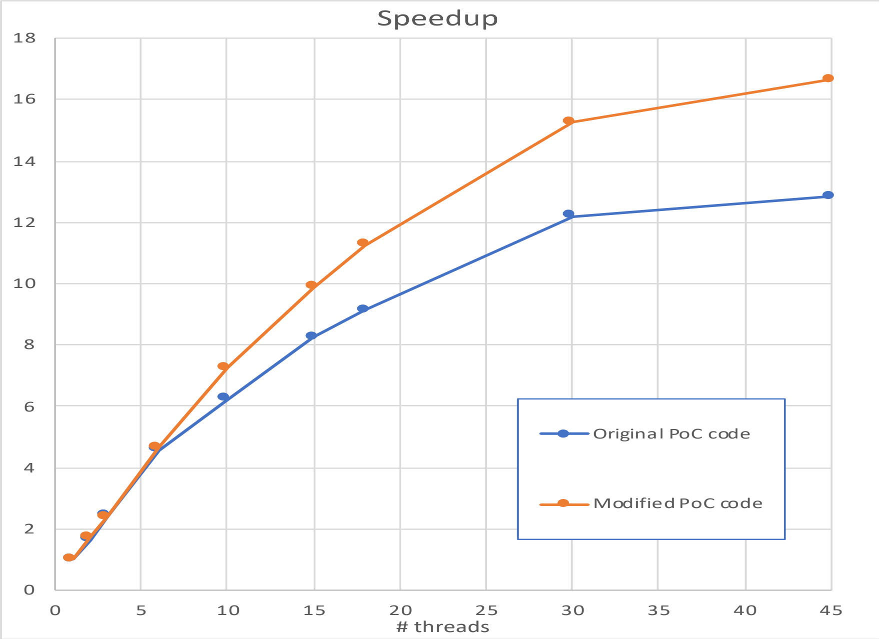
Code refactoring by the POP Proof of Concept service

- Use of OpenMP COLLAPSE clause to improve load balance
- Move some calculations outside the loops & remove unnecessary calculations
- Use optimal loop ordering with nested loops

Example 3

Performance of modified code





The modified code

- is 1.6x faster on 1 thread due to reduced instruction count
- is 2.1x faster than original on 45 threads
- shows better parallel scaling with a speedup of 16.7 on 45 threads relative to 1 thread

Success Stories

13 JUL

POP Collaboration with PerMedCoE achieves a 1.45x Speedup in PhysiCell, one of PerMedCoE Core Applications

A collaboration between POP and PerMedCoE started with the performance assessment of PhysiCell.

READ MORE

25 MAY

Run time halved for OpenMP code

Having already identified the three causes of low efficiency in

READ MORE

17 MAY

A one-day POP online training for SURF

Jonathan Boyle and Federico Panichi from POP partner NAG (Numerical Algorithms Group) recently pr

READ MORE

21 APR

Diversifying the HPC community: boosting the uptake of advanced HPC training by women and underrepresented groups

HPC training is a crucial step in encouraging and building a diverse and inclusive workforce for

READ MORE

23 FEB

POP for Astronomy - 40% Reduction in Execution Time for the PIERNIK Code

PIERNIK is a parallel astrophysical fluid simulati

READ MORE

02 FEB

Performance Improvements by More Than 30% and a Data Race Fixed for CalculiX Code

CalculiX is an open source computational fluid dynamics code.

