Parallel Performance Analysis and Tuning as a Service
• 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
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](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
- Primary service
- Identify performance issues of customer code (at customer site)
- Small effort (< 1 month)

Parallel Application Performance Plan
- 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
- 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
• The following metrics are used in a POP Performance Audit:

• Global Efficiency (GE): \( GE = PE \times \text{CompE} \)
  
  • Parallel Efficiency (PE): \( PE = LB \times \text{CommE} \)
    
    • Load Balance Efficiency (LB): \( LB = \frac{\text{avg}(CT)}{\text{max}(CT)} \)
    
    • Communication Efficiency (CommE): \( \text{CommE} = \text{SerE} \times \text{TE} \)
      
      • Serialization Efficiency (SerE): \( \text{SerE} = \max \left( \frac{CT}{TT \text{ on ideal network}} \right) \)
      
      • Transfer Efficiency (TE): \( \text{TE} = \frac{TT \text{ on ideal network}}{TT} \)

• 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

CT = Computational time
TT = Total time
## POP Users and Their Codes

<table>
<thead>
<tr>
<th>Area</th>
<th>Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational Fluid Dynamics</td>
<td>DROPS (RWTH Aachen), Nek5000 (PDC KTH), SOWFA (CENER), ParFlow (FZ-Juelich), FDS (COAC) &amp; others</td>
</tr>
<tr>
<td>Electronic Structure Calculations</td>
<td>ADF (SCM), Quantum Expresso (Cineca), FHI-AIMS (University of Barcelona), SIESTA (BSC), ONETEP (University of Warwick)</td>
</tr>
<tr>
<td>Earth Sciences</td>
<td>NEMO (BULL), UKCA (University of Cambridge), SHEMAT-Suite (RWTH Aachen) &amp; others</td>
</tr>
<tr>
<td>Finite Element Analysis</td>
<td>Ateles (University of Siegen) &amp; others</td>
</tr>
<tr>
<td>Gyrokinetic Plasma Turbulence</td>
<td>GYSELA (CEA), GS2 (STFC)</td>
</tr>
<tr>
<td>Materials Modelling</td>
<td>VAMPIRE (University of York), GraGLEs2D (RWTH Aachen), DPM (University of Luxembourg), QUIP (University of Warwick) &amp; others</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>OpenNN (Artelnics)</td>
</tr>
</tbody>
</table>
Customer Feedback (Sep 2016)

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

- Extremely responsive

- Very responsive

- Moderately responsive

- Slightly responsive

- Not at all responsive

0% 10% 20% 30% 40% 50% 60% 70% 80%

- Excellent

- Good

- Not so good

- Bad

0% 10% 20% 30% 40% 50% 60% 70% 80%

• 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
  • Commercial tools (if available)

• ... 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
Proof-of-Concept Examples
• 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
• 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
Sustainability

• H2020 CoE’s are supposed to sustain themselves after some point
  • Proposals had to include a business plan

• Current plan: 3 sustainable operation modes
  • Pay-per-service
  • Service subscriptions
  • Continue as non-profit organisation (broker for free + payed services)

• Requires to have more industrial rather than academic/research customers
  • Experience so far
    • Typically require NDA ⇒ delays services by months
    • No access to code/computers ⇒ guide (inexperienced) customer to install tools + measure ⇒ delays services by months
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.