Process-Oriented Performance Engineering Service Infrastructure for Scientific Software at German HPC Centers

Whitepaper — November 2020

Dieter an Mey, Alesja Dammer, Robert Dietrich, Jan Eitzinger, Nicole Filla, Georg Hager, Jonas Hahnfeld, Paul Kapinos, Anara Kozhokanova, Thomas Röhl, Daniel Schürhoff, Sandra Wienke, Frank Winkler, Thomas Zeiser

Gerhard Wellein	—	**Friedrich-Alexander Universität Erlangen-Nürnberg, Regionales Rechenzentrum Erlangen (RRZE)** – (coordinator)
Matthias S. Müller	—	**RWTH Aachen University, IT Center (RWTH)**
Wolfgang E. Nagel	—	**Technical University Dresden, Zentrum für Informationsdienste und Hochleistungsrechnen (ZIH)**

Contents

Chapter 1

Introduction

High Performance Computing (HPC) systems have always had complex system architectures that are subject to constant change, which presents a challenge for the use and programming of these systems. Furthermore, the increasing investment and operating costs of modern HPC systems require their efficient use. For this reason, HPC data centers offer their HPC users support services at various levels. Basic support is always available at helpdesk level, covering questions about available resources or access modes. In addition, larger centers often provide support for basic programming, code performance and parallelization issues. Some centers offer expert assistance on specific application areas, HPC topics, or may be involved in advanced HPC teaching topics. These activities are often based on the research topics of the academic institution in which the HPC center is embedded. The lack of experienced support staff and available financial resources has further increased the specialization of the HPC centers in in-depth HPC user support and training topics. In parallel, the diversity and complexity of HPC systems has increased significantly and new application communities have entered the field. These developments require the implementation of sustainable concepts for the provision of appropriate and qualified HPC support services, including training activities and nationwide joint efforts to use distributed HPC expertise in a transparent manner.

To this end the DFG has issued the call "Performance Engineering for Scientific Software" (Nov. 2015; see [1]) to demonstrate the need for hardware efficient utilization of HPC systems and to establish appropriate HPC user support structures. A special focus was put on HPC centers embedded into German universities. The project "Process-oriented Performance Engineering Service Infrastructure for Scientific Software at German HPC Centres" (ProPE) dealt with this call for proposals in the context of a structured distributed Performance Engineering (PE) initiative involving three established academic HPC centers distributed throughout Germany: IT Center in Aachen, ZIH in Dresden and Regional Computer Center Erlangen. The aim of the project was to evaluate and implement common processes, methods and tools to create a blueprint for a nationwide PE infrastructure in which distributed HPC expertise can be used transparently.

The three ProPE partners have long provided the HPC support services described above and have continuously participated in HPC training activities. They are strongly embedded in different application communities and pursue complementary HPC research directions. As they are spread over three states, they also represent different approaches of nationwide coordination structures in HPC, which also need to be integrated into a national PE infrastructure.

This white paper summarizes central methods, processes and tools for distributed PE that were developed and implemented within the ProPE project. It also provides experience in the process of building the distributed infrastructure and identifies infrastructures, processes and data/documentation formats that need to be centrally maintained. The project also emphasizes the need to share structured and appropriate performance engineering approaches, clearly defined IT service processes and documentation, and reliable and robust tools. We first identify the requirements for a distributed PE support infrastructure (see Chapter 3), which components are necessary and how these components could be implemented (see Chapter 4). The paper concludes with a management summary (see Chapter 5).

[1] `https://www.dfg.de/foerderung/info_wissenschaft/2015/info_wissenschaft_15_75/index.html`

Chapter 2

Current status of German HPC landscape

The landscape of HPC centers in Germany is organized as a pyramid following the recommendations of the German Council of Science and Humanities [1] (Wissenschaftsrat, WR) [27]. It covers basically three tiers of HPC capacity/capability systems matching the user range to be served (national/european, regional, local) comprising centers of different sizes (see Figure 2.1 for the current state of the pyramid). The high-end demand on European and national (**Tier-0/1**) level is served by the three **Gauss Center for Supercomputing** (GCS) members HLRS (Stuttgart), LRZ (Garching) and JSC (Jülich). A sustainable funding stream for GCS has been established by the federal government and the hosting states covering full costs of operation including investments, personnel and energy costs. The centers offer complementing application support teams but have no overarching user support structures.

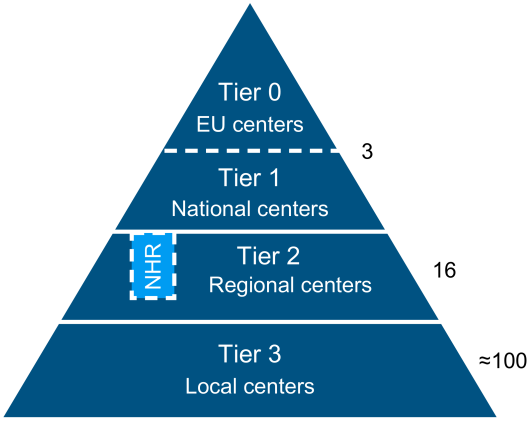

Figure 2.1: HPC pyramid characterizing the HPC landscape in Germany.

At the **Tier-2 level** there is a broad range of mid-sized HPC centers mostly at academic sites and centers serving dedicated user communities (e.g. German Climate Computing Center) or user societies (e.g. Max Planck Computing and Data Facility). Though, Tier-2 academic sites should serve users on a regional scale, funding has been highly local as only 50 % of the investment costs have been funded through DFG while all other costs had to be covered by local authorities. The only cross-state organizational body at Tier-2 is the North German Supercomputing Alliance (**HLRN**) founded in 2001 with 7 member states in northern Germany. Besides the joint funding for two Tier-2 centers (ZIB Berlin and GWDG Göttingen), HLRN also offers dedicated user support personnel in all member states. A nation-wide effort to coordinate HPC centers across the Tiers and to promote HPC as a strategic research activity is the **Gauss-Allianz e.V.** [2] (GA), which is a non-profit organisation. All Tier-2 centers, several Tier-3 sites and the GCS hold a GA membership. The GA aims

[1] `http://www.wissenschaftsrat.de/en/about.html`
[2] `https://gauss-allianz.de`

to improve the national and international visibility of German HPC research efforts, accomplishes scientific events, consults decision makers and provides recommendations concerning national HPC infrastructures. Its web page is a central hub that provides information about BMBF and DFG funded HPC programs (e.g. the DFG call under which ProPE project has been funded) and projects, and presents all membership centers in a coherent way. The annual GA HPC status conference offers a platform to present results from BMBF and DFG funded HPC research projects and discuss new HPC developments. Complementing the GA activities, the ZKI AK Supercomputing [3] is an interest group that primarily provides a platform for exchange of knowledge and experiences between HPC centers across all Tiers aiming mostly at operational and administrative issues. It organizes two annual meetings at member sites.

In addition some states have established state-wide HPC competence or coordination networks: bwHPC in Baden-Würtemberg [4], HKHLR in Hessen [5], KONWIHR in Bavaria [6], and most recently HPC.NRW [7] in North-Rhine Westphalia. The scope of these efforts is very different. The **bwHPC** association has dedicated hardware resources, which are distributed across state universities and provides central infrastructures (high speed networks, IDM, documentation platforms) for state-wide access. The involved Tier-2/3 HPC centers focus their efforts on specific application communities. A different approach is implemented by the Bavarian competence network for HPC (**KONWIHR**) which fosters research and software projects for efficient utilization of modern HPC systems for more than two decades. The KONWIHR projects receive dedicated funding and special support from the two leading HPC centers in Bavaria (LRZ Garching and Erlangen Regional Computing Center). The **HPC.NRW** and **HKHLR** offer and coordinate state-wide training activities and foster HPC user support of relevant Tier-2 and Tier-3 centers. Figure 2.2 gives an overview of currently existing HPC federations and the embedding of the ProPE partners.

Figure 2.2: Overview German HPC landscape. Shown are locations of Tier-0/1 and Tier-2 locations. ProPE partners are marked with red dots. The five regional networks are colored.

In April 2015 the German Council of Science and Humanities recommended the implementation and long-term financing of a nationwide coordinated Tier-2 infrastructure for High Performance Computing (Nationales Hoch- und Höchstleistungsrechnen, NHR) at academic institution in Germany [29]. In January 2020 the DFG has issued a call for "Admission of Computing Centres to the **NHR Alliance**" [8] which seeks for "suitable computing centres with complementary profiles". In this context, the "NHR Alliance shall provide resources

[3] https://www.zki.de/ueber-den-zki/vereinsstruktur/supercomputing/
[4] https://www.bwhpc.de
[5] https://www.hkhlr.de
[6] https://www.konwihr.de
[7] https://hpc.dh.nrw
[8] https://www.dfg.de/foerderung/info_wissenschaft/2020/info_wissenschaft_20_03/index.html

and nationwide services with respect to scientific computing and thus meet the universities' nationwide need for high-performance computing". The program will provide complete funding for each center over a timeframe of 10 years including personnel and operational costs. Centers in the NHR Alliance will not only coordinate their hardware investments but are also expected to collaborate in HPC user support and training and shall promote "trans regional and interdisciplinary collaborations and cooperation". This will allow scientists from all over Germany to access NHR's systems and benefit from the combined know-how of the NHR Alliance in the fields of performance engineering, HPC-applications and numerical methods. A nationwide HPC environment is to be created hosting high-quality HPC systems, distributed HPC know-how and internationally recognized HPC research. The NHR Alliance is expected to start operations in January 2021 with eight to ten NHR centers distributed across Germany.

The funding and operational structures of smaller academic HPC centers and HPC systems at the research group levels (**Tier-3**) will continue. With the strengthening of the academic Tier-2 centers through the NHR program, the NHR centers may act as regional hubs for accessing the NHR resources and providing and distributing advanced HPC expertise into Tier-3 level.

Chapter 3

Requirements for a national Performance Engineering infrastructure

This white paper discusses basic components and central structures of a potential nationwide Performance Engineering (PE) infrastructure at academic HPC centers. The proposed concept aims to leverage distributed HPC expert knowledge to provide high-quality problem-specific user support at all participating centers. A blueprint of this infrastructure has been established across the three participating centers in the ProPE project. These centers can cover many relevant HPC topics: RWTH Aachen focuses on identifying scalability issues and optimizing OpenMP codes, FAU Erlangen is known for its node-level PE activities for CPUs and GPUs, and TU Dresden has a strong focus on I/O optimization and visual performance analysis tools (see B.1 for a comprehensive overview of the competence fields). All centers have a long-standing track record in both daily HPC user support and research contributions in their competence fields. They are furthermore well known for related training and teaching activities.

The term *Performance Engineering* is often used in HPC for any activity to improve the time to solution for a given code or problem. This includes trial-and-error optimization approaches, where textbook code transformations are tested for improved runtime without gaining a deeper understanding of the underlying performance problems. Performance Engineering as done by the ProPE partners goes substantially beyond this: **Performance Engineering is a well-defined, structured process to identify the relevant performance bottlenecks and then derive appropriate code changes** or other measures to improve the resource efficiency of programs. Various approaches are available in ProPE to identify the performance bottleneck and understand its implications: threshold analysis, performance patterns and performance models can be used depending on the requirements of the analyzed problem. The profound bottleneck analysis aims for a deeper understanding of the interaction between hardware and software for the application code, solver or kernel operation at hand, and the insights reach beyond the optimization effort for a specific HPC system and parameter setting. The underlying **core process of our PE approach** consists of three steps: **performance measurement and analysis, bottleneck identification (testing thresholds or performance patterns, establishing performance models) and performance optimization**. Finally, code optimization in our understanding is any measure to improve time to solution for a given code including code transformations, parallelization, improved compiler settings or optimized execution parameter settings. While our PE approach does not explicitly include algorithmic optimizations, it often provides guidelines about basic features of alternative algorithms to be considered; choosing an algorithm with higher computational intensity is a typical example. As the centers are embedded into an academic environment, researchers from applied math can be involved in a PE case if code optimization did not deliver the required improvements or indicate some potential for choosing alternative algorithms.

An important component of any PE activity is the **identification of (potential) PE cases**. First, these can be triggered **by users or software developers** who have identified a performance problem themselves or require faster time to solution. The users/developers contact the local PE engineers through the established support structures. These PE cases often require in-depth analysis and the users/developers will be strongly involved in the PE activity. The second alternative is the identification of potential performance problems **by active performance monitoring** of the HPC resources. With appropriate monitoring of the hardware utilization in place, the HPC support team can use bottleneck identification procedures from the PE process (thresholds or performance patterns) to pinpoint badly performing applications, e.g. which only a fraction of the allocated nodes or which show very low hardware utilization. Typically most of these severe performance problems can

easily be solved after the performance engineer contacts the user, e.g. by adapting job scripts or choosing better compiler (options). For users with large allocations further in-depth analysis is offered. Please note, that the hardware performance monitoring data should also be made available to the users. This raises the awareness for hardware efficiency and may encourage them to investigate if actual performance behavior is inline with their expectations. Thus, a system-wide, continuous job-specific hardware performance monitoring which provides reliable and relevant utilization metrics (such as main memory bandwidth, FLOP-rates, instruction throughputs, vectorization ratios, IO-rates or communication frequencies/volumes) is a foundation of any PE-oriented user support. Beyond the application of PE for their local users all three centers regularly contribute their PE expertise to research projects often in the context of a co-design approach (e.g. the three centers have been involved in five DFG SPPEXA projects) or within state-wide HPC networks such as KONWHIR (FAU) or HPC.NRW (RWTH).

Setting up a nation-wide PE infrastructure covering all relevant tier-2/3 center faces the challenge of very different PE knowledge levels of the HPC support people involved in this infrastructure as well as the application scientists and code developers accessing this infrastructure. **Knowledge documentation and transfer** as well as **training** of support personnel and users/developers is a central issue to keep the coordination effort between the centers low and minimize the number of PE cases which cannot be resolved locally but have to be escalated to the central infrastructure. To achieve this goal a central documentation platform process needs to be established and a structured nation-wide training program is required that is tailored to user groups, knowledge levels and application domains.

At the start of the ProPE project each participating center has performed such PE activities by local staff (performance engineers) for local users for a long time. Further they had implemented different PE approaches, user support structures, and HPC system environments.The task of the ProPE project was to identify and implement components for a distributed PE infrastructure which integrates existing local support structures and PE expertises into a nation-wide infrastructure while maintaining local administrative structures. Central goals of this infrastructure are continuous knowledge exchange, coordinated knowledge transfer and training, coherent PE processes allowing for an easy transfer of PE cases to remote centers, and a joint job-specific monitoring strategy with a coordinated format to exchange job performance data.

