Future Generation Computer Systems 162 (2025) 107472

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

15+ years of joint parallel application performance analysis/tools training with *Scalasca/Score-P* and *Paraver/Extrae* toolsets

Brian J.N. Wylie [a,*], Judit Giménez [b,c], Christian Feld [a], Markus Geimer [a], Germán Llort [b], Sandra Mendez [b], Estanislao Mercadal [b], Anke Visser [a], Marta García-Gasulla [b]

[a] *Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, 52425, Jülich, NRW, Germany*
[b] *Barcelona Supercomputing Center, Plaça Eusebi Güell, 1-3, Barcelona, 08034, Catalunya, Spain*
[c] *Universitat Politècnica de Catalunya – BarcelonaTech, Carrer de Jordi Girona, 31, Barcelona, 08034, Catalunya, Spain*

ARTICLE INFO

Keywords:
Hybrid parallel programming
MPI message-passing
OpenMP multithreading
OpenACC device offload acceleration
HPC application execution performance measurement & analysis
Performance assessment & optimisation methodology & tools
Hands-on training & coaching

ABSTRACT

The diverse landscape of distributed heterogeneous computer systems currently available and being created to address computational challenges with the highest performance requirements presents daunting complexity for application developers. They must effectively decompose and distribute their application functionality and data, efficiently orchestrating the associated communication and synchronisation, on multi/manycore CPU processors with multiple attached acceleration devices structured within compute nodes with interconnection networks of various topologies.

Sophisticated compilers, runtime systems and libraries are (loosely) matched with debugging, performance measurement and analysis tools, with proprietary versions by integrators/vendors provided exclusively for their systems complemented by portable (primarily) open-source equivalents developed and supported by the international research community over many years. The *Scalasca* and *Paraver* toolsets are two widely employed examples of the latter, installed on personal notebook computers through to the largest leadership HPC systems. Over more than fifteen years their developers have worked closely together in numerous collaborative projects culminating in the creation of a universal parallel performance assessment and optimisation methodology focused on application execution efficiency and scalability, and the associated training and coaching of application developers (often in teams) in its productive use, reviewed in this article with lessons learnt therefrom.

Contents

* Corresponding author.
E-mail addresses: b.wylie@fz-juelich.de (B.J.N. Wylie), judit@bsc.es (J. Giménez).

https://doi.org/10.1016/j.future.2024.07.050
Received 8 January 2024; Received in revised form 10 July 2024; Accepted 29 July 2024
Available online 2 August 2024

1. Introduction

Current operational and anticipated high-performance computing (HPC) computer systems are particularly large and complex, resulting in significant challenges both to write correctly functioning and performant code that scales and executes efficiently. Parallelism has long been required for both shared and distributed memory of increasingly heterogeneous hardware comprising multicore/manycore processors (CPUs) and attached accelerator devices (typically GPUs). Distinct parallel architectures have motivated dialects and extensions to common programming languages or entirely new ones.

At the University of Edinburgh, one author [BJNW] was first exposed in their 1986 final-year B.Sc. physics project to the massively-parallel ICL Distributed Array Processor programmed in `DAP-Fortran`, motivated their subsequent Ph.D. harnessing the Edinburgh Concurrent Supercomputer Meiko Computing Surface and CS-2 transputers[1] with `occam` for computational fluid dynamics simulations, and lead to an application scientist position in the nascent Edinburgh Parallel Computing Centre (EPCC) developing the award-winning *PARAMICS* massively-parallel 'microscopic' traffic simulator [1] in `C*` for the Thinking Machines Connection Machine CM-200.

From this hotbed of parallel computing innovation (which continues unabated), it was early recognised the need for comprehensive training to aide application developers effectively exploit these regularly evolving computer systems. With Stephen Booth, Lyndon Clarke, Neil MacDonald, Mike Norman, Nick Radcliffe, Greg Wilson[2] and many others, highly effective training centred on incremental constructive hands-on parallel programming and software engineering exercises was developed, and via Train-the-Trainer Training (TTT) a multitude of instructors trained to deliver the training in many courses, tutorials and workshops. Subsequent career positions developing parallel application performance analysis tools in partnership with NEC Corp. and with Sun Microsystems Inc. further emphasised the concurrent need for effective training and coaching in the use of those powerful and sophisticated tools.

After joining the R&D groups of Bernd Mohr and Felix Wolf supporting application developers with portable parallel performance analysis tools at Forschungszentrum Jülich in 2004, and more specifically the Virtual Institute-High-Productivity Supercomputing (VI-HPS), member discussion quickly motivated establishing a series of workshops[3] dedicated to hands-on parallel performance analysis and tuning with VI-HPS member tools. The *Scalasca* toolset from JSC (originally before integration of *Score-P*) and *Paraver* toolset from BSC were featured in the first VI-HPS Tuning Workshop hosted by RWTH Aachen and almost all of the subsequent forty-plus (in-person and virtual) workshops. A wide variety of bespoke training workshops (for HPC Centres of Excellence [3] and other organisations) and hands-on tutorials at conferences have also paired the toolsets.

Each toolset has its own strengths and limitations, suiting it for particular uses and users according to their needs and experience. Opportunities to learn about and try out each, with actual application codes on a specific computer system, are therefore especially valuable. As well as the application developers finding the tools that best suit them, such joint training events provide insight to the tools developers as to the opportunities and priorities for improvements.

Tools developed by hardware and system vendors may access internal and undocumented aspects of their computer systems that are not readily available to open-source tools. Such information can be extremely helpful for platform-specific insight and optimisations, while complementing multi-platform tools that facilitate performance comparisons between systems and ease user migration from one to another. Additionally, Linux interfaces are exploited by a large set of tools for portable performance analysis on HPC, cloud-based and other computer systems down to individual workstation and notebook computers. Training covering this spectrum of available tools is therefore essential.

Application developers targeting extreme-scale HPC systems — such as *JUQUEEN*, *Kei* computer and *Fugaku*, heterogeneous systems such as *MareNostrum5*, and modular supercomputing architectures such as *JUWELS Cluster+Booster* and the first ExaFLOPs supercomputer in Europe *JUPITER*[4] — need effective tools to assist with porting and tuning for these unusual systems.

The *XcalableMP* compilation system (and directive-based language) [4–7], *Paraver/Extrae/Dimemas* performance analysis tools [8,9] and *Scalasca/Score-P/CUBE* execution measurement and analysis tools [10,11] are notable examples of tools developed by RIKEN R-CCS, BSC and JSC for this purpose. The performance tools utilise the PAPI library from UTK to acquire metrics from hardware and software counters [12].

Within the Joint Laboratory for Extreme-Scale Computing (JLESC) a dedicated project extends support of these tools for members' HPC systems and exploits their capabilities in an integrated workflow for porting and tuning of parallel applications on extreme-scale parallel systems. Existing training material has been adapted to collaborators' large-scale HPC systems, augmented with newly prepared material, and refined for better uptake based on participant evaluations and feedback. Travel and accommodation expenses of training presenters to participate in joint training events (such as VI-HPS Tuning Workshops outside of Europe in Japan and USA) was supported. Collaborative work with application developers assessed the effectiveness of the tools, and helped direct development of new tool capabilities.

This article summarises more than fifteen years of joint parallel performance analysis/tools training with our tools, discussing lessons learnt therefrom.

[1] Contemporaneously in 1991 the first introduction to parallel computing for JG at Universitat Politècnica de Catalunya & Parsys España SA. https://en.wikipedia.org/wiki/Transputer

[2] Wilson later established The Carpentries global community teaching foundational data and coding skills through instructional workshops. [2] https://carpentries.org/.

[3] Now more commonly referred to as hackathons, testathons or tunathons.

[4] Joint Undertaking Pioneer for Innovative and Transformative Exascale Research: https://jupiter.fz-juelich.de.

Table 1
Institution project membership, HPC systems and tools.

Institution	JLESC	POP CoE	VI-HPS	(Prior)/Current/[Future] HPC Systems	Application tools
ANL	+	–	–	*(Mira)/Polaris – Theta/[Aurora]*	Darshan
BSC	+	+	+	*(MareNostrum4 – CTE-Power)/MareNostrum5*	Paraver Extrae Dimemas
FAU-RRZE	–	–	+	*Alex – Fritz – Meggie – Woody*	LIKWID
FZJ-JSC	+	+	+	*(JUQUEEN)/JURECA – JUWELS/[JUPITER]*	Scalasca/CUBE Score-P
HLRS	–	+	+	*(HazelHen)/Hawk – Kabuki – Vulcan/[Hunter]*	Memchecker
INRIA	+	–	–	*Grid'5000*	SIMGRID
IT4I	–	+	[a]	*Barbora – Karolina*	MERIC
INESC-ID	–	+	[a]	*–/[Deucalion]*	CARM IntelRoofline
Linaro	–	–	+	internal systems only	Forge DDT/MAP/PR
LLNL	–	–	+	*(Sequoia)/Sierra/[ElCapitan]*	Caliper PnMPI STAT
LRZ	–	–	+	*(SuperMUC)/SNG – BEAST – CoolMUC/[SNG-2]*	Cachegrind
R-CCS	+	–	–	*(Kei)/Fugaku*	XcalableMP
RWTH	–	+	+	*(CLAIX-2018)/CLAIX-2023*	MUST Archer
TERATEC	–	+	–	*Curie – Joliot*	
TUD-LPP	–	–	+	*Lichtenberg*	DiscoPoP Extra-P
TUD-ZIH	–	–	+	*(Taurus)/Barnard*	Score-P Vampir
TUM	–	–	+	as LRZ	mpiP GPUscout
UIUC-NCSA	+	–	–	*(BlueWaters)/Delta/[Delta-AI]*	
UO-PRL	–	–	+	*Franken*	TAU PDT ParaProf PerfExplorer
UTK-ICL	+	–	+	*(Kraken)/–*	PAPI
UVSQ	–	+	+	as TERATEC	MAQAO

[a] membership pending.