READ MORE

05 NOV

588x and 488x Execution Time Speedups of a Volcanic Hazard Assessment Code

The Probabilistic Volcanic Hazard Assessment Work Flow package (PVHA_WF) is a workflow created fo

READ MORE

success stories

More than 350 services since 2015 across all domains, e.g. engineering, earth & atmospheric sciences, physics, biology and genetics

https://pop-coe.eu/blog/tags/success-stories

Online Content

www.pop-coe.eu

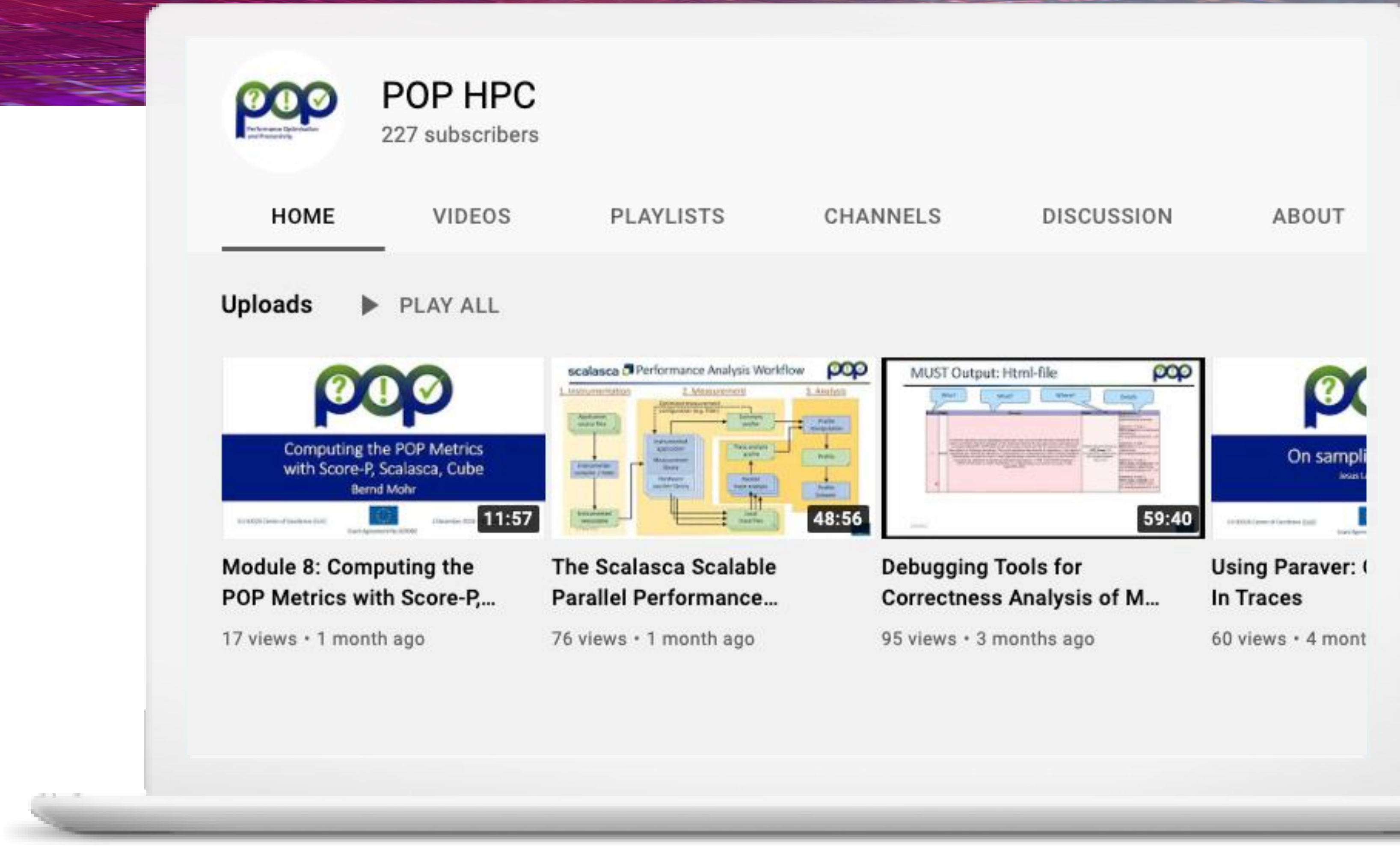
POP Website

https://pop-coe.eu/services

All the information you need to access POP services
Blogs, More Learning Materials, Newsletter,
subscribe and see past issues