To this end we have identified and implemented the following components:

- A systematic **performance engineering process** following scientific practices.
- Robust and simple processes that cover the required use cases of a **distributed support infrastructure**.
- A system-wide **job-specific performance monitoring infrastructure** at each site to identify both pathological jobs and those with high optimization potential.
- A **central knowledge base** containing documentation of all relevant PE core activities and all required specifications
- A structured **collection of training activities** within the German HPC landscape categorized, e.g., by content, knowledge levels and target groups.

Further, a common classification and nomenclature is required for the following areas:

- **Performance metrics**: Define common metrics that characterize application performance.
- **Job Classes**: Group jobs by their performance characteristics and hardware utilization.
- **Target groups**: Group of persons by task and position.
- **Knowledge levels**: Indicate the degrees of preliminary knowledge.

Above components and requirements are sufficient from a technical point of view. Still from a governance point of view the financial investment of a performance engineering infrastructure has to be justified. The infrastructure and processes used should ideally either provide metrics for increased scientific productivity (e.g. measured in publications made possible by the use of an HPC system) or, with a finer granularity, the ratio of money saved by an activity to the investment consumed by that activity. To this end, we have further proposed a **cost model** to measure and argue for the effectiveness of the performance engineering process.

Chapter 4

Components of a national Performance Engineering infrastructure

4.1 ProPE Performance Engineering Process

4.1.1 Iterative scientific process for performance engineering

As discussed in Chapter 2 we consider **Performance Engineering** as a well-defined, structured process to improve the resource efficiency of application programs. The ProPE project offers a generic PE process based on scientific principles (shown in Section 4.1.1), which can be employed in support actions and training activities. This process can be carried out in different levels of detail, and we describe two variants in Sections 4.1.3 and 4.1.4.

Within the regular support activities, the PE process can be initiated in different ways: (1) detection of performance issues via job-specific performance monitoring, (2) a user's own performance analysis, (3) a user request for more effective use of a resource allotment, or (4) a user request for performance models of their own code in order to get better insight into performance issues. Irrespective of the path that was taken, the PE process follows the cyclic pattern schematically shown in Figure 4.1. Before it can begin, however, it is pivotal to define a relevant test case that allows for quick turnaround time while benchmarking but still reflects production performance behavior. The basic iterative performance engineering procedure requires the following steps:

1. Acquire a runtime profile to determine which parts of the code require significant time to execute. Even this first step may impose a significant obstacle since the profile of complex applications my be quite "flat," i.e., the runtime may be spread rather evenly across many small loops or routines. Even worse, inlining may distort the view on the profile since hotspot functions can disappear by the compiler inserting their code at the call site. Standard tools may or may not handle this situation gracefully. It is a recom-

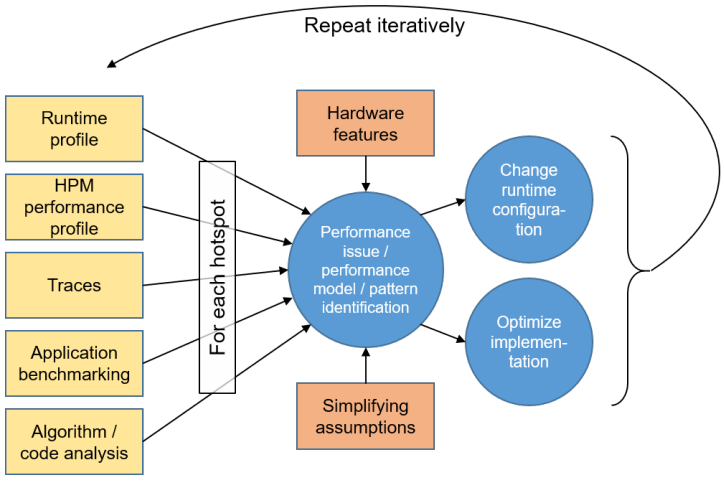

Figure 4.1: Simplified overview of the PE process. Note that the construction of a full performance model or the identification of a pattern for each hotspot is optional and also not always possible. For each specific hotspot, the cycle terminates as soon as no relevant performance improvements can be attained any more. HPM stands for Hardware Performance Monitoring using hardware performance counters.

mended practice to endow complex applications with explicit timing functions so that at least a coarse overview is readily available. Note also that the runtime profile for the parallel version of an application may differ significantly from the sequential version because node-level bottlenecks augment the impact of bandwidth-bound code regions in the parallel case.

2. For all parts of the code that consume a significant fraction of the total runtime (hot spots), a **performance analysis** must be done. The level of detail may vary here since there is a wide spectrum of possible performance issues, not all of which require all the data to be addressed. In the following we list the possible components of the analysis step and comment on their applicability and utility:

   - *Static code analysis*. This step concentrates on the high-level aspects of the code. It tries to answer questions like "Which execution resources are needed (data structures, execution units,...)?", "Does the compiler have enough information to produce 'good' assembly?", "Are there any basic problems with data access patterns?", etc. While it usually starts with the high-level language, later passes of the PE cycle may consider assembly code as well. Note that the data acquired here does not require to run the code (although the questions addressed can certainly be motivated by results from the steps below).
   - *Application benchmarking*. A single performance or runtime measurement per hotspot is rarely sufficient to acquire all data necessary to form a solid hypothesis about performance issues. Data about the behavior with changing runtime parameters such as number of cores, threads, and processes, thread-core affinity, problem size, problem geometry, etc., may be pivotal to understand performance issues like bandwidth limitations, cache reuse, and communication overhead. This data is especially well suited for forming first hypotheses about performance patterns (see below). In complex applications it may not be straightforward to investigate the impact of the influence factors separately for each hotspot. Then it can be advisable to extract loops or sub-algorithms into **proxy apps** or **mini-apps**, i.e., small, stand-alone programs that are easier to handle than the full application.
   - *Acquire and analyze runtime traces*. Especially in cases where a significant part of the runtime is not spent in the numerical code but in runtime libraries (such as MPI or OpenMP), the chronology of events is of major interest. There are numerous tracing tools that can visualize the interaction of the user's numerical code with the parallel programming library. Timeline and aggregated views allow high-level insights into issues like communication hotspots, load imbalance, and desynchronization.
   - *Hardware performance counter profiling*. Hardware events allow for a thorough investigation of bottleneck behavior on the core and node level. Raw counts and derived metrics such as memory and cache bandwidth, load/store to flop ratios, IPC, vectorization type and ratios, etc., can substantiate a hypothesis about a performance issue and underpin or validate a performance model.
   - *Operating system level data sources*. This involves for example information about file and network IO, but also things like load, memory consumption and memory organisation, as well as the state of clock frequency settings and other system settings.

   One should not fail to mention that an experienced performance engineer might skip one or more of these activities because the available evidence is sufficient to arrive at a solid conclusion.

3. Based on the data acquired by above activities and considering the machine properties, a hypothesis about potential performance issues is formed. This hypothesis should be connected with a quantitative performance expectation or even a full-fledged **performance model** (like Roofline or ECM on the node level, or a combination with a communication model in the highly parallel case). **Performance patterns** are a helpful instrument in categorizing performance issues if a quantitative model is not desired or possible (see Section 4.1.4 below). Finally, a restricted set of hardware performance counter data can be used for a **threshold analysis** (see Section 4.1.3).

4. The performance expectation, pattern, or threshold analysis points directly to possible optimization actions (e.g., appropriate blocking techniques, parallelization strategy, efficient data layouts, distribution of workload, or changing the runtime configuration). It should be noted that the performance profile can change considerably if hotspot code profits from optimizations.

Regardless of the level of detail, above steps must be repeated several times for all hotspots until a required or sufficient performance level is achieved. This makes it is an *iterative* process, and it is the task of a PE consultant

to determine when to stop. The level of detail is adapted to the importance of the project, ranging from drafting simple performance expectations to detailed modeling and optimization. After optimization it must be ensured that the optimized variants (within mini-apps or not) are used in regular production and exhibit the expected efficiency boost.

4.1.2 Required skills

Several specific skills and areas of knowledge that are not common among software engineers are required to perform the above iterative process in full detail:

- **Application benchmarking**: This includes defining relevant test cases, ensuring reliable timing and performance measurements, controlling and monitoring thread and data affinity settings, and managing the system options influencing performance (e.g., turbo mode, cache coherence options, and ccNUMA settings).
- **Microbenchmarking**: Microbenchmarking is an indispensable tool in performance engineering and serves a variety of purposes: it provides upper performance limits for various resources (e.g., memory bandwidth), it helps to find performance bugs in architectures (e.g., badly designed cache hierarchies), it can uncover undocumented but relevant processor properties (e.g., non-overlapping caches), and it can quantify the cost of programming model constructs or runtime environments (e.g., OpenMP barrier latency). In the context of PE, microbenchmarking is not a black box but a tool to create insight about hardware-software interaction.
- **Runtime profiling**: This may involve more than just running a profiling tool, since compilers and instrumentation may interact with the profile in unwanted ways. Instead of instrumentation, *sampling* may be a more effective way to obtain this data, but statistical variation has to be kept under control.
- **Performance counter profiling**: A plethora of tools exist that support hardware performance counting, but taking the data in the appropriate way and interpreting the numbers correctly requires considerable experience. Moreover, event counting is sometimes unreliable, e.g., counters may give wrong results, so it is pivotal to be able to separate out nonsensical data.
- **Performance expectation**: If possible, a performance expectation or analytic model is the reliable way to identify optimization strategies. Although some tools exist to aid in the construction of analytic performance models (especially for steady-state loops), there are cases in which such models fail or are just too cumbersome to construct, too complex to set up, or inapplicable because their prerequisites are not met (such as Roofline or ECM if the loop is too short to amortize startup and wind-down latencies). Black-box models (curve fitting, machine learning) may be an alternative in very complex situations and where hidden scalability bugs are suspected. Whatever the approach, performance modeling is a science in itself and not an automatic tool that always gives correct answers. It is thus not an activity that can be carried out by inexperienced support staff.
- **Performance optimizations**: While there are generic optimization techniques such as work reduction, tiling, vectorization, avoiding communication and synchronization, etc., the details and actual implementation are specific to the application. A code optimization that requires significant effort should thus only be done if there is a good indication that it will pay off. Again, this decision requires experience.

In addition to these specific skills, a performance engineer should have knowledge of application algorithms, processor and parallel computer architecture, operating system internals, programming models and their implementation, and the behavior and options of compilers. To make things even more complicated, many of these areas are moving targets where problems depend on the version of the operating system or compiler, not to mention features introduced with new processor architectures. Most of these skills are not part of a standard software engineering curriculum or even a parallel programming course. Therefore continuous training of all participating groups and a shared knowledge base is essential for success.

4.1.3 Threshold-based performance analysis

As described above, the analysis of benchmark results and/or hardware performance metrics is a difficult initial step in any PE effort and requires considerable experience. The ProPE threshold-based approach supports a

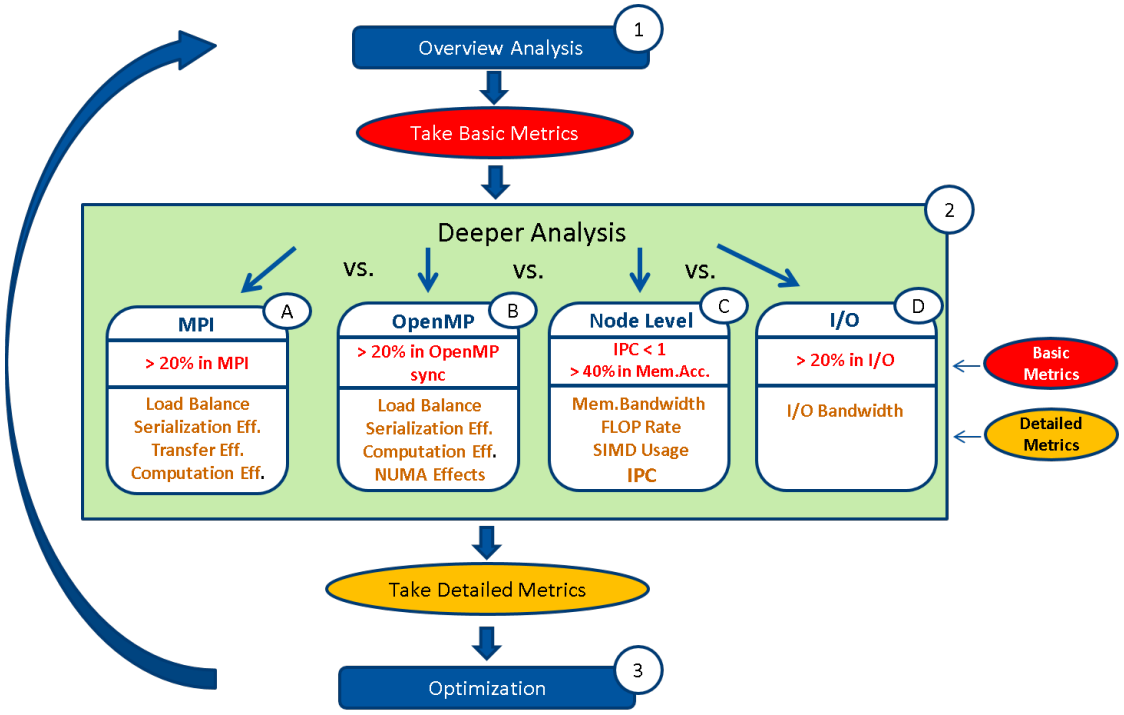

Figure 4.2: Overview of the threshold-based PE process. Initially, four basic metrics are obtained (step 1, indicated in red) and compared to fixed thresholds. If one (or more) of the four thresholds has been exceeded, a set of detailed metrics is acquired, ideally leading to a possible solution of the performance problem.

simplified version of the full PE cycle. It is based on work of the EU POP Center of Excellence[1] project, in which the ProPE project partner RWTH is an active member. Together with POP project members the existing approach was improved and extended with an in-depth node-level coverage. It is based on thresholds for easy-to-obtain metrics and is useful for a rough initial performance analysis that is also suitable for beginners. After this first step, a more detailed pattern-based analysis can be employed, if necessary, by experienced developers.

The threshold-based analysis considers four performance aspects: MPI issues, OpenMP issues, node-level performance, and I/O. See Figure 4.2 for an overview. The process consists of three repetitive steps: Overview analysis, detailed analysis of one of the performance aspects, and optimization of this performance aspect. Each aspect provides a set of additional metrics that can be used to identify and optimize performance issues. Once one of the performance aspects has been optimized, the process is started again from the beginning to focus on optimizing the next performance issue. In the following we describe the three steps in more detail.