2. Partner organisations & projects

Barcelona and Jülich Supercomputing Centres have partnered for many years in a wide variety of collaborations on parallel application performance analysis tools, where joint training played a central role. Some of these collaborations are summarised in Table 1, including each institution's HPC systems and locally-developed application tools, and detailed in this section.

2.1. VI-HPS

The Virtual Institute-High Productivity Supercomputing (VI-HPS) was established to improve the quality and accelerate the development process of complex simulation codes in science and engineering that are designed for and running on the most advanced highly-parallel computer systems. Activities comprise development and integration of HPC application development tools, primarily those for performance analysis and correctness checking of parallel applications, training with those tools via tutorials and workshops/hackathons, plus organising symposia with a technical program related to tools usually embedded in an HPC-related conference program.

VI-HPS tools development efforts place particular emphasis on scalability and ease of use, with training and support an essential component of the mission to inform application developers of the benefits they can obtain by using them. While platform-specific tools are not excluded from the VI-HPS portfolio, tools which support multiple platforms of diverse processors and architectures are favoured. In this regard, open-source and free to use tools are also preferred, while tools with commercial licenses are required to support multiple platforms for them to be included.

Complementing the other VI-HPS activities, ProTools workshops on *Programming and Performance Visualisation Tools* have been held annually in conjunction with the SC (Supercomputing) conference series since 2019. Prior to this, ESPT workshops on *Extreme-Scale Programming Tools* were held in conjunction with SC conferences between 2012 and 2018, and PROPER workshops on *Productivity and Performance Tools* held in conjunction with the Euro-Par conference series between 2008 and 2014.

Start-up funding from the Helmholtz Association of German research centres in 2006 supported the founding partners of VI-HPS: Jülich Supercomputing Centre of Forschungzentrum Jülich, Centre for Computing & Communication (later IT Center) of RWTH Aachen University, Centre for Information Services & HPC of Dresden University of Technology (Technische Universität Dresden) and associate partner Innovative Computing Laboratory of the University of Tennessee (Knoxville).

Since the expiry of Helmholtz seed funding in 2011, VI-HPS has added another ten partners from around Europe and USA: Friedrich-Alexander-Universität (FAU) Erlangen-Nürnberg, Leibniz Supercomputing Centre (LRZ), Technical University of Darmstadt, Technical University of Munich (TUM) and University of Stuttgart (HLRS) in Germany, Barcelona Supercomputing Center (BSC), Lawrence Livermore National Laboratory (LLNL), University of Oregon, Université de Versailles St.-Quentin-en-Yvelines (UVSQ), and finally the Allinea software team now part of Linaro Ltd.

A *VI-HPS Tools Guide* booklet [13] offers a brief overview of the respective tools contributed by the project partners, showcasing their individual debugging, correctness checking and performance measurement and analysis capabilities, and indicating their support for parallel computer systems, programming models and languages. Detailed information about each particular tool can be found on the websites listed therein and on the VI-HPS website itself.[5]

2.2. POP CoE

The Performance Optimisation and Productivity Centre of Excellence (POP CoE) for HPC Applications has been funded since 2015 by the European Union Horizon program and from 2024 by the EuroHPC Joint Undertaking (JU).[6] It gathers leading European experts in parallel performance tools/analysis and programming models to offer services to the academic and industrial communities to help them better understand the execution behaviour of their applications, suggest the most productive directions for optimising the performance of the codes and help implementing those transformations in the most productive way.

Partners are Barcelona Supercomputing Center (BSC), University of Stuttgart High Performance Computing Center (HLRS), Jülich Supercomputing Centre of Forschungzentrum Jülich, RWTH Aachen University IT Center, TERATEC association (France), Université de Versailles St.-Quentin-en-Yvelines (UVSQ, France) and VŠB-Technical University of Ostrava IT4Innovations National Supercomputing Center (IT4I, Czechia), joined in 2024 by Instituto de Engenharia de Sistemas e Computadores: Investigação e Desenvolvimento em Lisboa (INESC-ID, Portugal). Numerical Algorithms Group Ltd. (NAG, UK) were also

[5] https://www.vi-hps.org/
[6] https://www.pop-coe.eu/

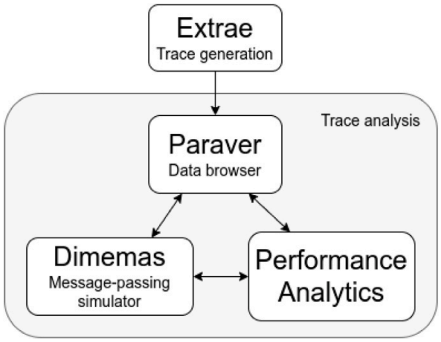


Fig. 1. BSC-Tools ecosystem.

a partner in POP CoE until 2022. With most partners contributing performance and correctness tools to POP services and developing training for the tools and an associated parallel execution efficiency and scalability methodology, synergies with VI-HPS are exploited. (Both INESC-ID and IT4I are expected to join VI-HPS in 2024.)

2.3. JLESC project

With training in performance tools at the core of VI-HPS and POP CoE, and motivated to outreach to major HPC institutions in Japan and USA, it was beneficial to establish the project *Developer tools for porting and tuning parallel applications on extreme-scale parallel systems* in 2014, between Jülich Supercomputing Centre of Forschungszentrum Jülich, Barcelona Supercomputing Center (BSC), RIKEN Center for Computational Science (R-CCS) and research partner the University of Tennessee Knoxville (UTK), in conjunction with Institut national de recherche en informatique et en automatique (INRIA, France), National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign (UIUC), and Argonne National Laboratory (ANL) within the Joint Laboratory for Extreme-Scale Computing (JLESC).[7]

3. Tools

We first introduce the toolsets developed by Barcelona Supercomputing Center and Jülich Supercomputing Centre that have collaborated in our JLESC project and related activities.

3.1. BSC: Paraver/Extrae toolset

The *Paraver* toolset was created in 1991 under the research of UPC (Universitat Politècnica de Catalunya–BarcelonaTech). Since its establishment in 2005, the Barcelona Supercomputing Center (BSC-CNS) has been responsible for the performance tools research and development [9]. The suite of tools is designed to streamline the performance analysis of parallel applications, and is available as open source under the GNU LGPL v2.1 license. This toolset encompasses the generation of detailed execution traces, a data visualisation browser for qualitative analysis and quantitative calculations, a message passing simulator, and what we term performance analytics–data analytics techniques applied to performance data. These tools work together seamlessly in an integrated ecosystem, as shown in Fig. 1, creating a powerful synergy that helps developers and analysts optimise application performance, identify scalability bottlenecks, and make informed decisions to enhance overall efficiency in high-performance computing environments.

3.1.1. Extrae

A typical analysis workflow starts from the Extrae tracing framework [14], which uses several mechanisms to transparently capture performance data, ranging from dynamic instrumentation of unmodified production binaries to static linking. Yet, one of the most straightforward and preferred methods is the `LD_PRELOAD` interposition to intercept function calls of production binaries at loading time.

The traces collected by Extrae include entry and exit to the programming model runtime, hardware counters through the PAPI library,[8] callstack references, periodic samples and some system calls. Users have the flexibility to add program routines' invocations (not included by default) and instantaneous events. Focusing on the activity of the parallel runtime guarantees minimal overhead in most scenarios, given that the application's use of the parallel runtime should be not excessively fine-grained.

Extrae supports the main parallel programming models, namely MPI, OpenMP, CUDA, OpenCL, pthreads and OmpSs, and new interfaces such as GASPI, OpenACC and ROCm. It supports programs written in C, Fortran, Java and Python, as well as combinations of different languages, hybrid and modular codes. It is available for most UNIX-based operating systems and has been deployed in all relevant HPC architectures and platforms, including x86-64, ARM, ARM64, POWER, RISC-V, SPARC64, BlueGene, Cray, and HPC accelerators.

XMP extension. In the JLESC project, Extrae was extended to instrument the XcalableMP runtime [5], a directive-based language extension which allows users to develop parallel programs for distributed memory systems easily and to tune the performance by having minimal and simple notations. The instrumentation was achieved using XMPT, the generic tool API of XMP, based on runtime callbacks and using the Extrae API to emit events from an independent library, serving as a proof-of-concept for an external plugin within the tool.

In addition to capturing entry and exit points in the XMP runtime, the tool can also track the underlying MPI activity. This allows to present views correlating XMP functions with the MPI layer, as illustrated in Fig. 3 for a detailed iteration of the main computing loop in an 8-process execution of the Impact3D benchmark. This depiction shows XMP calls over time at the top and MPI calls with communication lines at the bottom.

Furthermore, this capability allows for the precise identification of overhead introduced by individual layers within the parallel runtime, including both the XMP and MPI. Fig. 2 displays the time percentage ratio of internal MPI activity in XMP calls. The left table reveals that nearly all the time spent inside `xmp_barrier` is attributed to MPI_Barrier. However, within `coarray_write` and `coarray_read`, XMP introduces an overhead exceeding 50% on top of the MPI layer.

Overall, the availability of XMP performance data provides valuable feedback not only to application developers regarding their code performance, but also to XMP runtime developers concerning the efficiency of their implementation.

3.1.2. Paraver

Paraver is a highly flexible data browser [8,15]. It enables the analyst to create two main types of views: timelines and tables, allowing the display of a vast number of metrics with the available data. The timelines depict the activity over time (x-axis) of each process/thread (y-axis) in the execution. The tables, whether in the form of profiles or histograms, aggregate statistics over any selected region. These views can display categorical information, such as the calls made to the parallel runtime, illustrated by Figs. 4(a) and 4(b) for the MPI calls, and continuous metrics, such as the duration of the computing phases, illustrated by Fig. 5(a), and the histogram in Fig. 5(b).

Users can customise metrics in Paraver through a filter module, time functions, and mechanisms for combining timelines and correlating different metrics. The expert's knowledge can be captured by saving any

[7] https://jlesc.github.io/

[8] https://icl.utk.edu/papi

	Outside MPI	MPI_Barrier
Average	7.92111 %	92.07889 %
Avg/Max	0.92510	0.99221

(a) `xmp_barrier`

	Outside MPI	MPI_Put
Average	54.37011 %	45.62989 %
Avg/Max	0.98963	0.99161

(b) `coarray_write`

	Outside MPI	MPI_Get
Average	60.83533 %	39.16467 %
Avg/Max	0.98558	0.97749

(c) `coarray_read`

Fig. 2. XMP overhead on top of the MPI layer in the `FFB_mini` benchmark.