https://www.youtube.com/pophpc

Past Webinars | POPCasts

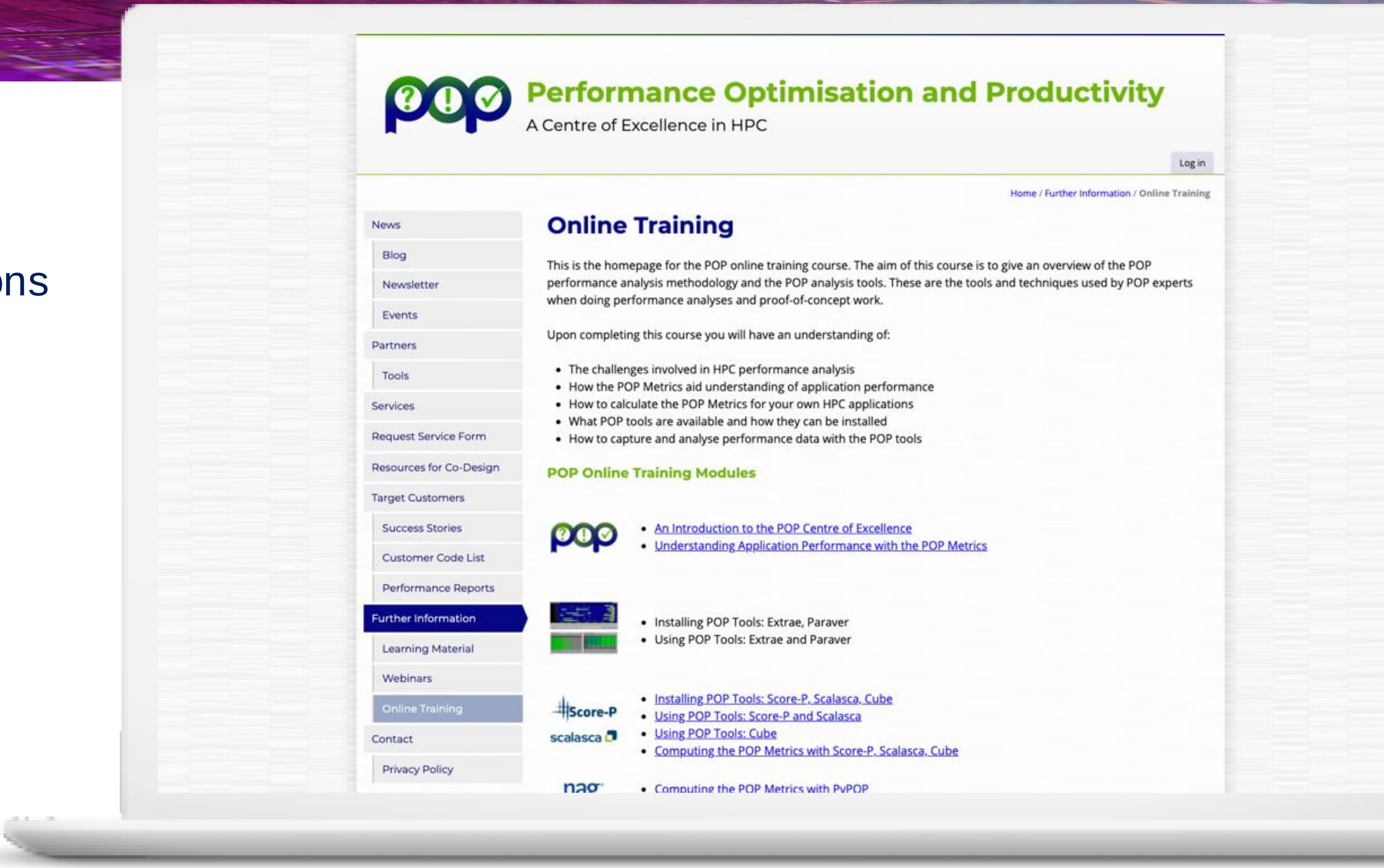


POP Online training

A series of self-study modules
For those with limited experience in performance analysis of HPC applications