Overview analysis

The first step is to perform a rough performance analysis for all four aspects using lightweight analysis tools such as Intel Application Performance Snapshot (free) or Arm Performance Reports (commercial). Both tools do not require recompilation and linking of the program or special settings of the tool. The tools provide overall metrics like the amount of time spent on MPI operations, OpenMP synchronization, number of instructions executed per cycle, etc. For each metric, a threshold is defined by experts in the relevant HPC area or by the analysis tools themselves. Each metric and its threshold identify a performance bottleneck for the application. This performance aspect should then be analyzed in detail and optimized in the next steps of the process. These are the metrics and their thresholds (letters correspond to the labels in Figure 4.2):

A. If more than 20% of the CPU time is spent in MPI operations, the communication structure of the code should be analyzed.

B. If more than 20% of the CPU time was spent in OpenMP synchronization (or, more generally, the OpenMP runtime library), the OpenMP parallelization should be analyzed.

[1] `https://pop-coe.eu/`

C. If the number of executed instructions per cycle (IPC) is smaller than 1 or more than 40% of the CPU time is spent in memory accesses, the data access properties of the code at the node level should be analyzed.

D. If more than 20% of the time was spent in I/O read and write operations, the I/O behavior of the application should be analyzed.

In case more than one threshold is exceeded, the most severe should be handled first.

Deeper Analysis

The overview analysis identifies a performance aspect that should be analyzed in detail. The detailed threshold analysis requires key metrics to be measured for selected application regions and for the most severely exceeded threshold (A. . . D above). Appendix A.2 lists the detailed metrics and which thresholds should be observed.

4.1.4 Performance analysis with patterns

The pattern-based approach provides a different angle to performance engineering at an intermediate level which does not reach as deep into the details of hardware-software interaction as the full PE cycle. It was initially presented in 2012 as a basic concept for describing and classifying typical performance motifs [22]. Currently, the available pattern set covers node-level issues only; it can be combined with the threshold analysis for highly parallel cases. The pattern analysis can only be used for a homogeneous ("steady-state") part of the code, usually the innermost loop nests that make up a substantial part of the runtime. This sets it apart from the threshold-based approach, which is used on the whole application or at least on larger parts such as entire solver components.

The process has the same structure as the generic PE process shown in Figure 4.1 but does not usually require to set up a performance model (although this is possible as an extension). Instead, the input data from the first stage (runtime profile, traces, etc.) is used to postulate a **performance pattern** or **motif**, which encapsulates the performance-limiting aspects of hardware-software interaction for the chosen hotspot. In case of doubt, the pattern hypothesis can be **validated** with additional measurements (e.g., more in-depth performance counter analysis or application benchmarking). Finally, with the correct pattern identified, code optimizations can be employed to mitigate the performance problem (if there is any – not all patterns actually point to problems).

A detailed list of performance patterns (extended from the original paper) can be found in the HPC-Wiki Portal[2]. The following table gives an overview of the most important patterns:

Name	Description
ALU Saturation	performance limitation caused by fully utilizing a functional unit inside a CPU core
Instruction Overhead	for a piece of high-level code, the compiler outputs excess instructions
Excess data volume	data is transferred more often than required within the memory hierarchy
Bandwidth Saturation	performance limitation caused by fully utilizing a shared data path
Bad data placement	performance limitation caused by data residing in remote locations with higher access times
Load Imbalance	problem when work is not equally distributed over all processing units
Synchronization overhead	performance limitation caused by frequent synchronization calls in parallel environments
Expensive instructions	use expensive instructions although there may be cheaper solutions
Inefficient instructions	usage of one kind of instructions although there exist more effective ones, e.g., scalar vs. vectorized FP instructions

Most of these patterns are directly connected to optimization strategies; for instance, if the *ALU Saturation* pattern applies, the only way to improve the performance is to reduce the amount of instructions executed by the bottlenecked execution unit. In case of bandwidth saturation, reducing the amount of data transferred over the data path is paramount.

[2] `https://hpc-wiki.info/hpc/Performance_Pattern_List`

4.1.5 The Performance Logbook

The documentation of activities, settings, and results is crucial for any performance engineering effort. It helps to keep track of changes, to make deterministic progress, and to decide which steps are up next. In addition, it contributes to the exchange of insights and knowledge and prevents redundant activities for all participants in the application lifecycle. We have created a performance log template that already contains placeholders for all important activities during performance engineering. The logbook consists of a central text document in markdown format, folders for referenced numbers, settings, and results, and everything else that should be tracked during the process. To use a markdown text file ensures that anyone can view and edit the file with a text editor. No special tools or software are required. As a further advantage, github and gitlab automatically convert markdown files to HTML and allow editing in the web browser. The inclusion of the performance log in revision control makes it possible to reference both code changes and PE activities that motivate specific code changes.

The logbook is intended for data centers running customer projects, but can be useful for anyone doing a performance project. The logbook comprises a preamble, several analysis sessions, and a summary. Each analysis session consists of one or more benchmarking blocks. There are five pre-configured activities within a benchmarking block:

- **Runtime Profile**: Create, analyze, and discuss a runtime profile
- **Performance Counter Profile**: Create, analyze and discuss a performance counter profile of any kind
- **Result**: Application benchmarking runs
- **Analysis**: Compilation of analysis, observations, and further planning based on previous activities
- **Optimization**: Record of attempted performance optimizations (e.g., changing runtime settings or changing the code)

This structure is certainly not cast in stone and can be adapted to the case at hand. It is, however, recommended to keep the overall structure in order to give performance engineers some common ground when collaborating on projects.

The performance log template with examples is available as Open Source on GitHub [8].

4.1.6 File format standards for Job Meta- and Metric-data

A common format for job metadata as well as metric time series data is useful to exchange job data between sites, for research and analysis purposes, and as a robust way for archiving job performance data. We have developed a JSON text file schema for job metadata and metric data. The format also contains basic statistics about jobs and nodes. The file formats are accompanied by a scalable directory hierarchy specification that can accommodate millions of jobs and an SQLite database schema with an access wrapper script. The JSON schema, helper scripts, and documentation is available at GitHub [7].

4.1.7 The broader context of performance engineering

At the beginning of the ProPE project, a focus was put on performance engineering as a part of software development. However, as HPC is on the way to become scientific mainstream, only a minority of all HPC users today are also software developers. Most users employ either *community codes* (e.g., OpenFoam or Gromacs), commercial software, or production-grade applications that were developed at a chair or institute. Therefore, it is not effective to focus on software development only; other areas such as system configuration, execution environments, and workflow management are at least as important to increase the efficient use of the system. The term *performance engineering* should be seen in a much broader context, encompassing all activities that affect the resource efficiency of application programs.

The following influence factors determine the execution efficiency of an application code on an HPC system. We also give some specific examples of aspects to look out for:

- **Choice of algorithm(s)**: Is the algorithm the best solution to the problem at hand? Aspects: computational complexity, resource complexity, flexibility.
- **Software implementation**: Is the code written in a way that at least allows for efficient execution? Aspects: Data structures, abstractions, parallelization approach, communication strategies.

- **Code generation**: Does the compiler produce efficient code? Aspects: abstractions, data structures.
- **System configuration**: Are system tuning knobs exploited that allow for improved application performance? Aspects: clock speed, huge pages, NUMA balancing, Cluster-on-Die/Sub-NUMA Clustering, hardware prefetchers, memory configuration.
- **Execution environment**: Are affinity choices, amount of resources, etc., adapted to the performance properties of the software? Aspects: Pinning, number of threads/processes per contention domain, subdomain mapping.

A software developer will consider all of these points. Someone who only installs the software focuses on the last two or three points. However, this can make a big difference in performance and also requires the use of performance engineering skills for performance analysis and application benchmarking. After all, an application user only considers the last two or even just the last point. Again, application benchmarking and performance profiling are essential to uncover performance opportunities here. This illustrates the importance of documenting PE activities, insights, and performance results, and sharing this information with all parties involved in the application life cycle.

4.2 Distributed Support Infrastructure

The need and desire for excellent science and research is constantly increasing. In an age in which the number of data and their processing are growing ever faster and more complex, appropriate resources and environments are needed to provide science and research with the appropriate infrastructure for calculation. However, the mere provision of resources is not sufficient without the appropriate and competent support from specialists in the system environments.

In order to satisfy the desire for excellent science and research, it is necessary to provide users with the necessary HPC infrastructure. The same applies to competent and specialized support in this field.

To enable efficient high performance computing for more excellent research it is crucial to introduce potential users of different scientific disciplines to the specific systems. Hardware specific surroundings, code optimization and performance engineering are those areas that are important for the implementation and achievement of research goals, especially for scientists in many research areas. A corresponding support infrastructure is therefore necessary as well. The goal is to help users solve problems with competent and specialized support and expertise.

Since the expertise differs in many HPC centers in Germany, competence and expertise oriented support, independent of the users and centers' locations, is of great additional value for science and research. Inquiries therefore do not need to stay unresolved; instead, the targeted use of expertise on remote sites is of great help and use.

The following parts of this white paper describe how the computing centers involved in the ProPE project are structured regarding their level support and how the corresponding expertise at the participating centers is defined. The selected methods as well as the approach of the development of the support infrastructure itself are displayed in the following parts. Focus on these next paragraphs will be laid on the developed supra regional support infrastructure and its processes, which are displayed and elaborated with regards to the process map. When designing and conceiving a site comprehensive support-service of this quality, it is not only sufficient to make optimum use of the procedural and infrastructural conditions. For this reason, it is also important to meet the needs of the users and to keep the added value for the target group as high as possible. In order to meet these needs sustainable, quantitative surveys have also been designed and developed within the framework of the Pro PE project. With the gathering of user feedback, we are aiming to ensure the quality of cross-center support and to evaluate processes accordingly. In the course of this, the results of a user satisfaction survey enable demand-oriented adaptation and optimization of the support-process in case needs change or comparable.

4.2.1 HPC Support Structures on the Participating Sites

The aim of this work is to develop, describe and establish processes and structures, which enable an efficient, sustainable and valuable multi-level user support in the field of HPC, between the participating computing centers in this project. The cross-site support structure of the participating HPC centers and experts aims also at an optimum use of resources across sites, both technical and human. The existing local support structures therefore need to be linked together. Here, we describe a regular Tier2-Support-Infrastructure. However, the

organization and standardization of the respective support levels poses challenge for the data centers involved, which are not organized in support levels.

First Level Support

In this context, data centers need to rely on an established first level support and the complementing specialized second levels. These include the HPC second level among all sites, which have been mentioned before. The customer contact is primarily regulated via the service desks in the first level. There, reaction times are maintained, initial solutions are provided and standard information is obtained from the users for the second level, if necessary. The first level support primarily includes requests relating to access authorizations and login as well as batch submission and availability of resources and status information for the respective systems. The forwarding to the second level HPC support is also carried out by the first levels.

Second Level Support

In the project's context, second level HPC support at computing centers needs to consist of specialists in the fields of HPC. The HPC staff may be both local and external employees. The second level is able to answer regular standard HPC service requests and has an advanced HPC knowledge level. In case requests cannot be answered or solved in the second level HPC, a third level support is of need. The resident second level categorizes the user request to the third level HPC support. However, this is no longer located on the home site.

Third Level

Level 3 HPC support (Third Level) consists of both local and external HPC experts in this context. The HPC experts in this particular kind of third level are personnel who support the second level support at the home site, but who step into the third level support function if required. In the ProPE-project, we have tested the supra regional support with three levels to make optimum use of expertise and resources.

4.2.2 HPC Expertise on the Participating Sites

In order to establish an expertise-oriented multi-level support structure, it is crucial to define expertise and focal points for the precise and efficient processing of customer inquiries. Experts in the context of HPC and in the field of Performance Engineering have different focal points resulting from research, system surroundings or simply competence and knowledge regarding specific aspects in HPC. Here, we are looking at Performance Engineering experts on three sites. These are local or external domain experts in the field of computational sciences. The supporting experts in the third level are members of the HPC Competence Network and are available to the cooperation partners with their expertise. The computing centers with their individual expertise are marked on the HPC landscape and defined as high-level. In the framework of NHR, a further concretization will take place including the creation of centers of excellence.

Serving as blueprint, the ProPE project partners have developed, tested and established a sustainable cross-site HPC support with three support levels. The three computing centers involved, RRZE, IT Center and ZIH, are defined with regard to their respective expertise. In order to provide an insight into the respective expertise, we offer an overview of them in the appendix B.1. Based on the mentioned above and defined expertise of the participating sites, the development of a sustainable and distributed support infrastructure has been developed. The development of the infrastructure aims especially at sustainability and includes the possibility that the concept can be adapted in other support contexts. The following part of this section shows an overview of the development of the distributed support infrastructure in the context of the ProPE project.

4.2.3 Development of a Distributed Support Structure

In order to offer high-quality IT-infrastructure and support, which works efficiently both location-independent and site-wide, the necessary competences are indispensable, as well as the organization of such support structures. To provide HPC support across three support levels in three locations or more, processes need to be developed, tested and optimized. In the following section, we focus on the development and the concept for a multi-site support that serves not only users in the fields of research and science but also the national cooperation of university computing and research centers.

To provide a distributed and site-wide support infrastructure with several participating sites, proper and regulated communication between the centers must be ensured. In this context, clear-cut definitions are necessary, which define the extent of responsibilities and workflows for a quality customer support.

Testing Cross-Site Communication and Support

In a first phase of testing, an early concept of the support infrastructure has been examined. At its core, the participating sites processed customer inquiries across three support levels. It is therefore important that processes and work flows need to be defined from the beginning to avoid uncoordinated or double communication with support staff and users.

The following section shows in abbreviated form the course of the first test phase regarding communication, which lays the foundation for the later organized and defined concept of the desired and efficient support process. For this communication process, e-mail transmission via personal e-mail-accounts was selected and no ticketing tool was used. With the help of voluntary "friendly users", we simulated support requests in order to see how communication and problem solving in the context of PE takes place. It followed the main idea that customer inquiries are processed accordingly to established structures starting on one site, hereinafter referred to as home site. While customer inquiries have been processed accordingly to established structures up to the 2nd-level at the respective computing center, the 2nd-level supporter faced the challenge of getting the cross site support as 3rd-level involved.

Here, we relied on a process that had already proven itself with regard to the external support by the 2nd-level *sciebo*, which is located in Münster and thus represents an external support level. In this case, the specialist department with its expertise is not located at the home site. The process provides that requests related to sciebo can be handled to a large extent by the 1st-level support on site. However, in the case of queries that cannot be processed by the 1st-level, these queries, including all standard information, required by the specialist department to process inquiries, need to be forwarded by e-mail to the second level support sciebo. The e-mail, including original inquiry and standard information, is sent to the external second level in Münster from the ticket tool used by the 1st-level on site. This way the processing of the inquiry is sustainable documented and can be used for the further processing by all support staff in the respective support levels. For this purpose, the sciebo support in Münster has its own support e-mail address to which the respective inquiries are sent.