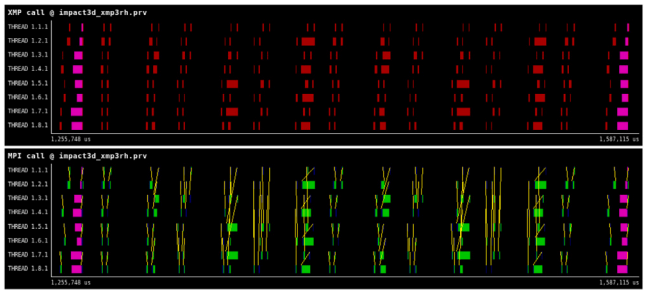
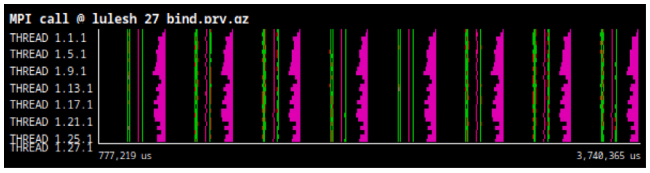


Fig. 3. XMP calls (top) correlated with underlying MPI calls (bottom) in 8-process execution timelines of the `Impact3D` benchmark.

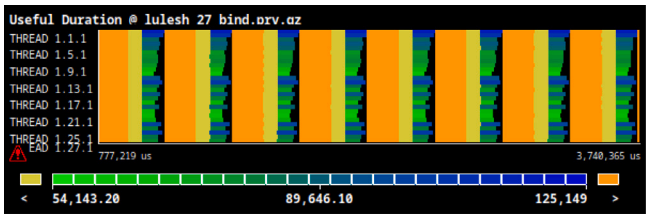


(a) MPI calls over time for the iterative phase

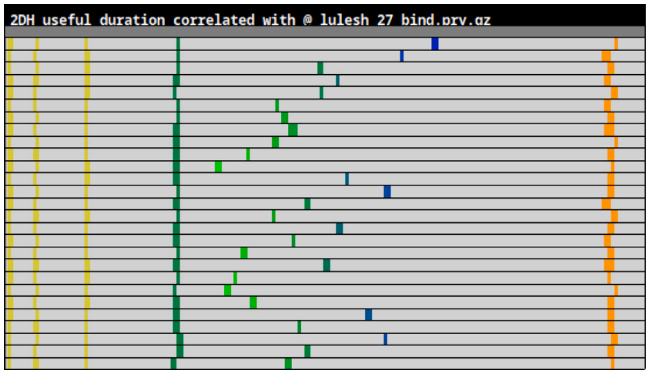
	Outside MPI	MPI_Allreduce	MPI_Waitall	MPI_Wait	MPI_Isend	MPI_Irecv
Total	2,415.17 %	259.18 %	17.66 %	3.53 %	2.43 %	2.02 %
Average	89.45 %	9.60 %	0.65 %	0.13 %	0.09 %	0.07 %
Maximum	99.50 %	16.17 %	1.14 %	0.42 %	0.18 %	0.16 %
Minimum	83.27 %	0.01 %	0.20 %	0.05 %	0.04 %	0.04 %
StDev	4.12 %	4.07 %	0.28 %	0.08 %	0.03 %	0.03 %
Avg/Max	0.90	0.59	0.57	0.31	0.50	0.47

(b) Percentage time profile per MPI call

Fig. 4. MPI views for the `LULESH` benchmark.



(a) Timeline depicting computations duration with gradient focusing on the unbalanced region



(b) Histogram correlating the duration of computation regions (from lower to higher duration, left to right) with the corresponding number of instructions (from lower to higher instructions, indicated by a gradient from green to blue colors)

Fig. 5. Useful computation views for the `LULESH` benchmark.

view or set of views as a Paraver configuration file, facilitating analysis repetition, multi-experiment comparisons, or applying the same views to different applications using timestamped traces, which are openly accessible and easy to generate [16].

Using the previous Paraver views to illustrate how the tool assists in gaining insight, and starting with Fig. 4(a), a repetitive pattern of MPI calls throughout several iterations of the main computing loop in `LULESH` benchmark [17] execution with 27 MPI processes can be observed. The profile in Fig. 4(b) indicates that the program is engaged in computations almost 90% of the time, with nearly all of the remaining 10% consumed by MPI_Allreduce, also revealing poor *Load Balance* (Avg/Max) of 0.59 during this synchronisation phase. To understand the cause of this imbalance, it is necessary to examine the computing phase preceding the synchronisation. Fig. 5(a) shows the computing phases of the program, with the gradient from light green to dark blue representing the length of the computations, from shorter to longer. Centred on the computation before MPI_Allreduce, this region is notably unbalanced, with different processes exhibiting varied colours, indicating differing durations. Fig. 5(b) displays a histogram correlating the duration of the unbalanced computing phase with the instructions count. The scattered area in the centre of the histogram corresponds to the unbalanced region, and as points move to the right, the colour shifts from green to blue, indicating that longer computations are associated with a higher number of instructions. This indicates that the imbalance in this area is due to work imbalance.

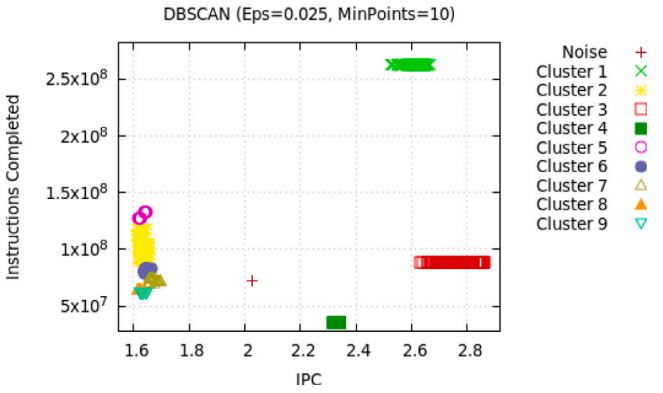
This brief example exemplifies how the tool enables users to navigate through the data, formulate hypotheses, and validate them both visually and quantitatively, empowering users to understand in detail the behaviour of their application.

3.1.3. Performance analytics

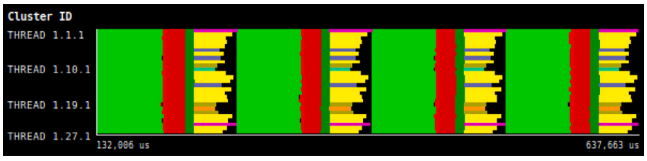
Traces provide a detailed capture of the application's execution that inexperienced users may find difficult to analyse. For this reason, BSC initiated the development of performance analytics tools in 2007 to automatically extract valuable insights from the data. In this context, *BurstClustering* employs density-based clustering analysis [18] to automatically classify computations using a selection of the metrics measured (typically hardware counters). This approach allows to characterise computing phases of the program with distinct performance behaviour through a few clusters, revealing trends and exposing underlying structure.

As clustering is an unsupervised algorithm, correlating the clusters from different executions may not be easy. *Tracking* [19] leverages movement tracking methods to study the evolution of clusters across multiple experiments with different settings within the performance space. This method provides valuable insights into how different setups influence the program's performance over time.

For detailed performance evolution, *Folding* [20] combines instrumentation and coarse-grain sampling. This approach allows to describe very small regions with a high level of detail and minimal overhead, and then break down the performance of these regions using top-down models based on hardware counters.



(a) Computations classified w.r.t. Instructions Completed and IPC



(b) Distribution of identified clusters over time

Fig. 6. Clustering views for the LULESH benchmark.

These techniques can be applied postmortem but also at runtime. To this end, Extrae was extended with on-line mechanisms [21] to orchestrate the use of analytics to expose structure, identify patterns, select representatives of common behaviour, and overall, intelligently direct the gathering of data to maximise the amount of useful information collected while minimising the size of the traces.

During training courses, where time constraints are a consideration, clustering analysis is predominantly utilised due to the tool's ease of use and the quick insight it provides. Fig. 6(a) illustrates the results of *BurstClustering* applied to a trace of the LULESH benchmark. The scatter plot correlates the number of instructions executed on the *y*-axis with instructions executed per cycle (IPC) on the *x*-axis. Clusters represent the aggregation of all instances of computing phases in the program that share a similar performance behaviour in terms of these two metrics.

The plot highlights four main behavioural trends ordered by their importance in the execution. Cluster 1 (green) executed the highest number of instructions (topmost), and elongates horizontally, indicating variability in IPC. Cluster 2 (yellow) is the slowest (leftmost) and elongates vertically, indicating variability in the number of instructions executed. It also splits into multiple clusters at the top and the bottom, highlighting significantly distinct behaviour and outliers. Cluster 3 (red) executes with the highest but variable IPC (rightmost), as indicated by the horizontally elongated shape. Finally, Cluster 4 (dark green) executes the lowest number of instructions (bottommost). Correlating with Fig. 6(b), which displays the distribution of clusters over time, we can observe all processes executing computations belonging to the same cluster, following an SPMD paradigm. In the yellow phase, all the outliers observed in the scatterplot occur concurrently, indicating an imbalance between MPI ranks in this region.

A brief characterisation like this is useful because it quickly reveals structure in the data and provides a meaningful description of the program's main computing phases' behaviour. Moreover, it facilitates the identification of imbalances by showing clusters with elongated shapes, and suggests areas of the program that may be interesting to improve, either because they execute a large number of instructions or operate at low IPC.

3.1.4. Dimemas

Dimemas [22] is a coarse-grain simulator designed for MPI programs. It reconstructs the time behaviour of a parallel application on a machine modelled by a set of performance parameters, offering accurate predictions of the performance the application would achieve on the target machine.