Learning Objectives

- The challenges involved in HPC performance analysis
- How the POP Metrics aid understanding of application performance
- How to calculate the POP Metrics for your own HPC applications
- What POP tools are available and how they can be installed
- How to capture and analyse performance data with the POP tools



Summary

POP Performance Metrics

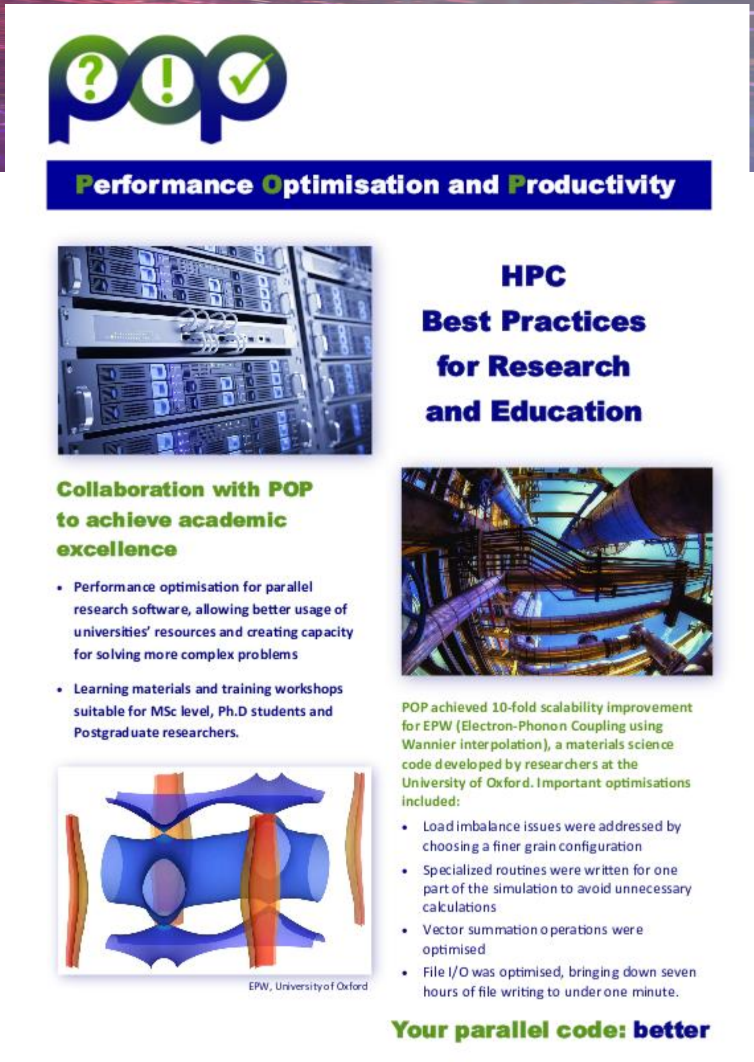
- Build a quantitative picture of application behavior
- Allow quick diagnosis of performance problems in parallel codes
- Identify strategic directions for code refactoring
- So far metrics for MPI, OpenMP and Hybrid (OpenMP + MPI) codes

POP works

- Across application domains, platforms, scales
- With (EU/UK) academic and industrial customers including code developers, code users, HPC service providers and vendors
- To apply for a POP service go to https://pop-coe.eu/services

POP CoE

- Promotes best practices in parallel programming
- Encourages a systematic approach to performance optimization
- Facilitates and invests in training HPC experts



A Centre of Excellence in HP

Performance Optimisation and Productivity

Contact:

https://www.pop-coe.eu

pop@bsc.es

@POP_HPC

youtube.com/POPHPC