Testing Cross-Location Support

To start the cross-site support, the 2nd-level of the home site used their local ticket tool to contact the other centers by forwarding the original inquiry. This initial contact of the centers has already depicted an obstacle in terms of communication. Although the 2nd-level supporter of the home site contacted the other HPC specialists, the other 2nd-levels of the remote sites could not be informed about the processing and status of the inquiry. While one site started to process the inquiry by communicating with the customer by using different communication paths, i.e., various e-mail-addresses and telephone contacts, the other party did not document their progress or findings. This led to the assumption that the inquiry has not been processed for some time. In the mean time, other supporting specialists were dedicated to the customer inquiries. Therefore, communication took place via various electronic and telephone routes. This broad variation led to gaps of information and the documentation of those on all sites involved. The different sites contacted the customer by e-mail as well as by telephone in order to try to process the request and also to gather support-relevant information. Information, which may be necessary for supporters in the 2nd-level of the home site and the supporter functioning as a 3rd-level support for processing the request. Without the documentation of the previous processing steps, the support personnel involved lack knowledge of the processing status. This circumstance has prevented shared access to solutions found on the one hand, but also justifies the need to create structures that guarantee a quality-assured processing of customer inquiries.

A further advantage of the documentation of process and work steps is that supporters collect experience through the expertise of colleagues at other locations and thus benefit. In order to share insights and solutions, it is necessary to document processing steps that lead to solutions of the customer concern. This way, the idea came up to use a common ticket tool, which is used by all specialists in the individual 2nd-levels HPC who potentially work on the same requests. In the individual processing steps of the tickets, the solution approaches can be documented and then potentially collected in a common knowledge base to provide edited information in a sustainable manner for support and customers. The following section shows how we handled the experiences from the first test phase to develop an efficient concept, which has been tested in a second test phase.

Handling Communication and Inquiries

We mainly tested the operationalization of the technical requirements to ensure all comprehensible communication and documentation. The cooperating computing centers face the challenge of how to handle customer inquiries with two ticketing tools. One requirement is that employees have to transfer inquiries from their local ticketing tool to the one that is centrally provided and serves as communication base. Therefore, one challenge is that personnel involved might have to learn working with the technical aspects of a second ticket tool. Using one common ticketing tool is the quintessential aspect for this concept and serves as the main idea to provide a supra-regional support infrastructure. This concept also goes hand in hand with the definition of responsibility for the respective original ticket. For each opened ticket in the common ticket tool, a responsible person must be declared who observes the processing to avoid the following escalation scenarios, which ought to be considered:

- The customer request is not processed. This can be the case, for example, due to illness, holidays or insufficient resources.
- A solution to the concern cannot be found; the person in charge has the opportunity to consult another center.

To avoid these scenarios, we have agreed that the responsibility lies within the 2nd-level of the home site. Further agreements for the support process have been found and are presented below as a formal support process. Starting with the common basic requirements, such as the similar support level structure at the cooperating computing centers, to the solution of the request and documentation of the knowledge.

In this course, a completely new concept for the support was developed. The following part shows the developed concept for a distributed support infrastructure and its process map. The formal support process, which embodies the core process of the process map, displays the actual processing of a user request running through all three support levels.

4.2.4 The Process Map for a Multi-Tier Distributed Support Infrastructure

Process mapping allows communicating visually interdependence and conditions of important aspects regarding workflows. It provides and spotlights on important details without detailed verbalized descriptions. However, a formalization of the support process itself explains in detail specializations and instructions. Figure 4.3 provides an understanding of the management process, the core process and the underlying process, which represent interdependence and influences on the anticipated output, the distributed support infrastructure for HPC.

Management Process

The top layer of the process map depicts the management processes (MP). General functions of the MP are planning, organizing, leading and controlling. In the field of planning and deciding, the course of action is determined. Coordination of activities and resources, such as strategies aiming at visions and requirements. In the ProPE project, clear-cut goals regarding determined courses of actions and resources have been made and defined in the existing project proposal by the DFG. It served as a guideline while providing defined goals and tasks during the project and is therefore placed on the level of MP.

The requirements and work status derived from the DFG proposal was evaluated and discussed regularly at conferences. Regular exchange talks via telephone conferences and project meetings served to coordinate tasks at the cross-location data centers. Since the entire project has been organized into four work packages, plans and execution of those individual work packages have been discussed in, for example, weekly meetings on-site and also with the help of telecommunication or personal meetings across site. A well-organized coordination of individual tasks, responsibilities and communication of results and work statuses is necessary for a project like this. In conference calls and project meetings, off-site goals and decisions are reviewed, staff changes elaborated and responsibilities assigned.

Core Process

The heart of a process landscape is the core process itself (as displayed in Figure 4.3). It spotlights the support process for the customer support regarding a distributed HPC support with expertise on remote sites. It

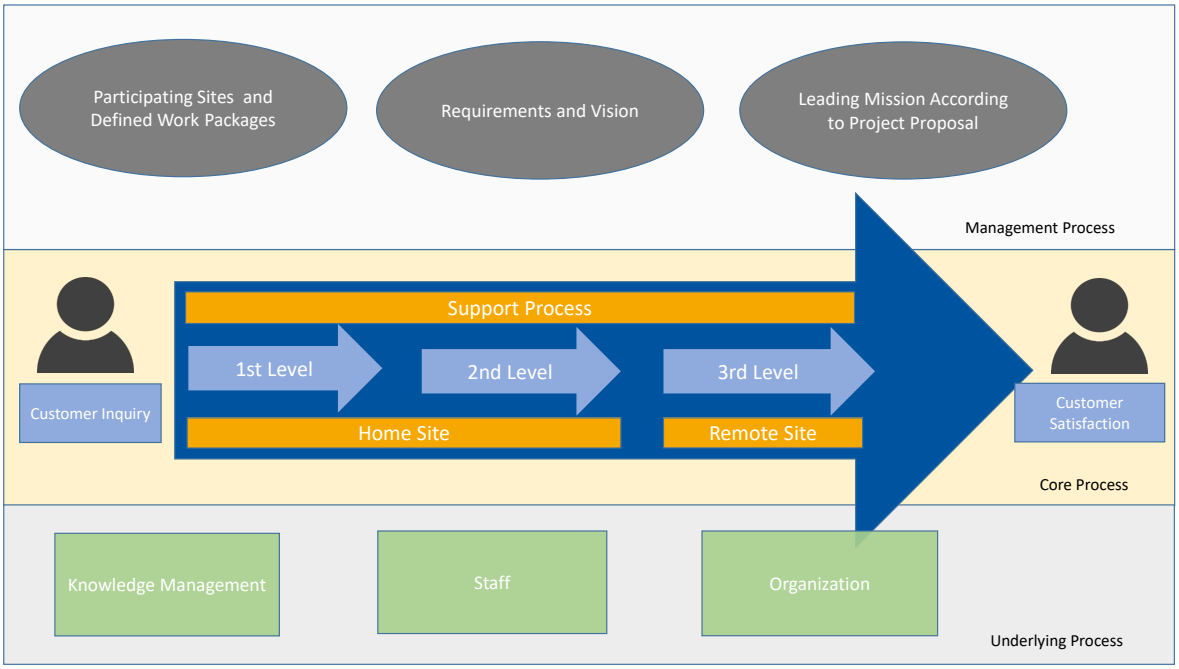

Figure 4.3: Process Landscape for ProPE: management, core, and underlying process.

visualizes the support process across three service levels. This is illustrated in the process map by showing the different relationships between the support levels and the sites. While the first and second level support are located at the home site, the third level support moves over to the remote site.

It is important that the second level of the home sites can take on the function of a third level support at any time, if a customer request exceeds the expertise of the second level and requires the defined competence field of the third level. The aim of the competence-oriented support is to offer the users the best possible support. Not only by a specific expert, but also by many experts at the remote sites. The result is the use of existing resources without the limiting factor of the location.

Within the scope of the project, the challenge was on the one hand to use a cross-center ticket tool and, on the other hand, to enable customer support in compliance with data protection regulations. Among other features, a common ticket tool offers the possibility to set up a shared ticketing queue to which all agents of the different sites have access. This simplifies the communication and processing of customer requests between the sites as well as the transfer of customer requests from the second to the third level. Therefore, a cross-center ticket instance was necessary for the implementation, which in our case was suspended by the Gauss Alliance. Here, the decision was made in favor of the OTRS ticket system, in which the “Performance Engineering” support queue was set up specifically for this purpose.

Similarly, it is important not only to establish a common and smooth workflow with an appropriate technical realization, but also to comply with data protection agreements. These privacy policies are especially important for the processing of personal data of users, regardless of which organization they belong to. While first and second level each remaining on the respective home site, the third level interacts across sites. Here, all second level agents on the participating sites inhibit the potential function of third level support agents. They enter the third level support function as soon as a support request arrives that matches the defined field of competence. At the latest in the third level, the supporters come into contact with personal data of users of other facilities. This includes not only the personal data, but also, for example, code that is passed on as part of the support request. This content must also be protected and the user must be informed that, in order to solve the request, this information may have to leave the home site and be passed on to third parties for further processing. The core process aims at the satisfaction of the customers whose support requests were made at the home location, whereby all possible resources are used in the network, even with the help of a third level on a remote site.

Underlying Process

The bottom layer shows the underlying processes, which support the overarching processes. The task of these is to provide the necessary resources for providing the best possible support. Within the framework of the ProPE

project, these are the following processes:

Knowledge Management Both, workflow and forwarding tickets to the home and remote sites are clearly defined by a process documentation. The aim is to maintain a transparent documentation of the ticket forwarding and processing and to ensure that the handling of personal data was in compliance with data protection regulations. Ultimately, the responsibilities have to be clearly defined and, thus, the quality standards for support have to be maintained even when a request is passed on. Following the test phase, the satisfaction of the users (here: support agents using OTRS) with regard to the functionality was recorded. Interviews were conducted by support agents and the data was evaluated anonymously. The findings were used to improve support structures for both users and agents. A first finding is that the ticket transfer shows a need for optimization. Especially since the different agents at the sites have to find their way with (likely) two different ticketing tools at the same time. The specific guidelines and workflows as well as templates for declarations of consent are documented in a wiki section of the versioning tool GitLab, accessible to all participants. These have proven themselves as necessary for correct ticket processing. Work steps and solutions are documented in the respective service units of the tickets. These serve as a potential source for subsequent entries into the HPC Wiki, which serves as user documentation. This way a sustainable handling of solutions and findings is ensured.

The next step remains in the establishment of regular in order to gain insights and measure the quality of support with the help of the gathering of user satisfaction.

Staff If the aim is to have a support structure based on expertise and competence orientation that functions across centers, it is also necessary to ensure the availability of human resources. In the project described here, there is at least one person at all locations who can assume the function of third level support. To maintain accessibility, a deputy should always be sought in order to be able to maintain reaction times and processing times.

Absences are regulated in such a way that, if unexpected, the respective team is informed at the home site and, if necessary, measures are taken at the home site. In the case of planned and foreseeable absences, these are communicated to the entire project team. Absences can also be stored in the ticket tool used so that this information is transparent for every staff member involved no matter which site he or she belongs to. Furthermore, at all locations, an absence note is stored in the personal mailbox informing about the foreseeable duration.

Organization Depending on the site, there are different internal regulations and cycles in which the respective exchange between employees takes place. This often also depends on the size of the team at the home site and the need for exchange. For example, the site-internal meeting cycle can take place weekly. As part of the ProPE project, it has proven helpful for teams from several work packages to exchange information on the respective processing status on a weekly basis. Nevertheless, it is necessary that both results and upcoming tasks are recorded in meeting minutes, so that they can be accessed if necessary.

Decisions can be traced in this way and in case of absences it is possible to inform oneself about the decisions, developments and upcoming as well as to be completed tasks. In the case of the ProPE project, for example, these protocols have proven to be helpful not only in maintaining the level of information, but also in retrospectively evaluating developments and decisions and, if necessary, optimizing them. Basically, it is necessary to make protocols available to all participants. We have used the versioning tool GitLab and used the wiki section to store and make protocols available. Depending on the amount of support required, the cycle can be adapted as required.

For coordination and mutual information in a cross-site context, however, it is advisable to hold telephone conferences at regular intervals. These not only serve to exchange information regarding technical implementations, but also, for example, to ensure the coordination of service requests in the ticket queue. In order to coordinate the responsibility for the ProPE queue in the ticket tool, a concept for the periodic assignment of responsibility is created in the form of a calendar concept. The responsibility is planned for the rest of the project duration and the respective schedule in form of a table is documented in the Wiki section of the GitLab-project and available to all members of the project. Appointment invitations for organizational purposes are sent in advance as ICS files. Every month a reminder e-mail is sent to the responsible agents and persons acting as coordinators.

A formalized description of the support process and verbalized workflow, which represents the above elaborated core process can be found in the appendix B.2.

4.2.5 Gathering User Feedback

With the goal to establish a distributed and sustainable support infrastructure to ensure customer support in the fields of academic science and research, it is important to take the users' satisfaction into account. An efficient and suitable way to measure user satisfaction is the qualitative survey using online questionnaires. The concept, developed in this work package envisages inviting users to the online survey twice a year by e-mail and drawing their attention to it. The participation in the user satisfaction survey is carried out anonymously and under condition that general data protection guidelines are adhered to.

With the results of the gathering of user satisfaction processes and services can be evaluated from another perspective. It is of high value for organizations to know about the actual user needs and therefore to act upon the results. This way, adaptations to processes and technical implementations can be evaluated and optimized if necessary. Another benefit of collecting user satisfaction data is the documentation and reporting of the support service. The collected data serves as a concrete and objective value, which can be measured to observer the quality development of the service. This is explicitly one of the defined goals, which were defined for the project to develop the support service sustainably and possibly quality-assured.

Insights and impressions on a first draft of a potential customer can be found in the appendix B.3.

4.2.6 Cost Model

While hardware acquisition costs were the prominent factor in the past, the increasing expenses for power consumption of recent HPC machinery have also raised the awareness of total ownership costs (TCO) in the HPC community. However, the importance of brainware [3], i.e., people tuning the code or supporting corresponding activities, is often neglected in today's HPC cost metrics. Nevertheless, these human efforts are crucial to improve resource efficiency on HPC systems and, thus, can even contribute to improved productivity of HPC centers [25]. To quantify the improvement of resource efficiency promised by applying a structured (support) process for performance engineering, we investigate suitable cost metrics that incorporate traditional costs as well as costs for human efforts. Here, the goal is to have a cost model that is as simple as possible and as complicated as necessary at the same time. Following this goal, we introduce our basic 3-element cost model that covers the three most relevant cost factors: hardware, energy, brainware. It can be used to evaluate the PE process activities and focuses on the user project's perspective.