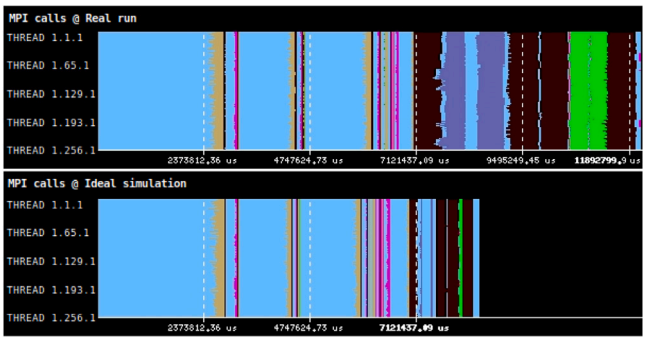


Fig. 7. Network impact analysis for the GADGET application.

The simulator generates trace files that can be analysed with Paraver, allowing users to conveniently examine performance issues identified during a simulation run. The supported target architecture comprises a cluster of nodes, potentially heterogeneous and equipped with CPUs and optionally accelerators. This architecture can be extended to include multiple heterogeneous clusters connected through a network and direct connections.

Dimemas simulates communications using a linear performance model, but also accounts for some non-linear effects like network conflicts. The tool also allows users to specify different task-to-node mappings. Through Dimemas' abstract architectural parameters, users can analyse the importance of various performance factors and assess the possible benefits of specific code optimisations.

The main uses of Dimemas include both parametric studies and what-if analyses. Arguably the most interesting scenario that can be simulated is the *"ideal machine"*, modelling instantaneous communication with zero latency and unlimited bandwidth. This simulation helps identify the portion of MPI time caused by actual data transfer.

Fig. 7 illustrates comparison through Paraver traces of an original run (top) and the ideal scenario (bottom) for the GADGET astrophysics code.[9] In these timelines, light blue represents computation, and other colours correspond to MPI calls. The black region at the end of the ideal simulation corresponds to the time reduction with instantaneous data transfer. We can see that even with an instantaneous network not all the MPI time disappears, and while some MPI calls significantly reduce their time others maintain the same behaviour. Those that reduce their weight in the execution are the ones affected by data transfer and would improve with a faster network. Those that do not reduce their duration highlight waiting time in MPI caused by imbalances or serialisations.

3.1.5. Efficiency model

With the need to classify the time spent in MPI depending on its originating cause, and inspired by the simulation of the *"ideal machine"*, an efficiency model was developed in 2008 [23]. This model is being used in most of BSC's training sessions since 2012 as an easy method to diagnose the sources of inefficiencies.

The core metric of the model is *Parallel Efficiency*, that expresses the percentage of useful time considering the time spent inside the parallel runtimes (e.g., the MPI library) as parallelisation overhead. Using Dimemas' ideal simulation, *Parallel Efficiency* can be split into *Load Balance*, *Serialisation Efficiency* and *Transfer Efficiency*, weighting the factors that generate the time in MPI. The achieved speed-up is not only related to *Parallel Efficiency*, but also depends on the *Computation Scaling*, originally decomposed into *Instructions Efficiency* (amount of work), and *Instructions per Cycle (IPC) Efficiency* (speed).

[9] https://wwwmpa.mpa-garching.mpg.de/gadget/

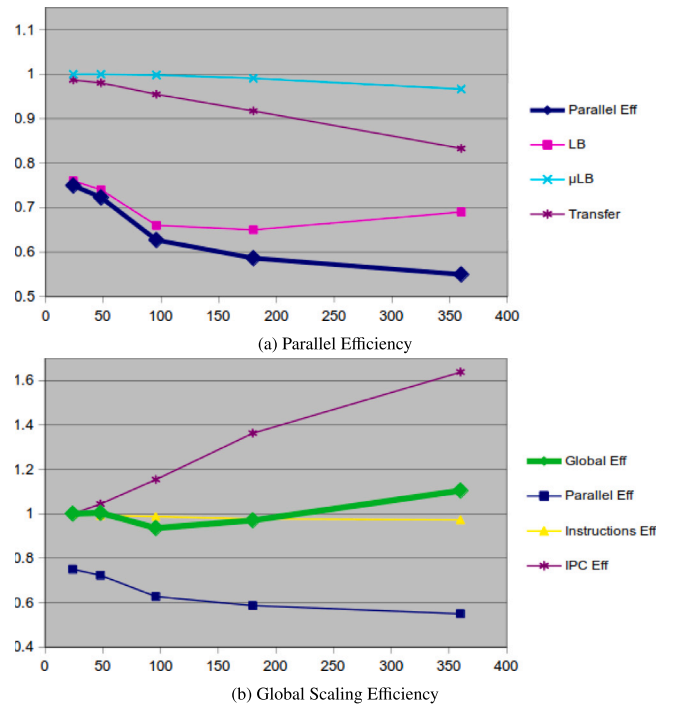


Fig. 8. Efficiency analysis applied to CGPOP benchmark executions with 24 to 360 MPI processes.

Fig. 8 reports the analysis of efficiencies for the CGPOP benchmark [24], which exhibits a scaling in the range from 24 to 360 processes. Fig. 8(a) illustrates the *Parallel Efficiency* for all the scales as well as its components. The first thing to notice is that the *Parallel Efficiency* is low even for the smallest scale (around 0.75 with 24 cores expressing that 25% of the execution time is spent in MPI). Secondly, when scaling up, the efficiency degrades to 0.55 with 360 cores. The main reason for the long time in MPI is poor *Load Balance* (ratio of mean to maximum computation time), but the model also reports that *Transfer Efficiency* (inherent communication time in the absence of waiting) becomes increasingly relevant when scaling up.

The reason why the application seems to report a perfect *Global Scaling Efficiency* with low *Parallel Efficiency* is due to IPC improving with the scale, compensating the *Parallel Efficiency* loss, as illustrated by Fig. 8(b). Based on experience, this situation is quite frequent in strong scaling codes where a large core count usually implies smaller data blocks that can better benefit from the caches, while communication and data transfer become more important because of the reduction in the computation/communication ratio.

The model was originally developed for MPI applications and it was adopted by the POP CoE where it has been extensively used and extended to model *File Input–Output (IO) Scaling*, *Clock Frequency Scaling*, and hybrid executions such as MPI+OpenMP and MPI+CUDA [25].

3.2. JSC: Scalasca/Score-P/CUBE toolset

The *Scalasca* toolset for scalable performance analysis of large-scale parallel application executions [10] is now considered "pretty standard" on current HPC systems ranging from the largest leadership systems (e.g. *Frontier*, *Fugaku*, *LUMI* & *Leonardo*) down to individual multicore notebook computers. It uses the *Score-P* instrumentation and measurement infrastructure and *CUBE* utilities and GUI for analysis report generation, processing and exploration. Automated analysis of MPI and OpenMP communication and synchronisation can be done immediately following trace collection via event replay initiated on the compute nodes and cores allocated to a batch job under control of a nexus utility that combines launching execution of both. All components are available as open source distributed under a 3-clause BSD license, and execution traces can additionally be interactively explored and analysed with the commercial *Vampir*[10] tool.

Application codes for *Scalasca* analysis are prepared via the *Score-P* instrumenter which configures adapters for various measurement interfaces and links associated measurement libraries. Adapters are provided for MPI and SHMEM process-level parallelism, OpenMP and POSIX thread-level parallelism, OpenACC, OpenCL, CUDA, HIP and Kokkos accelerator-based parallelism, MPI-IO and POSIX file I/O activity, as well as source-code instrumentation done via compilers or manually by users. Events from these adapters are timestamped or combined with counters read by PAPI, PERF or rusage, to be aggregated into call-path profiles and/or accumulated in execution event trace buffers per thread which are written to disk during execution finalisation. Instrumenter flags and measurement configuration via environment variables provide flexibility for customisation and refinement.

Fig. 9 shows *Scalasca/CUBE* presentation of a *Score-P* profile summary report collected from development version 7.2 of `Quantum-Espresso` [26] using MPI with OpenACC+CUDAFortran hybrid parallelisation executed on 108 quad-A100 GPU-accelerated compute nodes of *JUWELS-Booster*. The *Score-P* instrumenter orchestrated the NVHPC compilers' instrumentation of application functions, OpenACC constructs from the NVHPC runtime, CUDA events via the CUPTI interface, and MPI library routines. For measurements using the instrumented executable, both OpenACC and CUDA events from both CPU hosts and GPU devices were enabled, since libraries exploiting CUDA(Fortran) were also employed.

The open-source `HemeLB` software[11] developed by University College London and others within the EU HPC CoE for Computational Biomedicine (CompBioMed) is their flagship solver for high-performance parallel lattice-Boltzmann simulations of large-scale three-dimensional hæmodynamic flow in vascular geometries. It supports a range of collision kernels and boundary conditions, and is optimised for sparse, patient-specific geometries. `HemeLB` has traditionally been used to model cerebral bloodflow, and is now being applied to simulating the fully-coupled human arterial and venous trees with high fidelity [27].

As part of a series of POP performance assessments of `HemeLB` on *Archer*, *BlueWaters* and *JUWELS* supercomputers, attention turned to *SuperMUC-NG* at LRZ [28]. It was built with Intel 19.0.4.243 compilers and MPI library, configured to use MPI-3 shared-memory windows within each compute node to reduce memory requirements when loading the initial lattice data. For scalability testing, a cerebral arterial "circle of Willis" geometry dataset of 21.15 GiB was used (corresponding to a lattice spacing of approximately 6.4 microns). After reading and distributing this dataset, the time to simulate blood flow for 5000 lattice time steps (without writing intermediate or final state) was recorded for this strong scaling benchmark.

48 MPI processes were executed on each *SuperMUC-NG* compute node (i.e., one per core, not using additional hardware threads per core) with processes bound to cores and socket-local memory. Executions ran with 864 to 6,144 MPI processes (on 18 to 128 'fat' compute nodes with 768 GiB), whereas executions with 12,288 and more MPI processes ran on regular 'thin' compute nodes (with 96 GB memory).