A detailed breakdown of the model's components and the model's formula for calculation can be found in the appendix B.4. Furthermore, the appendix illustrates a case study that is based on these formulas.

Challenges of Brainware Metrics

Including a 'personal' metric like *human effort* in a cost model, we face several challenges in this project. First, effort needed to accomplish a PE task is dependent on numerous factors as, e.g., the HPC expert's preknowledge on a specific topic, or the available software and tools stack on an HPC system. However, the consideration of such impact factors is out of scope of this project and is subject to further research at RWTH Aachen University [26].

Second, the collection of such personal effort data must adhere to personnel regulations and also consider the European data protection regulation (DSGVO). One solution is complete anonymity when submitting effort data. Alternatively, effort data could only be assigned to a group of people by preventing the backtracing to a single person. Both approaches are often difficult in real world when an HPC expert works on a specific PE project. That is why we rely on voluntariness in providing effort data.

Third, tracking effort spent for PE activities usually means some additional effort for the PE expert. If the effort tracking cannot be easily integrated into their daily workflow, data collection is not realistic. Due to personal preferences, and different workflows or use cases, a single solution to effort tracking is not feasible. Therefore, we suggest one of the following approaches:

- Ticket tool: The distributed support infrastructure that we have established (Section 4.2) allows to denote (voluntary) effort data directly in the ticket assigned to an HPC expert. This kind of effort tracking is similar to keeping a classic developer's diary.
- PE logbook: The PE logbook that we have established (Section 4.1.5) can also be used to log time spent for the different PE activities. This is also the fashion of using a developer diary.
- EffortLog: Since traditional developer diaries often suffer from inaccurate data, another approach relies on the electronic developer diary *EffortLog* [26]. It is interval based and reminds the HPC expert to

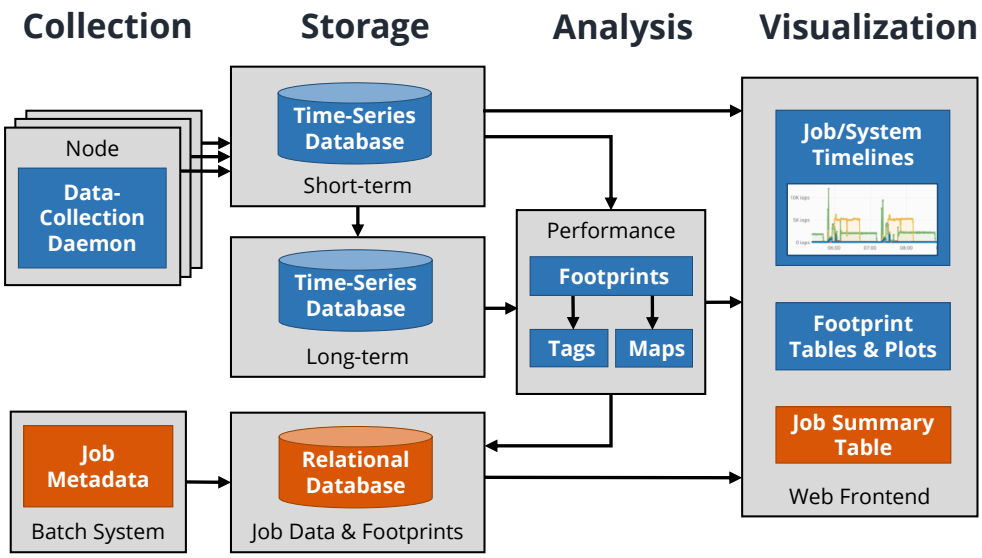

Figure 4.4: Monitoring Infrastructure

register previous done work and performance results. Then, effort data can automatically be derived (voluntarily and locally on the expert's computer, i.e., not server based).

Challenges of Hardware and Energy Values

Quantifying TCO across different HPC centers is highly political and their calculations are inherently difficult because of the complexity and the short investment cycle of HPC installations. In this project, we decided to average real cost numbers (for hardware and energy) across the three participating partners and numerous (non-accelerated) HPC systems (compare Table B.1). These values can serve as basis when other cost data is not available. Nevertheless, these average values must be updated regularly to keep up with technological changes. In the appendix B.4, the usage of these values are illustrated in a case study.

4.3 Performance Monitoring and Analysis

Nowadays, performance optimization is more or less an established procedure in HPC centers. It starts with a suspicion of a certain inefficiency, then first checks are made, before experts together with the users thoroughly analyze scalability, bottlenecks, etc., and hopefully find a better software solution or more suitable run-time parameters. To sustainably increase compute efficiency of HPC systems, we need to increase the awareness of efficiency at the users' side and automatically notify performance experts *in case of problems*.

Therefore, we propose an infrastructure for continuous monitoring and analysis (see Figure 4.4), which automatically characterizes HPC jobs, detects pathological performance behavior, and identifies optimization potential. The monitoring has an negligible overhead on the compute nodes and must neither influence nor limit the user programs. Job data can be visualized at runtime or post-mortem and is finally stored for long-term analysis.

4.3.1 An Infrastructure for System-Wide Job Monitoring

HPC jobs range from scripted sequential code to highly-scalable parallel programs. In addition, there is a large variety of script and programming languages. Therefore, the monitoring must cover all types of HPC jobs and operate independently of the executed applications. Data acquisition avoids any intrusive operations such as instrumentation of individual programs. A subsequent analysis of collected data enables the characterization and tagging of jobs.

There are numerous commercial, community, and site-specific tools for characterization of applications with respect to certain metrics. To enable a comprehensive analysis on all common HPC systems, we identified a set of metrics that covers compute performance, I/O utilization, and network traffic (see 4.3.2). The metrics

Metric	Proposed Name	Data Source
CPU		
Usage	cpu_usage	/proc/stat
Main Memory Utilization	mem_used	/proc/meminfo
IPC	ipc	LIKWID
FLOPS (SP-normalized)	flops_any	LIKWID
Main Memory Bandwidth	mem_bw	LIKWID
Power Consumption	rapl_power	LIKWID
Network Bandwidth	net_bw	
Infiniband Bandwidth	ib_bw	/sys/class/infiniband/...
Ethernet Bandwidth	eth_bw	/sys/class/net/eth*/...
I/O Bandwidth & Metadata		
Local Disk	read_bw, write_bw	/proc/diskstats
Lustre	read_bw, write_bw & open, close, create, seek, fsync, read_requests, write_requests	/proc/fs/lustre/llite/*/stats
GPU		NVML
Usage	gpu_used	
Memory Utilization	gpu_mem_used	
Power Consumption	gpu_power	
Temperature	gpu_temperature	

Table 4.1: Monitored Metrics

have been selected with the criteria to be available on most systems as well as their value for a reasonable analysis of the job performance. They are collected on each compute node.

The proposed job monitoring and analysis infrastructure is illustrated in Figure 4.4. It is divided into four performance monitoring layers: collection, storage, analysis and visualization. Data collection (see 4.3.3 for details) distinguishes between runtime data and metadata. The former is acquired directly on the compute nodes and must not noticeably influence the executed jobs. Therefore, it is a critical task. Meta data can be gathered from multiple sources provided by the batch system. The collected job data are stored for a post-mortem analysis and visualization. Section 4.3.3 describes requirements and challenges in storing meta and runtime data for short- and long-term. The visualization and the analysis of job data are depicted in section 4.3.5. The proposed monitoring infrastructure provides a system and a job view from the same data sources.

4.3.2 Performance Metrics

In order to enable a meaningful evaluation of the job data without significantly influencing the job performance, a minimum number of selected metrics is collected. Table 4.1 lists a minimum set of metrics, which enable a rough performance analysis and categorization of jobs (see Table 4.3).

IPC (instructions per cycle), *FLOPS* (floating point operations per second), *Main Memory Bandwidth*, and *Power Consumption* are acquired with hardware counters provided by LIKWID [21]. The first two are recorded per CPU-core, whereas the second two are only available per socket. Since *FLOPS* can have different precision, they are normalized to single precision, e.g. one double-precision FLOPS is stored as two single-precision FLOPS.

Main Memory Utilization, *Infiniband Bandwidth*, *File I/O Lustre Bandwidth*, *File I/O Lustre Metadata* and local disk bandwidth are collected with node-level granularity, e.g. by reading from the *proc* filesystem. To evaluate the I/O performance, file I/O bandwidths and metadata are collected. For example, the bandwidth should be considered with the number of read and write request and the data volume per request. Depending on the file system type and the underlying storage hardware, pathological I/O is detected. For example, meta operations, such as file create and close, are usually more harmful on parallel file systems than on local disks.

Data Field	Description
Job ID	(unique) job identifier
User	(unique) user identifier
Project	name of the project
Job State	running, aborted, completed (successfully), out of memory, walltime exceeded
Start	start time (epoche time in seconds)
End	end time (epoche time in seconds)
Walltime	requested wall time
Partition	name of the partition or system
Node Exclusive	exclusive node usage
Tags	bit pattern of job properties (maximum 64)
Requested Resources	
Number of Nodes	number of allocated nodes
Number of Cores	number of allocated cores
Node and CPU List	list of allocated node and CPU IDs (for exclusive jobs, CPU lists are omitted)
Main Memory	amount of main memory in bytes
GPUs per Node	number of GPUs per node

Table 4.2: Metadata

4.3.3 Data Collection

We distinguish between two types of job data: runtime data and metadata. Runtime data are acquired and collected on a per-node, per-socket or per-core granularity. Metadata are collected per job and provide the basic parameters of the job execution. They are also required to map runtime data to a specific job.

Job Meta Data

In HPC or on computing clusters, job schedulers are used to assign computing resources to a job. Usually, they also offer the possibility to retrieve and store metadata of individual jobs. Table 4.2 lists all relevant metadata that is acquired per job.

SLURM is a job scheduler widely used in HPC. It offers the possibility to query various job properties after a job has been submitted. During the job's prolog and epilog phase start and end time of a job can be determined directly, while SLURM also provides a small set of environment variables with key parameters on the job: job identifier, user and a list of compute nodes. However, this information is not sufficient to evaluate the efficiency of resource usage, especially if multiple jobs per node are allowed, details on the actual resource allocation is missing.

The SLURM controller provides the most detailed source of job metadata, which can be directly stored into a separate job database. However, some if this data is required on the compute nodes. In order to enable efficient access to the relevant data, an in-memory database or message broker such as RabbitMQ[3] can be used. This makes the data available before the job's prolog and, thus, also enables live visualization and analysis. The weakness of this approach arises from a potential overload of the in-memory database, which can be caused by a large number of simultaneously submitted throughput jobs. Furthermore, it has to be ensured that relevant metadata is kept for a sufficiently long period of time.

SLURM also provides a job database that contains more details about each job. Although the database is not intended to be stressed with additional queries for each job, it can be used complementary to query missing metadata, e.g. in case other methods failed to acquire the data.

Job Runtime Data

A continuous data acquisition can be performed with self-written scripts or collection daemons. The latter usually come with a large number of plugins and are maintained by a vendor or the community. Collectd[4] is a

[3] https://www.rabbitmq.com/documentation.html
[4] https://github.com/collectd/collectd

data collection daemon, which supports plugins written in Python, Perl, Java and C. It is open source and has a large user and developer community.

To collect the metrics listed in Table 4.1, seven read plugins are used: cpu, disk, gpu_cuda, infiniband, likwid, lustre and memory. A write plugin is used to send the data to InfluxDB. The InfluxDB, LIKWID, Lustre and Infiniband plugin have been developed in the context of the project ProPE. Furthermore, Collectd has been extended to execute read plugins at the same time on each compute node, which enables simpler analysis and more intuitive visualization.

Job-Agnostic Collection

Time-series data can be collected job-aware or job-agnostic. The latter is advantageous for several reasons. First of all, a job-agnostic collector is less complex. It can be executed as a daemon process without the need to query for active jobs or listen for job begin and end signals. There is no need to map per-process and per-core metrics to jobs at runtime, which avoids additional runtime overhead and redundant data to be transferred to the database.

Furthermore, the queried data sources are the same for each measurement time and no redundant data is stored. For example, some metrics can only be acquired on a per-node basis (see Section 4.3.2). For throughput jobs that share the same node, job-aware per-node data is stored redundantly. Additionally, the job identifier was an additional column to be stored for each measuring point, which would require more overall storage space.

Eventually the construction of time-series databases and the speed of queries have to be considered. Job-aware data points impose higher demands on the database. In order to be able to perform job-specific queries within an acceptable time frame, the job identifier would have to be saved as a tag (see Section 4.3.4) or in an additional indexed column. Time-series databases such as InfluxDB keep tags in main memory for performance reasons. Since the amount of job ids is increasing over time, the amount of storable jobs was limited by the available main memory for such databases.

Another advantage of job-agnostic data is the ability to easily generate a system view. Job-specific data is queried using the job's meta data (start and end time, host list, and the CPU list per host).

4.3.4 Data Storage

Time-Series Database

The chronological sequence of measurements is stored in a time-series database (TSDB), which enables faster writing and better compaction for temporal data sets compared to relational databases and elastic search [5]. One possible choice is InfluxDB[5], which can easily handle the workload for the proposed set of performance metrics and 2500 compute nodes. InfluxDB is an open-source TSDB with a large community. It provides an SQL-like language. Each metric is represented as a table, which is called measurement in InfluxDB terminology. Each measuring point contains the following data: time, host, and value. Some metrics additionally contain the CPU core or socket number as integer value.

Time-series databases such as InfluxDB use the timestamp and so-called tags to identify a measuring point, which enables faster queries, similar to indexes in relational databases. An alternative to InfluxDB is the PostgreSQL extension TimescaleDB[6], an open-source database built for analyzing time-series data with SQL.

Relational Database

Metadata are stored in a relational database such as MariaDB, MySQL or PostgreSQL. For efficient queries, several columns have to be indexed. The job data from Table 4.2 are stored in a single SQL table with *Job ID* being the unique key. Using this key in a separate footprint table, each job can be assigned a performance footprint that is generated from the timeseries data.

[5] https://github.com/influxdata/influxdb
[6] https://github.com/timescale/timescaledb

Tag Name	Formula and Threshold
unrestrained	-
memory-bound*	$\frac{\texttt{memory bandwidth (measured)}}{\texttt{memory bandwidth (maximum)}} > 80\%$
compute-bound*	$\frac{\texttt{FLOP/s (measured)}}{\texttt{FLOP/s (maximum)}} > 70\%$ or $\frac{\texttt{IPC (measured)}}{\texttt{IPC (optimal)}} > 60\%$
GPU-bound	$\texttt{GPU utilization} > 70\%$ or $\texttt{GPU utilization} > \texttt{CPU utilization}$
IO-heavy	$\frac{\texttt{IO bandwidth (measured)}}{\texttt{IO bandwidth (maximum)}} > 60\%$
network-heavy	$\frac{\texttt{network bandwidth (measured)}}{\texttt{network bandwidth (maximum)}} > 60\%$

Table 4.3: Job tags currently used by the monitoring system. Tags that are marked with an asterisk use formulas and thresholds that are explained in more detail in the *Node-Level Performance* section of Appendix A.2.