3.2.1. Execution and scaling efficiencies

The *CUBE* metric panel makes a large number of measured and derived performance metrics accessible via a hierarchical tree. For example, expanding the *Time* tree distinguishes execution time of attached devices from that of the CPUs, and CPU execution time is further split into MPI, OpenACC and CUDA components, before these are

[10] https://www.vampir.eu
[11] https://www.hemelb.org/

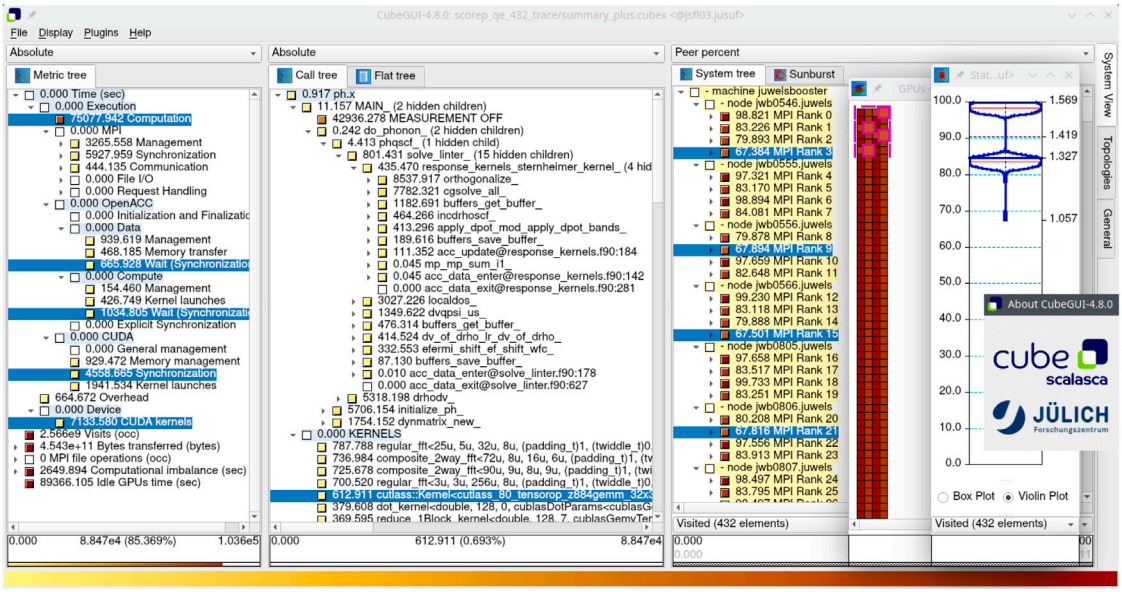


Fig. 9. *Scalasca/CUBE* analysis report explorer presentation of *Score-P* profile from `QuantumEspresso` execution on 108 quad-A100 GPU-accelerated nodes of *JUWELS-Booster* with 432 MPI processes. From the hierarchy of metrics in the left panel CUDA kernel time for GPU device computation and CPU metrics for computation, OpenACC (data and compute) wait synchronisation and CUDA synchronisation time are selected, and this aggregate metric shown in panels to the right. From the call-tree hierarchy in the middle panel, the `cutlass::Kernel` taking the fifth longest GPU execution time has been selected, and the time for this callpath is presented (as percentage of the maximal peer value) in the right panel for the MPI/GPU ranks. Next to each numeric value in the trees is a small box coloured from white through yellow and orange to red according to its percentage of the total metric value to facilitate identification of those which are most significant. Also shown are a violin plot of the distribution of metric values and a 2D topology map of values (compute nodes increase going down and the GPU index on each node is horizontal). These displays clearly show that most GPUs are in two distinct groups and that four of the GPUs (for MPI process ranks 3,9,15,21) take one-third less time than those that take longest, which can be explained by the fact that they have one-fifth fewer instances (visits).

Table 2
`HemeLB` application execution efficiency and scaling relative to 1,152 processes on 24 compute nodes of *SuperMUC-NG*.

Compute nodes	24	32	48	64	96	128	192	256	384	512	768	1024	1536	2048	3072	4096	6452
Processes	**1152**	**1536**	**2304**	**3072**	**4608**	**6144**	**9216**	**12288**	**18432**	**24576**	**36864**	**49152**	**73728**	**98304**	**147456**	**196608**	**309696**
Global scaling efficiency	**0.79**	**0.79**	**0.84**	**0.80**	**0.82**	**0.75**		**0.73**	**0.72**	**0.73**	**0.74**	**0.68**	**0.68**	**0.65**	**0.62**	**0.57**	**0.45**
- Parallel efficiency	0.79	0.80	0.87	0.83	0.86	0.80		0.75	0.74	0.74	0.77	0.71	0.72	0.70	0.72	0.70	0.73
- - Load balance efficiency	0.79	0.80	0.88	0.84	0.86	0.80		0.75	0.74	0.75	0.78	0.72	0.74	0.72	0.74	0.73	0.80
- - Communication efficiency	1.00	1.00	1.00	1.00	1.00	1.00		1.00	1.00	0.99	0.99	0.99	0.98	0.98	0.97	0.96	0.92
- Computation scaling	*1.00*	0.99	0.96	0.96	0.95	0.93		0.98	0.98	0.98	0.96	0.96	0.94	0.93	0.87	0.81	0.61
- - Instructions scaling	*1.00*	1.00	1.00	1.00	1.00	1.00		1.00	1.00	1.00	0.99	0.97	0.94	0.89	0.79	0.67	0.45
- - IPC scaling	*1.00*	0.99	0.96	0.96	0.95	0.93		0.98	0.98	0.99	0.98	0.99	1.00	1.04	1.11	1.21	1.36
IPC	1.411	1.395	1.353	1.355	1.342	1.316		1.377	1.387	1.396	1.383	1.390	1.417	1.473	1.566	1.704	1.919
											Key:	<0.65	<0.75	<0.85	<0.95	<1.00	>1.00

Legend for hierarchy of efficiencies:
Global scaling efficiency is the product of parallel efficiency and computation scaling.
Parallel efficiency is the ratio of mean computation time to total runtime of all processes.
Load balance efficiency is the mean/maximum ratio of computation time outside of MPI.
Communication efficiency is the ratio of maximum computation time to total runtime.
Serialisation efficiency is estimated from idle time within communications where no data is transferred.
Transfer efficiency relates to essential time spent in data transfers (where there is no waiting).
Computation scaling is the relative total time in computation (outside of MPI).
Instructions scaling is the relative total number of instructions executed (outside of MPI).
IPC scaling is the relative value of instructions executed (outside of MPI) per CPU cycle.
(Scaling efficiencies are relative to a serial execution or the smallest parallel execution configuration.)

further subdivided. While this is a natural organisation of these metrics that can be explored, the sheer number of available metrics can initially be overwhelming. As part of the collaboration with BSC within the POP CoE, the efficiency metrics defined there [29] have been incorporated within *CUBE* where they provide a convenient characterisation of the execution measurement that directs investigation of specific inefficiencies. While by default the entire execution is characterised, it is generally more appropriate to select a specific call-path as focus of this analysis which is the key execution phase. *CUBE* supports interactive exploration of call-paths and their evaluation. With the efficiencies and related metric values from a set of measurements with increasing numbers of processes (or threads), additional scaling efficiencies can be determined.

Table 2 summarises execution efficiencies and scalability of `HemeLB` executed on *SuperMUC-NG* with up to 6,452 compute nodes (309,696 MPI processes). Computational instructions retired per clock cycle (IPC) was a reasonable 1.9, compared to 1.4 for the smaller execution configurations, suggesting better cache efficiency as the lattice partitions get smaller. Perfect instruction scaling up to 768 compute nodes thereafter deteriorates as there is more processing of lattice block boundaries compared to their interiors. Since these two effects counteract each other, very good computation scaling above 0.87 is sustained. Efficient non-blocking communication to exchange fluid particles between neighbouring lattice blocks maintains excellent communication efficiency above 0.97. The most significant inefficiency at all scales tested is load balance, generally around 0.80 but dropping to

0.72 in some larger execution configurations. While this is still fairly good, it presents the largest opportunity for performance improvement and warrants more in-depth investigation.

3.3. Complementarity and development of BSC & JSC tools

The POP efficiency model implemented within BSC and JSC tools helps ensure that performance analysts can relate their observations using a common terminology, providing a convenient starting point for more in-depth investigations best suited to one or the other tool, and facilitating switching between them if necessary to address distinct aspects. Their approaches for specification, customisation and refinement of instrumentation, measurement and analysis are quite complementary.

Score-P measurements require prior instrumentation of applications, however, its call-path profiles provide an initial starting point for selective refinement prior to trace collection. Such profiles may already contain sufficient information to diagnose problems, particularly when there is no need to distinguish communication efficiency factors. By scoring the profiles according to the events they aggregate, indicative estimates of required trace (buffer and file) sizes are possible, and their content can be optimised by various instrumentation and measurement customisation capabilities prior to eventual trace collection. *Scalasca* automated analysis of the traces conveniently determines the additional inefficiency metrics required to calculate the additional communication efficiency factors. The inherently parallel implementation of this analysis also enables it to be applied to much larger traces, however, it relies on strict semantics for recorded events.

On the other hand, the semantic-free nature of *Extrae* traces enables much more flexible user-driven analysis with *Paraver* which can be applied to a wider set of trace measurements including those which are incomplete. *Extrae* also conveniently avoids the need for initial instrumentation of application executables, and rather than including user functions only captures events of the parallel runtime by default. Where traces would still be prohibitively voluminous from larger and longer executions, more compact traces can be obtained by switching to a mode that periodically switches sets of hardware counters.

While MPI message-passing and OpenMP multi-threading are well-established parallel programming paradigms, they continue to be extended and support by tools has to keep up. Often this depends on the associated compilers, runtimes and libraries. Attached or co-located accelerator devices are increasingly common, and a range of programming and offload mechanisms are available which need to be supported. Along with OpenMP, OpenACC, OpenCL, there are proprietary CUDA, HIP/ROCm, SYCL and others, all with different measurement interfaces and execution characteristics. As BSC and JSC tools develop measurement support for these additional paradigms, corresponding adaptation and extension of the analysis and efficiency model is required.

Close interaction with other tools and application developers at training workshops and within the context of JLESC and VI-HPS has facilitated targeted refinement of our tools and methodology.

4. Training

To support people interested in learning about parallel performance analysis tools and their use to understand and improve the execution performance of their parallel applications, we have developed a range of training material which is employed in a variety of training formats serving distinct user profiles. Specific lessons learned from each are included here, with more general aspects discussed in the next section.

4.1. Tutorials

Tutorials of a full-day (6 or 7 h) or half-day (3 or 3.5 h) durations have been regularly selected and offered at the foremost HPC conferences, the International Supercomputing Conference (ISC) High Performance annually in Germany and the International Conference for HPC, Networking, Storage and Analysis (SC) annually in USA.

Half-day tutorials offering an overview of the variety of tools available *Supporting performance analysis and optimisation on extreme-scale computer systems* [30] have been complemented by full-day hands-on training in *Hands-on practical hybrid parallel application performance engineering* with core VI-HPS tools centred on the *Score-P* instrumentation and measurement infrastructure with complementary analyses provided by *Scalasca/CUBE*, *TAU/ParaProf/PerfExplorer* & *Vampir* [31]. (The *Periscope* Tuning Framework [32] was also sometimes included [33].) For half-day versions of this tutorial where it was impractical to include hands-on activity, demonstrations using the tools were substituted.

An additional half-day hands-on tutorial *Determining parallel application execution efficiency and scaling using the POP methodology* has also been offered at ISC-HPC [34,35], concentrating on analyses derived with BSC (*Paraver*) and JSC (*Scalasca/CUBE*) toolsets from performance measurements of actual HPC application executions. Tutorial participants engaged with the tools via exercises on their own notebook computers to prepare them to locate and diagnose efficiency and scalability issues of their own parallel application codes.

Due to the time constraints and often limiting network infrastructure for tutorials held at conferences, exercises are restricted to provided mini-application/benchmark codes such as NAS Parallel Benchmark (NPB) [36] `BT-MZ` using hybrid MPI+OMP parallelisation. Small problem configurations can be executed interactively in a virtual machine running on participants' notebook computers or remotely in an AWS cloud instance, or larger problem configurations executed via the batch system on one or two compute nodes of an HPC cluster at JSC with provided training accounts. In response to increasing interest in use of GPU-accelerated computing, recent hands-on tutorials have progressed to using an MPI+CUDA parallelisation of the `TeaLeaf` code from the UK Mini-App Consortium[12] run on *JUWELS-Booster* quad-A100 GPU nodes.

Another commonly used benchmark is `LULESH` [17] that stresses compiler vectorisation, OpenMP overheads and on-node parallelism. It offers several benefits, including its small code size, easy compilation, and support for MPI, OpenMP, and their hybrid combination, CUDA, OpenACC, and OpenCL versions. This versatility allows adaptation to the diverse hardware platforms available in training courses. There is a restriction in running `LULESH`: the number of domains (equal to the number of MPI tasks) must always be the cube of a natural number. However, this restriction can actually be considered a benefit for educational purposes. Since machines are typically dual socket with an even number of cores per socket, running a cube number or processes, for example 27, results in an uneven distribution of processes per socket, causing a degree of imbalance. This restriction allows analysis of this effect using tools and provides an opportunity to experiment with system process mapping options to reconfigure the pinning of processes to cores, altering resource sharing and analysing the effects.

Experience with the use of ISO/OVA images running in a VM on notebook computers varied widely according to notebook capability and was constrained to very small execution configurations of typically four MPI processes each with 2 or 3 OpenMP threads. Even this results in CPU over-subscription that is not recommended for production HPC systems with associated high run-to-run and within-run variability that are not ideal for parallel performance analysis exercises. While those undesirable aspects are avoided using a dedicated remote HPC system, access has become increasingly complicated with the need for multi-factor authentication, installation of an SSH+X11 client for those using Windows OS on their notebook computers, and generally sluggish responsiveness of remote X11-based graphical tools. Remote access to compute resources in AWS cloud instances can be done conveniently

[12] https://uk-mac.github.io

via a standard web browser, with web protocols improving responsiveness of graphical tools, however, the expense of dedicated instances and variability otherwise were generally unsatisfactory, particularly for performance measurements of multi-node (accelerated) executions.

In the last couple of years, the best setup has been JupyterLab with Xpra remote desktop [37], providing remote access via web browsers to dedicated training accounts on JSC accelerated compute nodes for various hands-on tutorials (often running concurrently and sharing a reserved machine partition). Open OnDemand [38] interactive desktop provides a similar setup for other HPC systems (such as *Fugaku* at R-CCS and *Delta* at NCSA).

4.2. Tuning workshops

VI-HPS Tuning Workshops are distinguished from tutorials in various ways. They are hosted by HPC centres using their local HPC cluster or supercomputer system, often primarily for their own user community of students and application developers. They run over several (typically three to five consecutive) days, providing more time to cover a wider range of tools, often including as guests proprietary tools recommended by the system vendor (e.g. Cray/HPE, IBM, Intel or Nvidia) or less mature locally-developed tools for experimentation. In particular, after the usual tutorial-style tool introduction and hands-on exercise with a benchmark/mini-app, substantial time is reserved for participants to then apply the presented tools to their own application codes, with coaching and guidance provided by the instructors and classroom assistants.

Well-prepared participants work in small teams with a range of simple/short testcases to more complex and longer/larger execution testcases, such that they can rapidly experiment with various instrumentation and measurement options before progressing to more representative and interesting cases (which might run overnight). Participants (such as students) without their own application code, are given access to a variety of benchmark/mini-apps they can work with, or they may prefer to observe other participants work on their applications.

Often a couple of related tools are introduced each morning, with the entire afternoon for their own work, and a review of participants' experience (issues encountered and resolved, insights obtained and future investigations) concluding the day. By the end of the workshop participants have had the opportunity to try out and compare a range of tools and consider how they suit their application needs. While it is also possible to make small changes to source code or build and run configurations to investigate optimisation opportunities during such workshops, most analysis and optimisation generally occurs afterwards.

The first VI-HPS Tuning Workshop was organised and hosted by RWTH Aachen in March 2008, with workshop frequency increasing to three or four each year. Workshops have been hosted by HPC centres around the world and not always VI-HPS partner institutions, on a multitude of different systems and a wide variety of VI-HPS and guest tools included. Particularly distinguished workshops hosted by JLESC partners used *JUQUEEN* IBM BlueGene/Q at JSC, Fujitsu *Kei* computer with SPARC64 processors at RIKEN AICS, *MareNostrum4* and its GPU-accelerated *CTE-Power* module at BSC, and TACC *Stampede2* with Intel Xeon Phi processors for a workshop at UTK-ICL.

After 33 VI-HPS Tuning Workshops held in person, in 2020 the COVID19 virus pandemic forced all training to rapidly move online via a variety of videoconferencing solutions (some much less functional or suitable than others). Despite initial trepidation about the change of format and its impact on close interaction and engagement with workshop participants in hands-on sessions, it was quickly observed that breakout rooms and participant's sharing their screens generally worked well. Attendance of workshops remained high, with participants joining from countries who would not have otherwise been able to travel to attend events in person. While the timezone difference made participation from other continents inconvenient, it was largely compensated by the provision of session recordings that participants could refer to when suitable for them.

The nine virtual VI-HPS Tuning Workshops held to date also included the first featuring performance analysis of applications running on GPU accelerators (first using *Marconi100* at CINECA, then *JUWELS-Booster* at JSC). 2024 returned to in-person VI-HPS Tuning Workshops again, and the first hybrid VI-HPS Tuning Workshop split between RWTH Aachen and TU Dresden.

Multi-day events (typically lasting a full week) dedicated to application performance analysis, optimisation and tuning, often with particular emphasis on scaling to large fractions of HPC supercomputer systems have also become established, particularly as JSC targeted extreme-scaling to its entire BlueGene systems (*JUBL*, *JUGENE* & *JUQUEEN*) [39] and on subsequent modular supercomputers including the current flagship system *JUWELS Cluster+Booster*. Such events provide more focused engagement with application development teams geared to the considerable specific challenges of scaling their codes on particular supercomputer systems.

4.3. Staggered workshop series

With travel for both workshop participants and instructors not possible in 2021 due to the COVID19 pandemic, we had to adjust plans for a VI-HPS Tuning Workshop to be hosted by Durham University, UK. Instead of simply having a virtual version of the usual in-person workshop, we transformed it into a series of one-day sessions spread over an entire year, with the usual full day of tools instruction and hands-on assistance per month on a regular cadence. While such an arrangement is not practical for participants of in-person workshops, who would need to travel frequently, for a virtual workshop this offered many possible benefits that were worth investigating further.

On registration participants received access to the DiRAC *COSMA/DINE* cluster to prepare their application code and testcase(s) in advance of the workshop start, and could continue to use it throughout the year. Teamwork within application code teams was strongly encouraged, both when registering for the workshop series and via Slack channels established for interaction with instructors and other participants. Over a dozen international teams and also many individuals participated, introducing their application codes in the initial kick-off session and providing updates in subsequent sessions, including the concluding wrap-up session of the workshop. The four-week gap between sessions allowed plenty of time for participants to both investigate use of the tools with their applications and try out potential optimisations suggested by the analysts, with some significant successes reported. Unfortunately, the length of the gap between sessions resulted in drop off of attendance as the series progressed and the need for extensive recaps of previous sessions (despite recordings being available), while many participants were only active on the presentation days [40].

For the 2023 repetition of this workshop series, a weekly cadence of five full-day sessions was chosen that featured a smaller set of performance tools. This compressed format has been found to be more effective, with sustained engagement throughout, and was therefore repeated in 2024. A hybrid format was offered where participants were encouraged to attend in-person. While the in-person attendees reported benefits from close engagement with instructors and interaction with classmates, most participants found it more practical and convenient to join remotely.