4.3.5 Data Analysis and Visualization

Awareness of the efficient use of allocated resources is an important concern for users and system providers. Based on performance footprints, jobs can be automatically characterized, tagged and finally, inappropriate allocations identified.

Performance Footprint

The monitoring software stack includes an automatic analysis, which categorizes jobs based on their performance footprint. Therefore, the mean value for each performance-relevant runtime metric is determined and stored into an extra table in the database. The job footprint also comprises the job metadata and metrics that are not performance relevant, such as the main memory utilization, where the maximum value is of relevance, and temperature.

Using the performance footprint, jobs are tagged according to the formulas and thresholds in Table 4.3. We distinguish two main categories, unrestrained and resource-bounded jobs. The latter comprises jobs that use a large fraction of a specific resource. Therefore we determine the maximum value that can be achieved for each runtime metric and put them in relation to the measured values.

So far, we have specified five tags to more precisely describe a possible limitation by resources. The job performance can be limited by memory accesses, FLOPS or instruction throughput, GPU utilization, I/O operations and network traffic. A job can have none, one, or multiple tags.

The job tag already provides an optimization hint. Jobs that have been marked as bounded should be optimized for more efficient use of the respective resource. Alternatively, faster or more suitable resources can be used, e.g. for a compute-bound job a faster CPU or for an I/O-heavy job a faster I/O system.

It turned out that most jobs on TU Dresden's Taurus system are unrestrained. This is due to the large number of throughput jobs, which are often developed in script languages and use only one core. Furthermore, such jobs often share the node-local resources with other jobs on the same node, which can result e.g. in cache perturbation and thus a sub-optimal performance. To optimize throughput jobs, their execution has to be analyzed on exclusively allocated nodes.

Jobs that are not limited by a specific resource (tagged unrestrained) can also be promising optimization candidates, e.g. jobs with a low CPU utilization. If such a job consumes a significant amount of CPU hours, it is reasonable to inform the user about the job performance. The CPU usage metric is a simple indicator to determine whether the allocated resources have actually been used. However, the proposed analysis can only determine the resource usage and not the efficiency of the algorithm or its implementation. For a more detailed insight, performance tools such as Score-P and Vampir can be used.

Job Visualization

We have developed a powerful interactive web-based visualization as a supplement and extension of the footprint analysis. The web frontend uses the Angular web application framework with PHP backend and provides two access modes. The administrator mode is secured by password and provides access to the data of all jobs. The user view, with access to the user's job data only, uses on an already existing login authentication in the backend, such as Shibboleth, and thus can be easily integrated into an existing HPC web portal. Both views provide the same functionality and rely on the same implementation.

On TU Dresden's Taurus system we typically see 10,000 jobs or more per day. Therefore, we designed the interactive analysis as a top-down approach. Starting from the table view, the user can navigate from project data through groups of jobs to the metadata of an individual job and finally investigate the job's footprint and runtime metrics in a timeline view. To limit the jobs displayed, a time period can be specified. To find jobs with specific properties, the table can be sorted by any column, e.g. by consumed CPU hours to find jobs where an optimization has a large impact on the system utilization. Additionally, there is a filter mask to find jobs that match several properties. When a job has been selected, the timeline view opens.

We use timeline charts to visualize the resource utilization of a job over time. After a job is completed, timeline charts can help to identify periods of inefficient resources usage. However, they are also suitable for the live assessment of performance during the job's runtime. In case of unexpected performance behavior, the user can cancel the job, thus avoid long execution with subpar performance. Each monitored metric from Table 4.1 is represented by a timeline, whereby metrics with the same unit and data source are displayed in a common chart, e.g. different Lustre metadata operations. Timeline charts are shown for each individual job by selecting time-series data using the job's metadata (time range, hostname, CPUs, GPUs). Figure 4.5 shows how the timeline charts are arranged below the job metadata. By default, the charts are arranged in two columns and several rows.

Figure 4.5: The web frontend shows job metadata and timelines.

Complementary to the timeline charts, statistics on metadata and footprints over multiple jobs can be displayed with the footprint view. To analyze the footprints of a larger number of jobs, a visualization with histograms and scatter plots can be used. Histograms can be used to determine how a performance metric is distributed across the individual jobs. Regarding the representation of metrics in the histogram, we distinguish between three types: *Job States*, *Job Tags* and *Performance Metrics*. For example, one can see in Figure 4.6 that about 70% of all recorded jobs (10 millions) are successfully completed. It should be noted that we only record *Performance Metrics* for completed and timeout jobs. About 95% of all completed jobs could not be

tagged and thus marked as unrestrained, the remaining jobs are mainly compute and memory bound. Most jobs have an IPC value between 1.12 and 1.68, the most common flops rate is about 4 GFlop/s. Bars with a very small number of jobs and a very large metric value represent job outliers. A click on such an outlier bar lists all corresponding jobs in a table. This job table can then be used for further job analysis. It is also possible to display two metrics in a scatter plot as seen in Figure 4.7. Each dot is characterized by a color that represents the number of associated jobs. Points that are far away from the point cloud represent outlier jobs with respect to the two performance metrics. One can generate such scatter plots for each month or year over all jobs and compare them. For this reason, we call these plots *Performance Maps*, because they reflect the job behavior of a time span very well by comparing two performance metrics. The chart on the left in Figure 4.7 compares *IPC* and *CPU Power* and shows a hot spot for *IPC*=1 and *CPU Power*=40 watts. The chart on the right compares *GPU Usage* and *GPU Power* and illustrates that both metrics increase proportionally.

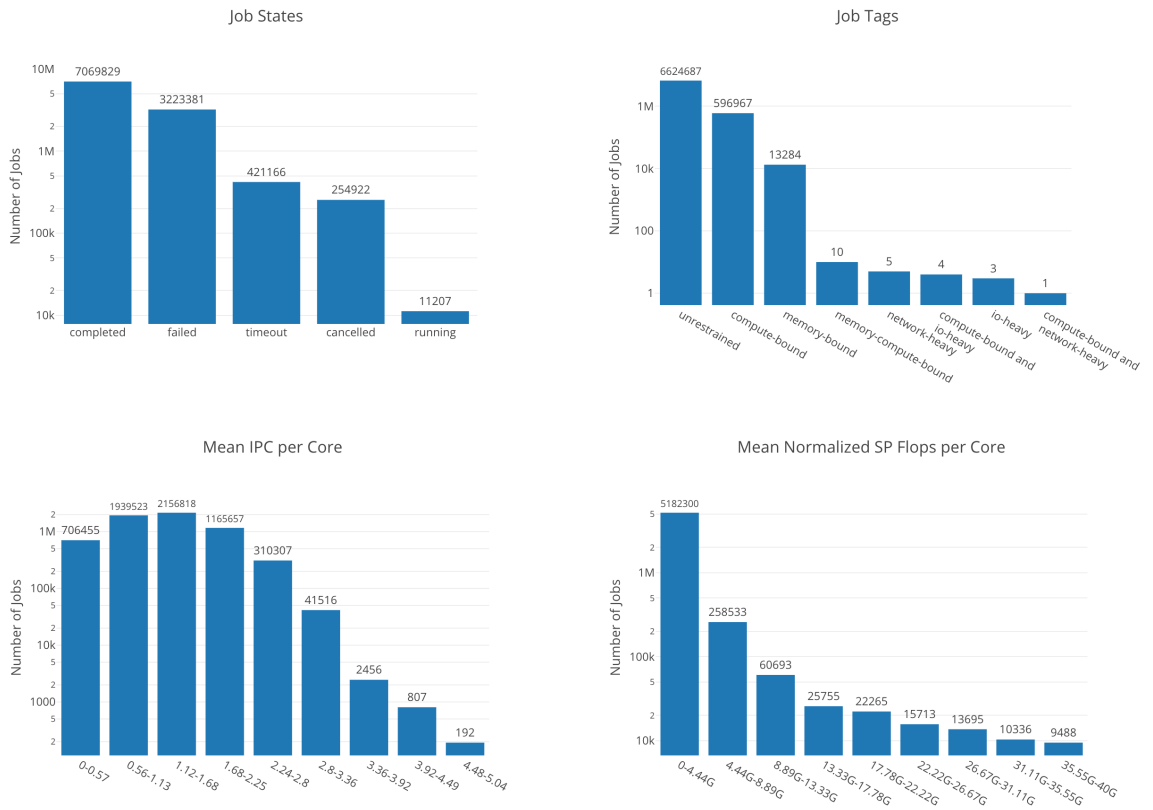

Figure 4.6: Performance footprint visualization of *Job States*, *Job Tags*, *IPC* and *FLOPS* as histograms for a period of one year on TU Dresden's Taurus system. About 70% of all recorded jobs (10 millions) are successfully completed (top left). About 95% of all completed jobs could not be tagged and thus marked as unrestrained, the remaining jobs are mainly compute and memory bound (top right). Most jobs have an IPC value between 1.12 and 1.68 (bottom left), the most common flops rate is about 4 GFlop/s (bottom right).

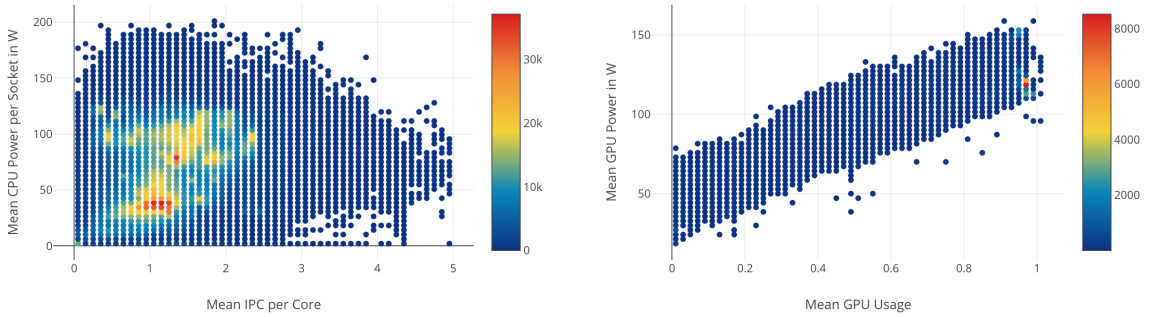

Figure 4.7: Performance footprint visualization as scatter plots (*Performance Maps*) for a period of one year on TU Dresden's Taurus system. The chart on the left compares *IPC* and *CPU Power* and shows a hot spot for *IPC*=1 and *CPU Power*=40 watts. The chart on the right compares *GPU Usage* and *GPU Power* and illustrates that as the GPU usage increases, the power increases proportionally.

4.4 Knowledge Transfer and HPC Curriculum

One important component of a successful national Performance Engineering effort is knowledge transfer. After conceiving a working Performance Engineering Process in Section 4.1, for maximal benefit this has to be combined with the respective tools to leverage hardware performance data in Section 4.3. However, to get away from having a theoretical blueprint, towards actually living this process, it has to be disseminated to the corresponding people on all levels of the spectrum. Starting with the Staff members at the HPC facility all the way to the HPC Users without programming knowledge, suitable training and high quality documentation is of paramount importance to actually fill this process with life. As described in Chapter 2, the current situation of the HPC landscape in Germany is overwhelmingly federally organized. This results in the above mentioned challenges in communication and coordination.

Currently most centers have their own isolated training programs and documentations that are neither well coordinated, nor communicated effectively on a federal level. One notable exception of this is the HPC calendar of the Gauss Alliance: Here an effort is undertaken to consolidate all training events into one comprehensive list. This project, however, is also not free from challenges. Initially filled with training dates of all members by staff of the Gauss Alliance, it is now up to the respective HPC centers to supply their training and workshop dates and insert them into the calendar. This decentralized approach is currently met with differing diligence and success. Since these dates have to be announced both locally as well as in the GA calendar, data duplication is the norm, resulting in additional work for personnel and therefore lower acceptance.

In order to sensibly and sustainably coordinate training and knowledge transfer of HPC contents nationwide, long-lasting solutions are needed with secured funding and binding commitment of the member sites. The example of HPC calendar shows two important points: On the one hand an optional offer can be used to increase awareness and visibility of own trainings when it is placed prominently and in connection with low amount of required work. On the other hand any kind of optional offer often struggles to reach widespread adoption due to the additional work involved, which is not anyone's dedicated responsibility. While it is not sensible, nor effective to neglect local differences and bake the knowledge transfer activities of all HPC centers into one monolithic solution, a solid underpinning would constitute a significant advantage from the current situation. To achieve this, a mandatory foundation of services and coordination would be an excellent way to enhance the current situation of knowledge transfer. This way HPC centers can do both: benefit from coordination advantages within this foundation, while still being able to build on top of this to specialize and cater to their different local needs and strengths. Prototypes on how this could be implemented were incepted and successfully demonstrated in the following sections 4.4.2 and 4.4.3.

One additional important common point in both training and documentation is the definition and differentiation by target audiences. Therefore this is covered in advance by Section 4.4.1 and then subsequently applied to the two following sections.

4.4.1 HPC target groups

A seasoned HPC user, equipped with experience of multiple different HPC centers, will require different information than somebody that has never before come into contact with Unix. Furthermore different users do not only need different information, but they need it on different abstraction levels and in their respective tongue. On the one hand, an expert needs the intricate workings and quirks of e.g. OpenMP. On the other hand, beginners are searching for a holistic understanding, covering the basic concepts without too many technical terms. Every one of these words is a hard problem, that they also need to look up and gain an understanding of. Furthermore the different user types could get an idea, which topics might be relevant for their level of knowledge and which areas they might currently be missing, without feeling lost in the sheer scope of HPC topics. This has the advantage of reducing the information for each user: While the expert does not need to sift through introductions to get the login-nodes of his HPC center, the beginner is taken by the hand and guided around the technical difficulties.

Therefore, we defined different target groups in Table 4.4. They should be used to not only be able to present information in the required depth, but also the appropriate tongue for the different users. While they are not completely disjunct and also do not encompass every possible user, we outlined some basic target groups to cater different information to users with different needs.

Table 4.4: HPC Target Groups

HPC Basics	These are absolute beginners. Without any prior knowledge of Unix or computer systems, here usually scientists from other disciplines want or need to use HPC resources. They need to be picked up in an easy-to-understand language and provided with an intuitive and colorful picture of most HPC terms. The aim here is to (over-)simplify things to a point, where total beginners get the qualitative knowledge they need to get started with Unix and HPC in a fast and easy way.
HPC User	An HPC user has basic unix knowledge and in general some background on computer systems. In contrast to an HPC Developer, a user does not change or modify the application, but only uses software which was or is developed by someone else. The aim here is to provide them with the necessary information or specifics to be able to effectively use a given HPC system with a given program.
HPC Dev	HPC developers are able to write and modify their programs. They have more intricate knowledge of computing systems and at least a basic understanding about parallel programming. They might need information on the concrete parallelization of their own code or how to avoid common pitfalls while trying to utilize more than one or few cores or employ existing and optimized software packages.
HPC Admin	HPC admins operate HPC systems. They usually have a very good knowledge about Unix systems, software packages and their users. Furthermore they are adept at network & cluster configuration and file service operation. Information about how to install and provide certain modules or on rolling out automatic performance monitoring could be an example for knowledge this group would benefit of.
HPC Expert	HPC experts are usually staff members at an HPC center. They have knowledge in all areas of HPC and additional specialized expertise in some HPC topics. They can provide help with the parallelizing of code and improving performance. They have advanced and fine-granular knowledge of the deployed computing systems and how to utilize their resources and specifics to their fullest extend.

In addition to our defined target groups, we also use the terminology ‘basic’, ‘intermediate’ and ‘advanced’ as vertical categorization for different target knowledge levels. This (latter) categorization covers a terminology widespread across all communities. This contains a certain ambiguity in its definition since, e.g., ‘basic’ or ‘beginners’ will be interpreted differently by different people. Furthermore, it does not reflect the variety in pre-knowledge of practitioners. Despite these challenges, it is necessary to have a form of vertical categorization to

differentiate between different courses aimed for the same target audience.

4.4.2 Training

HPC training is an integral part of the dissemination of HPC knowledge and enables end users to understand and apply PE-related activities. Existing training courses cover all aspects of HPC and scientific computing, and corresponding long-standing programs are mostly provided through the three German Tier-1/0 computing centers which are also PRACE Advanced Training Centers. Although HPC training courses in Germany show a good breadth and depth in general, the quality and (free) accessibility of training material varies and the material is often spread across many sites. Furthermore, training courses are usually developed and categorized by technical content and focus less on having a clearly defined target audience and oftentimes do not outline required knowledge for participants.

HPC Curriculum

Therefore, the community should aim to establish an overarching HPC curriculum including well-structured, accessible and shareable training material for a wide range of audiences.

One notable initiative in this direction is the HPC certification forum [2] started within the PeCoH [1] project. This certification forum aims to categorize and define a curriculum for HPC practitioners that is similar to a school curriculum. Their idea is focused around the HPC Skill Tree [11] that lists in detail all kinds of knowledge areas needed when working with HPC systems. Following, they want to certify HPC practitioners that have or gain knowledge in a specific area. On the other hand, HPC centers can provide training targeting particular skills defined in the tree, or HPC centers can make a subset of the skills a requirement for the access of an HPC system. However, the HPC certification forum itself does not focus on providing training material. Instead, they provide an abstraction layer on skills and leave it to the single HPC centers to create and carry out training courses.

In order to efficiently utilize HPC training resources we recommend establishing national distributed curricula across all Tier centers. Due to the large audience interested in entry level courses, local and regional HPC centers (Tier-2/3) should establish introductory HPC courses and develop a course curriculum for specific target groups (see Table 4.4) being in demand. In the case when more advanced level training is of interest, usually for a smaller subset of the audience, and if such a course is not offered locally, one should consider travelling to the HPC centers (Tier-1/0) where more specialized courses are available.

At RWTH Aachen, for complying with the above recommendation, the course *Parallel Programming in Computational Engineering and Science* (PPCES) has been split into *Introduction to HPC* and *PPCES* so that a broader local audience interested in the entry level course could attend it. The *Introduction to HPC* course was very well received by the students and it is now part of the HPC curriculum and offered regularly along with the more advanced *PPCES* course.

So far existing material has been provided in great variety by different sites. We therefore reviewed available training material of numerous Tier centers in Germany. Additionally, we started to categorize available course material by the users' knowledge level (compare Section 4.4.1) and by content-related skills.

Instead of compiling a new overview list of courses for marketing, we investigated and relied on the above mentioned course calendar of the Gauss Alliance[7], piping our findings and feedback back to the Gauss Alliance. The calendar lists the training events with date, venue, organizer, language, knowledge level (beginner, intermediate, advanced) and other related information. One important note, on the example of this course calendar, is that it prompts to enter information about a course in a consistent and comprehensive manner. Currently, german HPC centers have highly diverse course announcements where requisite information about a course can be missing. In order to better guide students in selecting an HPC course and help trainers to attain an audience with required target skills, the description of a course opening announcement must include a comprehensive syllabus and necessary information. To facilitate this, a common template is needed covering exhaustive information for a course description. The course organizing parties will be able to sustain consistency and clarity in course descriptions across all Tier centers by referencing such a template at the time of preparing a course announcement.

Additionally, the calendar presents the registered courses visually on a map of Germany. Such a centrally-provided platform efficiently presents an overview of courses. This supports promoting the rich set of already available courses in Germany. However, currently the usage of this great resource is optional for the HPC

[7] https://hpc-calendar.gauss-allianz.de

centers. This significantly increases the difficulty of getting a complete overview over the existing courses and training activities and therefore any work on consolidating german training activity exhaustively is exhausting and challenging.

Since PeCoH's Skill Tree presents a comprehensive and good overview on knowledge areas, we extended our list of existing training material in Germany with a categorization corresponding to this tree (see Table 4.5). The incorporation of these categories in to course descriptions will enable trainers and trainees to teach or select a course appropriate to the their needs. Moreover, the aggregated list shows a centralized overview of the HPC courses in Germany categorized by various course attributes. A complete document outlining the results of the findings can be provided upon request.

A next step would be to bring existing and new material into a consistent form and provide a well-structured base collection of material on a common platform like the HPC Wiki (refer to Section 4.4.3) to foster easy access and an easy ability to share it. This should include, e.g., links, slides, videos or web documentation.

Train the Trainer

First and foremost, the dissemination of HPC knowledge can only be fostered if appropriate HPC trainers are available and well-educated. In practice, teaching HPC content came from two sides: On the one hand the technical experts gave courses to their users to increase efficient usage of the available resources. On the other hand, university teaching started as part of HPC chairs.

While the latter teaches the concrete HPC usage and underpins this with theoretical and scientific knowledge, the former is only targeting the practical skills. And while the latter is organized and quality controlled by the universities, the former is usually grown historically into today's structures.

To streamline this, enable more people to train and assure a high and consistent quality of the training, train-the-trainer activities across HPC centers are of paramount importance in order to increase and improve the HPC teaching expertise.

While the concept of train-the-trainers is not new, very little information on best practices is publicly available in the HPC community. Within Germany, only the HLRS seems to promote train-the-trainer workshops on parallel programming [10] on a regular basis. They ask future trainers to attend a regular HPC training course, e.g., on programming with OpenMP and MPI, and provide all teaching and exercise materials to them. Furthermore, future trainers get paired up and are asked to help students during the exercises (after a short briefing on typical problems with the exercise). Additional short meetings are planned to review and discuss the train-the-trainer program. It is noteworthy that train-the-trainer attendees at HLRS are required to already have the technical knowledge that will be transferred during the course.

Another organization that focuses on the education of trainers is Software Carpentry [18], now also known as The Carpentries [20], with its sub idea of HPC Carpentry [12]. The Carpentries targets at broadening teaching skills and maintaining quality teaching material in a community approach. For that, they provide technical material mainly as prose (for self-studies) including an recommended schedule and instructor guides. Furthermore, The Carpentries certifies teachers by providing a two-day instructor course that covers teaching methods (instead of technical content).

While the preferences in the methods of organizing a train-the-trainer program at each HPC center is individual, the aim to constantly improve the methods should be common. Following this, we created a qualitative questionnaire that can help to refine the train-the-trainer program based on the feedback from participants (see appendix C.1). This template questionnaire is a base version that can be extended and modified further depending on the focus of interest, or e.g. can be transformed into a quantitative by replacing the yes/no answers with a scalar grading of choice.

Summarizing, combining the existing wealth of high quality teaching material into a well structured curriculum would be beneficial for HPC centers in Germany. Complimenting this with a comprehensive train-the-trainer program, would empower the staff members to not only have the technical skill set but also enable them to didactically and methodically teach those skills in an optimal way.

4.4.3 Knowledge Base

As outlined above in Section 4.4, the need for common and coordinated teaching and documentation is pressing. While being essential for smooth operation, maintaining a comprehensive knowledge base for an HPC center let alone all HPC centers in Germany is a challenging task. More often than not, documentation is hard to understand by the target user (e.g. a HPC beginner without any UNIX knowledge), badly (if at all) structured and extremely outdated. Updating old information, restructuring the material in response to new technological

Table 4.5: Exemplary audience categorization of some existing HPC-related training courses. Target groups (excerpt): B=Basics, U=User, D=Developer. Target knowledge level: B=Basic, I=Intermediate, A=Advanced. Attained skill: Refer to the HPC Skill Tree [11].

Content	Target Group	Target Knowledge Level	Attained Skill
Intro to HPC	B	B+I	KN1.2,1.1,2.1,3.1,2.2,3.2
OpenMP	U/D	B+I+A	K3.1+SD3.2
MPI	U/D	B+I+A	K3.1+SD3.2
PE Workflow	D	I+A	PE5+PE4+PE2.2
Cluster Usage	B	B	USE1,2,3
Parallel Architectures	U/D	I+A	KN1.2
Tools	D	I	SD2

developments or specifically formulating several versions for different users is frequently postponed due to high workload of administrative personnel and more pressing issues (e.g., the fix of a new "Spectre" security breach).

However, this leaves documentation in a state where users, have a hard time getting the information they need, to use HPC systems efficiently. This in turn has a host of different implications:

- new users are deterred from the complexity and stick to their workstation to answer their scientific questions, leading to suboptimal productivity
- users may not be able to run their code at all
- users may run code in a suboptimal way since they have little knowledge of what they are doing and how they should e.g. link a performance optimized library like fftw or MKL
- users may write their own code in a suboptimal way, leading to hard-to-parallelize code, since it was not designed with performance/parallelism in mind
- concrete guidelines are often missing how to optimize code and perform a structured PE process
- increase in support tickets asking for all the aforementioned missing information

One way of tackling this would be a common knowledge base between all HPC centers. While presenting its own set of challenging problems as detailed below, this has the significant synergetic advantage of distributing the workload of keeping documentation up-to-date and well understandable for different users across the different sites. While this knowledge base could replace the different documentations at the various HPC centers in Germany in the near future, at the beginning it presents the challenge of getting sufficient quality and differentiation into the knowledge base to reach widespread adoption. However, once the cross-site documentation does not have to be maintained next to the normal one, but starts to replace local documents, the synergetic effects of documenting everything only once centrally would ensure self-sustainability of this service. A more detailed discussion of which information will be centralized here and in which areas synergies can be achieved is outlined below.

We started a prototype of a cross-site knowledge base at `https://hpc-wiki.info`. The goal here is to provide a starting point with initial content and a thought-out structure. Following are some of the problems and guiding design decisions of the above-mentioned prototype.

Ease of Use

User experience of a documentation is the most important factor contributing or hindering the adoption and acceptance of a novel offering like this. In order to facilitate this, we selected MediaWiki as a platform, since most people are familiar with the look, feel and interface from Wikipedia. Furthermore it is intuitive to understand and use, even for newcomers.

Access

The easier access to the knowledge base is, the higher the chances of adoption and widespread usage. Therefore the main part of the knowledge base will be readable publicly without any authentication or access control to make life as easy as possible for the overwhelming majority of users, who want to look up something in an easy and quick way. Next to this, special protected sections can be created for certain users with e.g. confidential or expert information who want to limit access to specific information to a smaller and possibly well-defined group to use this as an exchange platform.

Write access is slightly more nuanced than the free-for-all reading policy: On the one hand, when someone e.g. stumbles across a minor error in the knowledge base like a link that changed (and therefore broke), an old/outdated description of a machine that was upgraded or a minor error in some conceptual article, they should be enabled and encouraged to change this information immediately. If it only takes a few seconds and is as easy as 3 mouse clicks, a significant number of users will immediately change incorrect information. Optimally this edit is credited to their name, so they leave feeling valued and engaged. While increasing the quality of the provided content, this also binds users to the knowledge base which they now have contributed to. Also it can spread adoption by these users possibly telling others of their edits and/or the knowledge base itself. Furthermore, writing new articles, changing outdated information or incorporating better visualisations/-explanations are ongoing tasks, where user interaction and contribution is highly desired. The easier this is for the users, the more likely they are to participate.

On the other hand, not everybody is able to contribute to every topic and some form of editorial control is essential to maintain a high quality of information. The administrative overhead of reducing spam risks or mitigate outright trolling by people e.g. purposely falsifying important information right before an assignment is due, is important and can not be neglected. The strategy we pursue in this regard is twofold: In the beginning, it is very important that usage is as easy as possible, so after authentication everybody will be able to edit information everywhere. While changes go live immediately and therefore reaping all the aforementioned benefits of that, the edits will be patrolled regularly by staff members to ensure quality. As the adoption and reach of the knowledge base increases, eventually this will be changed, so the user can still submit changes everywhere, but these only go live into the article after being approved by a staff member. To implement these access patterns, a Shibboleth authentication is employed as outlined below.

Shibboleth

Authentication of users, for both protected sections and write access in the public section, is done via Shibboleth. Most users of academic compute resources are affiliated with an institution providing a Shibboleth identification and for them this is the easiest way of authentication. They click the login button, log in at their home institution and are then automatically redirected back to the knowledge base. Using a Shibboleth authentication like this, writing, editing, moderation rights and access to restricted content can be managed in a simple way. No further confirmation, double registration, new passwords or login-credentials are required and existing infrastructure is utilized. This enables every academic user to authenticate and subsequently write and improve upon articles, while staff members (in particular their Shibboleth accounts) can be added to the respective groups to get the appropriate permissions to moderate and patrol these edits. This maximizes the ease of use of the knowledge base for both the end user, as well as staff members or administrative personnel.

Not only in light of the European General Data Protection Regulation (GDPR) of May 2018, but also to ensure privacy and data security for our users it is important to consulted both, the data protection officer as well as our local Identity Management team. This allowed us to reach a well-founded decision on which Shibboleth attributes to request, what to use them for and how to ensure compliance with the GDPR. Furthermore we are investigating to possibly populate user groups and their permissions (e.g. staff members) automatically from the eduPersonScopedAffiliation in the future. This could ease the administrative process of having to specify those groups manually for every new user, that wants to be added to e.g. the staff group. However, for the time being these permissions can be set manually by someone within the 'bureaucrats' group.