4.4. Workshops by & for under-represented groups in HPC

Participation of women at training events has varied considerably from event to event but generally reflected their low representation in the HPC community. While women are relatively well represented within VI-HPS, the POP CoE and associated tools projects, our teams of instructors provided from organising institutions have often contained only a single instructrix or sometimes none at all.

Women within the framework of the POP CoE were therefore inspired to promote and increase the participation of women in training, and two dedicated workshops have been organised. The main target was to provide training opportunities where all of the instructors presenting and their class assistants were women. Early registration favoured women and other under-represented groups in HPC, however, available places in the workshops were filled by others who applied to be on a waiting list: whereas the first workshop was almost exclusively women, men constituted around a quarter of the participants (and assistants) in the follow-up.

This kind of initiative has multiple positive impacts, from more straightforward to less obvious but not less important. The most direct one is more women attending technical training and thereby increasing their technical knowledge and HPC skills. But also offering priority over male colleagues who outnumber them to level out the field. Another positive impact is the opportunity for mid-career professional women to participate as trainers and gain invaluable experience.

These might be the most obvious, but we also obtain other more subtle benefits: to increase visibility of women in technical activities providing training of the same level and quality as those given by male trainers. It is essential to provide these role models to young women and the entire community, demonstrating that women can provide training in this area of the highest technical quality.

The experience has been a success and we were able to attract a similar number of attendees for these events as for the regular training while ensuring the same technical level. The feedback collected from the attendees was very positive. It can be summarised in two aspects: on one hand, the supportive atmosphere where they feel free to ask questions and make mistakes, and on the other hand, the empowerment of attending an event where the leading experts were women. Also, the feedback from the instructors and assistants illustrated their professional growth, motivating us to continue organising these training events in the future.

Due to their scheduling during the pandemic, these first two workshops have been organised online, which meant that we largely missed the networking aspect of such an event. We are planning to offer an in-person workshop in future where this will be addressed.

Experience from more workshops (and tutorials) is needed to evaluate the real impact and benefits of such events while monitoring the risk that having a specific event for women may have an undesirable side-effect of reducing their attendance to the unrestricted training events. Furthermore, they should not be considered second-class or remedial training that is less valuable than non-segregated training events which benefit from the full community diversity.

5. Discussion

Appropriate training with performance tools is needed for the entire spectrum of users of HPC computer systems, primarily those developing their own applications (who can most readily modify them as necessary) but also those running applications developed by others (where runtime configuration parameters can be adjusted). A variety of training needs have been addressed with distinct training formats, trading duration for depth of coverage and exploiting format opportunities to the greatest possible extent. As well as specific lessons learnt and already discussed for each of the training formats, some generic lessons can also be identified.

Training that showcases capabilities of a variety of tools provides benefits for both those just getting started as well as application developers with more advanced needs. Many tools have complex functionality to adapt to user and application requirements, which poses challenges in recommending a single tool or toolset as universal starting point for all. Trying out tools with participants' own application codes, or failing that with prepared examples, is essential for understanding and familiarising with the usage and analysis results of the tools.

While novices benefit most from introductory content focused on provided simple examples, particularly when they have yet to develop application code of their own, intermediate and advanced training is most productive when participants work with their own application codes. As well as ensuring that the code runs correctly on whatever computer system is being employed, the testcase configurations should be suitably defined to allow short executions on a few compute nodes as well as realistic large-scale executions that may require much longer (but can be scheduled to run overnight). Participants working together as code-teams are particularly productive.

Short tutorials are often sufficient to learn basic usage of performance tools and techniques, whereas comprehensive and advanced use is best spread over several days, allowing familiarisation with initial concepts before progressing to more advanced and less common aspects. Training provided in person has advantages but also drawbacks compared to virtual alternatives that are still being debated.

In-person training supports close interaction between instructors/assistants and participants, but has additional financial and logistics demands (particularly travel and accommodation) that can limit accessibility, whereas virtual training (particularly when recorded) can more conveniently be widely accessed but limits interaction and engagement. Virtual training also reduces the need to have training on consecutive days, with gap days or multi-week/multi-month spread all possible, so participants can make their own progress between scheduled instruction.

Hybrid training delivery combining both in-person and virtual aspects is starting to be offered, however, it requires adequate resourcing which essentially combines two separate concurrent workshops. There are additional organisational demands for live-streaming while catering for in-person participants. Instructors and assistants are expected to assist remote participants without neglecting those physically present in the classroom, such that all have an acceptable experience.

In practice, remote training participants have generally been found to be more looking for a general overview of tools' capabilities and less ready to immediately benefit from practical application of the tools. They are therefore able to selectively drop out of sessions dedicated to trying out and applying tools to participants' own codes, allowing instructors and assistants to dedicate their time accordingly.

Recordings of presentation sessions are valued by those who were unable to attend them, as well as allowing those who joined them to recap on details presented in following days and weeks. With recording typically organised by the hosts of the training events, and made available via a variety of mechanisms (some licensed only for registered participants), it makes it increasingly challenging to maintain an up-to-date archive of recordings accessible to the wider community.

Finally, it is worth noting that tools developers greatly value the opportunities provided in training events to directly observe how their tools are employed by a variety of users with diverse application codes and receive immediate feedback. As well as pinpointing problematic aspects, it allows rapid pivoting to jointly investigate alternative approaches and try out newly developed functionality.

6. Conclusion

Over more than fifteen years, parallel performance measurement and analysis training offered by the developers of the *Scalasca* and *Paraver* toolsets has adapted to evolving requirements, such as governed by the recent pandemic and changing application developer needs, and specifically to address under-represented groups in HPC such as women. Joint training events, based on common methodology with complementary tools, have been particularly productive and well received. We expect to continue this proven approach to the challenges of the exa-scale era, via the now established Centre of Excellence for HPC Applications Performance Optimisation and Productivity (POP) and Virtual Institute-High Productivity Supercomputing (VI-HPS), and supported by the Joint Laboratory for Extreme-Scale Computing (JLESC).

CRediT authorship contribution statement

Brian J.N. Wylie: Writing – review & editing, Writing – original draft, Software, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization. **Judit Giménez:** Writing – review & editing, Writing – original draft, Software, Methodology. **Christian Feld:** Software. **Markus Geimer:** Software. **Germán Llort:** Writing – original draft, Software. **Sandra Mendez:** Writing – review & editing. **Estanislao Mercadal:** Writing – original draft, Software. **Anke Visser:** Software. **Marta García-Gasulla:** Writing – review & editing, Methodology.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Brian Wylie reports a relationship with Association for Computing Machinery that includes: speaking and lecture fees and travel reimbursement. Brian Wylie reports a relationship with IEEE that includes: speaking and lecture fees and travel reimbursement. Brian Wylie reports a relationship with ISC Group that includes: board membership. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The authors do not have permission to share data.

Acknowledgements

This work was partially supported by JLESC project *Developer tools for porting and tuning parallel applications on extreme-scale parallel systems* also involving Itaru Kitayama, Chigusa Kobayashi, Hitoshi Murai and Miwako Tsuji (RIKEN R-CCS), and Anthony Danalis and Heike Jagode (UTK-ICL).

Our BSC and JSC performance tools are separately developed and maintained as open-source by many of our colleagues who are also active in our training activities, and have been funded by the European Commission's EuroHPC Joint Undertaking, BMBF, DFG and Helmholtz Gemeinschaft in Germany, and USA Dept of Energy.

Our performance measurement and analysis methodology was jointly developed with our partners from IT4Innovations, NAG Ltd, RWTH Aachen Universität, Universität Stuttgart (HLRS), and Université de Versailles Saint-Quentin (UVSQ) in the HPC Centre of Excellence Performance Optimisation and Productivity (POP CoE) funded by the European Commission Horizon programmes (grants 676553, 824080 and 101143931).

References

[1] G. Cameron, B.J.N. Wylie, D. McArthur, PARAMICS – Moving vehicles on the Connection Machine, in: Proceedings of the 1994 ACM/IEEE Conference on Supercomputing (Washington, DC, USA), 1994, pp. 291–300, http://dx.doi.org/10.1109/SUPERC.1994.344292.

[2] G. Wilson, Teaching Tech Together: How to Create and Deliver Lessons That Work and Build a Teaching Community Around Them, Taylor & Francis, ISBN: 978-0-367-35328-5, 2019, [Online]. Available: http://teachtogether.tech/.

[3] HPC Centres of Excellence Council (HPC[3]), Centres of Excellence for HPC Applications, [Online]. Available: https://www.hpccoe.eu/eu-hpc-centres-of-excellence2/.

[4] XcalableMP Handbook, 2018, [Online]. Available: http://xcalablemp.org/handbook/.

[5] H. Murai, M. Nakao, M. Sato, XcalableMP Programming Model and Language, Springer, Singapore, 2021, http://dx.doi.org/10.1007/978-981-15-7683-6_1.

[6] J. Lee, M. Sato, Implementation and performance evaluation of XcalableMP: A parallel programming language for distributed memory systems, in: 39th International Conference on Parallel Processing Workshops, ICPP, San Diego, California, USA, 2010, pp. 413–420, http://dx.doi.org/10.1109/ICPPW.2010.62.

[7] M. Tsuji, M. Sato, M.R. Hugues, S.G. Petiton, Multiple-SPMD programming environment based on PGAS and workflow toward post-petascale computing, in: 42nd International Conference on Parallel Processing, ICPP, Lyon, France, 2013, pp. 480–485, http://dx.doi.org/10.1109/ICPP.2013.58.

[8] V. Pillet, J. Labarta, T. Cortes, S. Girona, PARAVER: A Tool to Visualize and Analyze Parallel Code, Universitat Politècnica de Catalunya, 1995, RR-95/03.

[9] Barcelona Supercomputing Center – Performance Tools, 2023, [Online]. Available: http://tools.bsc.es.

[10] M. Geimer, F. Wolf, B.J.N. Wylie, E. Ábrahám, D. Becker, B. Mohr, The Scalasca performance toolset architecture, Concurr. Comput.: Pract. Exper. 22 (6) (2010) 702–719, http://dx.doi.org/10.1002/cpe.1556.

[11] A. Knüpfer C. Rössel, D. an Mey, S. Biersdorf, K. Diethelm, D. Eschweiler, M. Geimer, M. Gerndt, D. Lorenz, A. Malony, W.E. Nagel, Y. Oleynik, P. Philippen, P. Saviankou, D. Schmidl, S. Shende, R. Tschüter, M. Wagner, B. Wesarg, F.Wolf, Score-P: A joint performance measurement run-time infrastructure for Periscope, Scalasca, TAU, and Vampir, in: H. Brunst, M. Müller, W. Nagel, M. Resch (Eds.), Tools for High Performance Computing 2011, Springer, 2011, http://dx.doi.org/10.1007/978-3-642-31476-6_7.

[12] Innovative Computing Laboratory, University of Tennessee, Performance Application Programming Interface (PAPI), 2023, [Online]. Available: https://icl.utk.edu/papi.

[13] Virtual Institute – High Productivity Supercomputing, VI-HPS Tools Guide, 2023, [Online]. Available: http://www.vi-hps.org/tools/cms/upload/material/general/ToolsGuide.pdf.

[14] Barcelona Supercomputing Center – Performance Tools, Extrae, 2023, [Online]. Available: http://tools.bsc.es/extrae.

[15] Barcelona Supercomputing Center – Performance Tools, Paraver, 2023, [Online]. Available: http://tools.bsc.es/paraver.

[16] Barcelona Supercomputing Center – Performance Tools, Paraver – tracefile description, 2023, [Online]. Available: https://tools.bsc.es/doc/1370.pdf.

[17] I. Karlin, J. Keasler, R. Neely, LULESH 2.0 updates and changes, 2013, LLNL-TR-641973, Livermore, CA, USA, [Online]. Available: https://asc.llnl.gov/codes/proxy-apps/lulesh.

[18] J. González, J. Giménez, J. Labarta, Automatic detection of parallel applications computation phases, in: Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium, IPDPS, 2009, http://dx.doi.org/10.1109/IPDPS.2009.5161027.

[19] G. Llort, H. Servat, J. González, J. Giménez, J. Labarta, On the usefulness of object tracking techniques in performance analysis, in: Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis, 2013, http://dx.doi.org/10.1145/2503210.2503267.

[20] H. Servat, G. Llort, J. Giménez, K. Huck, J. Labarta, Folding: Detailed analysis with coarse sampling, tools for high performance computing 2011, in: Proceedings of the 5th International Workshop on Parallel Tools for High Performance Computing, http://dx.doi.org/10.1007/978-3-642-31476-6_9.

[21] G. Llort, J. Labarta, Intelligent Instrumentation Techniques to Improve the Traces Information-Volume Ratio, Universitat Politècnica de Catalunya – BarcelonaTech, 2015, [Online]. Available: http://hdl.handle.net/10803/326469.

[22] Barcelona Supercomputing Center – Performance Tools, Dimemas, 2023, [Online]. Available: http://tools.bsc.es/dimemas.

[23] M. Casas, R. Badia, J. Labarta, Automatic analysis of speedup of MPI applications, in: Proceedings of the 22nd Annual International Conference on Supercomputing, ISC'08, 2008, http://dx.doi.org/10.1145/1375527.1375578.

[24] A. Stone, J.M. Dennis, M.M. Strout, The CGPOP Miniapp, Version 1.0, Colorado State University, Computer Science Department, Tech. Report CS-11-103, 2011, [Online]. Available: https://www.cs.colostate.edu/TechReports/Reports/2011/tr11-103.pdf.

[25] J. Giménez, E. Mercadal, G. Llort, S. Mendez, Analyzing the efficiency of hybrid codes, in: Proc. 19th International Symposium on Parallel and Distributed Computing, ISDPC, Warsaw, Poland, 2020, http://dx.doi.org/10.1109/ISPDC51135.2020.00014.

[26] I. Carnimeo, F. Affinito, S. Baroni, O. Baseggio, L. Bellentani, R. Bertossa, P.D. Delugas, F.F. Ruffino, S. Orlandini, F. Spiga, P. Giannozzi, Quantum ESPRESSO: One further step toward the exascale, J. Chem. Theory Comput. 19 (2023) 6992–7006, http://dx.doi.org/10.1021/acs.jctc.3c00249.

[27] J.W.S. McCullough, R.A. Richardson, A. Patronis, R. Halver, R. Marshall, M. Ruefenacht, B.J.N. Wylie, T. Odaker, M. Wiedmann, B. Lloyd, E. Neufeld, G. Sutmann, A. Skjellum, D. Kranzlmüller, P.V. Coveney, Towards blood flow in the virtual human: Efficient self-coupling of HemeLB, J. R. Soc. Interface Focus (2020) http://dx.doi.org/10.1098/rsfs.2019.0119.

[28] B.J.N. Wylie, HemeLB on SuperMUC-NG Performance Assessment Report, POP2_AR_041, 2020, [Online]. Available: http://dx.doi.org/10.5281/zenodo.4105742.

[29] Performance Optimisation and Productivity Centre of Excellence in HPC, POP standard metrics for parallel performance analysis, 2016, [Online]. Available: https://pop-coe.eu/node/69.

[30] B. Mohr, M. Schulz, Brian J.N. Wylie, Supporting performance analysis & optimization on extreme-scale computer systems – Current state & future directions, in: Tutorial at ISC'12 Conference (Hamburg, Germany, June 2012 and SC12 Conference (Salt Lake City, UT, USA), 2012, [Online]. Available: http://hdl.handle.net/2128/4902.

[31] G. Corbin, C. Feld, M. Geimer, M. Gerndt, J. Linford, A. Malony, Y. Oleynik, M. Schlütter, S. Shende, R. Tschüter, A. Visser, M. Weber, B. Wesarg, B. Williams, F. Winkler, B.J.N. Wylie, J. Ziegenbalg, Hands-on practical hybrid parallel application performance engineering, in: Tutorial at ISC High Performance & SC Conferences: Denver, CO, USA, Nov. 2023; Hamburg, Germany, May 2023; Dallas, TX, USA, Nov. 2022; Hamburg, Germany, May 2022; Online/St.Louis, MO, USA, Nov. 2021; Frankfurt am Main, Germany, June 2021; Online/Atlanta, GA, USA, Nov. 2020; Denver, CO, USA, Nov. 2019; Frankfurt am Main, Germany, June 2019; Frankfurt am Main, Germany, June 2018; Denver, CO, USA, Nov. 2017; Frankfurt am Main, Germany, June 2017; Salt Lake City, UT, USA, Nov. 2016; Frankfurt am Main, Germany, June 2016; Austin, TX, USA, Nov. 2015; Frankfurt am Main, Germany, June 2015; New Orleans, LA, USA, Nov. 2014; Leipzig, Germany, June 2014; Denver, CO, USA, Nov. 2013; Leipzig, Germany, June 2013, [Online]. Available: http://hdl.handle.net/2128/33425.

[32] M. Gerndt, E. Cesar, S. Benkner, Automatic Tuning of HPC Applications — the Periscope Tuning Framework (PTF), Shaker Verlag, ISBN: 978-3-8440-3517-9, 2015.

[33] I. Compres, J. Protze, M. Schulz, R. Tschüter, B.J.N. Wylie, Tools for high productivity supercomputing, in: Tutorial at Europar'13 European Conference on Parallel Computing (Aachen, Germany), 2013, http://hdl.handle.net/2128/5923.

[34] J. Giménez, B.J.N. Wylie, Determining parallel application execution efficiency & scaling using the POP methodology, in: Tutorial at ISC High Performance Conference (Online, Germany), 2021, [Online]. Available: http://hdl.handle.net/2128/30380.

[35] M. García-Gasulla, S. Mendez, A. Visser, B.J.N. Wylie, Determining parallel application execution efficiency & scaling using the POP methodology, in: Tutorial at ISC High Performance Conference (Hamburg, Germany), 2024.

[36] R.F. Van der Wijngaart, H. Jin, NAS Parallel Benchmarks, Multi-Zone Versions, NASA Advanced Supercomputing Division, NAS-03-010, Moffett Field CA, 2003, [Online]. Available: https://www.nas.nasa.gov/software/npb.html.

[37] J.H. Göbbert, T. Kreuzer, A. Grosch, A. Lintermann, M. Riedel, Enabling interactive supercomputing at JSC lessons learned, in: R. Yokota, J. Shalf, M. Weiland, S. Alam (Eds.), ISC High Performance 2018 International Workshops, Revised Selected Papers, in: Lecture Notes in Computer Science, vol. 11203, Springer Verlag, 2018, pp. 669–677, http://dx.doi.org/10.1007/978-3-030-02465-9_48.

[38] D. Hudak, D. Johnson, A. Chalker, J. Nicklas, E. Franz, T. Dockendorf, B.L. McMichael, Open OnDemand: A web-based client portal for HPC centers, J. Open Source Softw. 3 (25) (2018) 622, http://dx.doi.org/10.21105/joss.00622.

[39] D. Brömmel, W. Frings, B.J.N. Wylie, B. Mohr, P. Gibbon, T. Lippert, The High-Q Club: Experience with extreme-scaling application codes, Supercomput. Front. Innov. 5 (1) (2018) 59–78, http://dx.doi.org/10.14529/jsfi180104.

[40] A. Basden, M. Weinzierl, T. Weinzierl, B.J.N. Wylie, A novel performance analysis workshop series concept, developed at Durham University under the umbrella of the ExCALIBUR programme, 2021, http://dx.doi.org/10.5281/zenodo.5155503, Zenodo & ExCALIBUR, [Online]. Available:.

Brian J.N. Wylie has been a scientific researcher in Jülich Supercomputing Centre since 2004, in the group developing the Scalasca toolset for scalable performance analysis of large-scale parallel applications. He established and continues to contribute to tools training activities of the Virtual Institute-High Productivity Supercomputing (VI-HPS). His current focus is the assessment of exascale readiness of applications comprising very large numbers of processes and threads on heterogeneous accelerated computer systems within the Performance Optimisation and Productivity (POP) Centre of Excellence in HPC.