Choice of Presented Information

In todays world, internet search engines provides answers to a lot of questions instantaneously, begging the question what information to present and what value this knowledge base can have over an internet search. The aim here is twofold:

- Provide a comprehensive documentation with high quality information that is needed to utilize the systems of the participating HPC centers
- Answer questions that the user cannot solve with search engines in a faster way e.g.:
  - Provide information that is not available on search engines (e.g. expert knowledge)
  - Facilitate an easy introduction for users, who do not know what they are searching for.

The first point is inherent to the traditional way of writing documentation and therefore will not be elaborated in this context: Cover most of the information that is needed to use the system at least in a basic way. The second point is a bit more nuanced: Commonly users are interested in finding the answer to their question or problem in the fastest way possible. Usually that involves internet search engines and starting with the first page of results to gain some understanding and optionally repeat this process a number of times. This process is executed by the overwhelming majority of users and we aim to support this process by systematically focusing the weak points:

Firstly, when a user starts out with a new topic, lacking any prior knowledge, they commonly do not know how to formulate their question in a way that a traditional search can easily answer. This happens frequently with complete beginners, who are missing a conceptual understanding of the basic principles and vocabulary. This ties in closely with the target orientation and especially the HPC Basics group presented in Section 4.4.1 and Table 4.4. Users starting out from square zero need a quick and really easy way of understanding the fundamentals on how to use a HPC system and generally are not able to formulate a question or search request, since the vocabulary is missing. Furthermore some areas have a shortage of beginner-level materials. With some exceptions (e.g. Vim and its interactive vimtutor, vim-game and so on) well-written/done beginner documentation is not common and sometimes not available at all. Combining a very rough and colorful overview to get started with well-written beginner-level documentation (or references to this if available) would lower the entry barrier significantly and therefore contribute to the knowledge transfer in the HPC environment. However, when available, the linking of existing material of high quality is of essential importance, since rewriting everything from scratch is neither productive, nor possible in the context of this project.

Secondly, also advanced or expert users might run into problems with simply using a search engine and expecting to quickly get the desired information. This can have any number of reasons between the complexity or specificity of the question, confidentiality, scarcity or outright unavailability of quality documentation in certain areas. Concrete examples are the upcoming admin section of our knowledge transfer platform and the structured Performance Engineering process conceived in Chapter 4.1 which is documented and will be improved iteratively.

Summarizing, we aim to provide information and content to support the native 'ask an internet search engine and then see from there' approach of most users. Our beginners section is completed and the HPC Admin, HPC Developer and Structured PE Process articles are under active development.

Site-independence

A knowledge base hosted by a central instance and not a specific HPC site faces several challenges not exhibited by a local documentation. One important part is the differentiation between site-specific information and universal concepts applicable to all HPC centers. For instance, the question 'what is a front-end and how do I use ssh' falls into the latter section while the available front ends at a given center and their addresses are samples of site-specific information. Being of different structure, these categories of information have to be gathered and stored separately. The site-specifics will be organized similar to a database or tables, while the universal concepts are mostly texts explaining the context of the factual site-specific information. The knowledge base is targeted to contain mostly those site-independent concepts which are supplemented by side specifics where needed. How to apply for computing time e.g. is something inherently unique for every different HPC center. So the wiki instead explains the common process of applying for computing time in general and provides a list of links to the application forms of the different institutions. Instead the focus lies on the documentation of site-independent concepts like how to use, e.g., Cuda, OpenMP or how to conduct a structured Performance Engineering process.

For a presentation of these different forms of content, it would be desirable that the user selects the center of their choosing (explicit or automatically via Shibboleth authentication) and the information gets assembled together from both concepts and site-specifics specifically for this user. This would lead to an integrated and comprehensive documentation experience for the user while keeping the knowledge base operable and maintainable for a host of different HPC centers.

While this is the preferred solution, the first step was to divide the information into the two categories and present them separately. This in itself is no trivial task since the site specific information has to be collected and maintained by all participating sites. This information has to be condensed at a central point to keep maintenance to a minimum: As mentioned above, it is a challenge to operate the side-independent knowledge base as a prototype since every HPC center still operates its own documentation and therefore the additional burden of maintaining up-to-date information at two different locations should be minimized. However, once the prototype phase is over, this overhead will cease and the synergistic effects dominate.

To make the information usable, while it is divided into site-specifics and independent concepts, links are employed to e.g. link the table of different front end nodes into the article explaining the purpose of a front end node in general. While side-dependent dynamic content assembly is not natively supported by MediaWiki, this has been implemented in the scope of this project in the form of a Plugin to the MediaWiki Software. The user is able to select their institution from a drop-down menu and see dynamically build pages with information supplied by their institution. The editors of the wiki, on the other hand, can inject site-specific information for their and other institutions into every Wiki page, utilizing a custom build interface. Summarizing, this offers a seamless integration of site-dependent content into site-independent articles for users of the knowledge base.

Chapter 5

Summary and Recommendation

This paper presented the **requirements and components** of a uniform **Performance Engineering** (PE) approach at a **national level**. The ProPE partners have designed, developed, and tested building blocks of a **distributed PE support infrastructure**. The results can serve as a **blueprint** for the integration of the existing HPC PE support activities on the Tier-2/3 level into the **German NHR infrastructure** as recommended by the WR. The proposed concept aims to leverage distributed HPC expert knowledge to provide high-quality problem-specific user support at all participating centers.

Performance Engineering is a well-defined, structured process to identify the relevant performance bottlenecks in an application and then derive appropriate code changes or other measures to improve resource efficiency. Various approaches are available in ProPE to identify performance issues and understand their implications: threshold analysis, performance patterns, and performance models can be used depending on the requirements of the problem. The underlying **core PE process** consists of three steps: **performance measurement and analysis, performance issue identification (testing thresholds or performance patterns, establishing performance models), and performance optimization**. These can be carried out to Various levels of detail, depending on the experience of the analyst and the importance of the task. In fact, only few HPC users are also able to employ all PE tasks, so the scope of PE should be very broad in practice, encompassing such seemingly simple activities as workflow management, tuning of the execution modalities, and documentation of system settings.

An important component of any PE activity is the **identification of (potential) PE cases**. First, these can be triggered **by users or software developers** who have identified a performance problem themselves or require faster time to solution. They contact the local PE engineers through the established support channels. The second alternative is the identification of potential performance problems **by active performance monitoring** of the HPC resources. With appropriate monitoring of the hardware utilization in place, the HPC support team can use bottleneck identification procedures from the PE process to pinpoint badly performing applications. Thus, a system-wide, continuous job-specific hardware performance monitoring which provides reliable and relevant utilization metrics (such as main memory bandwidth, FLOP-rates, instruction throughput, vectorization ratios, IO-rates or communication frequencies/volumes) is a foundation of any PE-oriented user support. Setting up a nation-wide PE infrastructure covering all relevant Tier-2/3 center faces the challenge of very different PE knowledge levels of the HPC support staff, application scientists, and developers. **Knowledge documentation and transfer** as well as **training** of support personnel and users/developers is a central issue to keep the coordination effort between the centers low and minimize the number of PE cases which cannot be resolved locally but have to be escalated to the central infrastructure. To achieve this goal, a shared documentation platform process needs to be established, and a structured nation-wide training program is required that is tailored to user groups, knowledge levels, and application domains.

In order to meet all these requirements and procedures, we have identified and implemented the following components:

- a systematic **Performance Engineering process** following scientific practices,
- robust and simple processes that cover the required use cases of a **distributed support infrastructure**,
- a system-wide **job-specific performance monitoring infrastructure** at each site to identify both pathological jobs and those with high optimization potential,

- a **central knowledge base** containing documentation of all relevant PE core activities and all required specifications, and
- a structured **collection of training activities** within the German HPC landscape categorized, e.g., by content, knowledge levels and target groups.

Further, a common classification and nomenclature was defined for the following areas:

- **Performance metrics**: Define common metrics that characterize application performance.
- **Job Classes**: Group jobs by their performance characteristics and hardware utilization.
- **Target groups**: Group persons by task and position.
- **Knowledge levels**: Indicate the degrees of preliminary knowledge.

The infrastructure and processes used should ideally either provide metrics for **increased scientific productivity** (e.g., measured in publications made possible by the use of an HPC system) or, at a finer level of granularity, the ratio of money saved by an activity to the investment consumed by that activity. To measure and argue for the effectiveness of the performance engineering process, we provide a cost model that includes hardware, energy and brainware costs.

For an optimal operation of the distributed PE network, a **central infrastructure** and personnel are required in addition to methodology and processes. It is recommended that **critical infrastructure** be set up within an **independent organization** with long-term, stable financing, such as DFN. The following central infrastructure is recommended for a high-quality distributed PE service infrastructure:

- a **web platform** that provides a knowledge base, e.g., as a wiki, for the joint creation of documentation and specifications,
- a **ticketing system** to manage and document the progress of PE projects,
- a **calendar** of events for workshops and tutorials in order to create a **central training curriculum**,
- a platform for **source code revision control**, e.g., GitLab, for source code exchange and for administration and documentation of source code changes within PE projects, and
- a platform or infrastructure for **secure transfers of large files**.

During the ProPE project, a dedicated **HPC wiki**[1] based on the MediaWiki software was set up at RWTH Aachen University. For a **ticketing system**[2] and a **calendar of events**[3] preexisting solutions, established within the GA network at TU Dresden, were employed and modified to suit the PE needs. The **Gigamove service**[4], also at RWTH Aachen University, was used to share large files. The ProPE project built on this existing infrastructure. One problem that is not specific to a distributed PE infrastructure is the lack of trust when handing over closed-source projects to third parties. While the local HPC center is usually trusted, this is not usually the case with a remote center. A shared infrastructure with the above components located at a trusted neutral organization would certainly increase trust in a nationwide network. Ideally, all HPC centers would also rely on a **central identity Management Infrastructure** (IDM), which should also be located at this neutral organization. This would facilitate joint work, transfer of users, and the confidence to work remotely between centers. The existing IDM system within the bwHPC network could serve as a blueprint.

[1] `https://hpc-wiki.info/`
[2] `https://servicedesk.gauss-allianz.de/`
[3] `https://hpc-calendar.gauss-allianz.de/`
[4] `https://gigamove.rz.rwth-aachen.de/`

Appendix A

Performance Engineering Process

A.1 Generic guidelines for performance optimizations

Overall a performance engineer must find the best compromise between lowest algorithmic complexity and optimal mapping to system hardware (see Figure A.1).

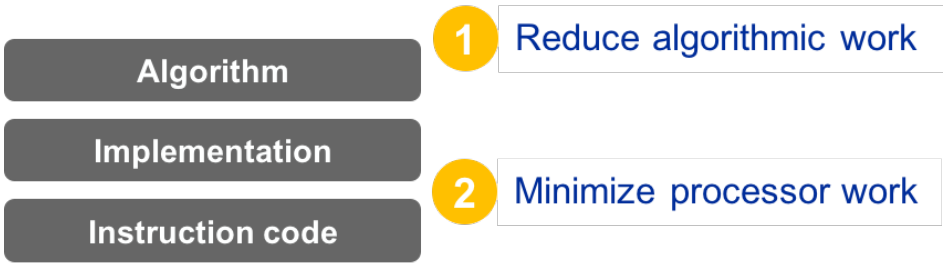

Figure A.1: Initially a performance engineer must explore the optimal algorithm reducing the work to perform a task. After implementing the algorithm is implemented in a programming language using a specific programming model the instruction overhead introduced must be minimized.

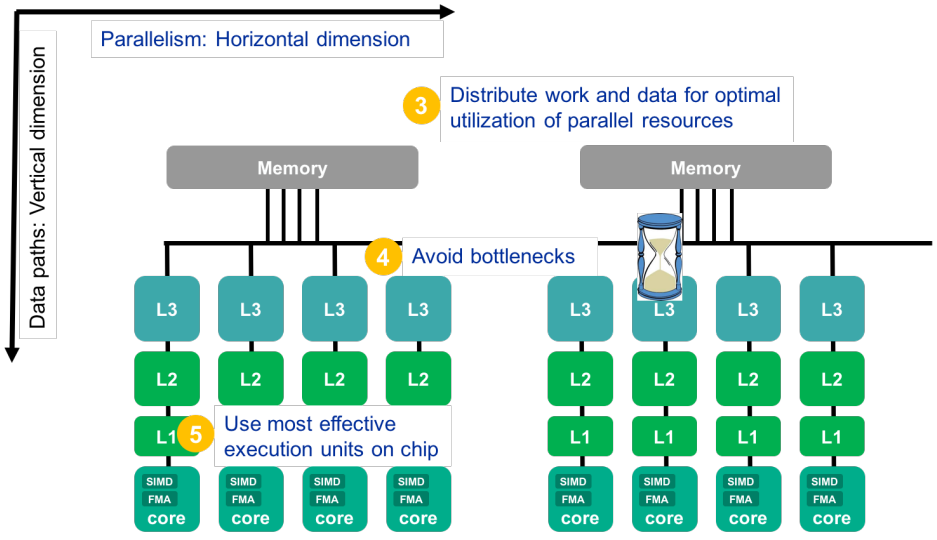

Figure A.2: There are two dimensions of optimization in modern processors: Horizontally distributing work to utilize parallel resources and vertically to manage data paths and transfer data over fast and scalable data paths. Potential bottlenecks as main memory bandwidth must be identified and the algorithm must be mapped to the most effective execution units available on the chip (e.g. SIMD units).

The following targets must be met (shown in the figures A.1 and A.2):

1. Reduce **algorithmic work**: Find an algorithm that solves the task at hand with the lowest possible complexity. Already at an early stage algorithms have also to be evaluated with regard to its suitability for parallelization and vectorization. Because modern processor architectures generate performance mostly by parallel execution it is not uncommon, that an initially inferior algorithm overcompensates its higher complexity through a more efficient parallelization or vectorization.
2. Minimize **processor work**: The primary processor work are CPU instructions. Additional instructions not related to the useful work of the applications may be introduced by the compiler, a runtime environment, or by implementation errors.
3. Distribute work and data for **optimal use of parallel resources**: Parallel processing is a essential requirement to exploit available compute resources. This involves to employ load balancing, prevent excessive synchronization, but also to make efficient use of parallel memory interfaces.
4. Identify and avoid **hardware bottlenecks**: Every chip design exposes bottlenecks. Usually these are shared resources, e.g. main memory bandwidth or external IO devices. A major optimisation strategy is either to reformulate the code such that the bottleneck is not triggered or to reduce, e.g., the data volume of slow data paths to increase performance.
5. Use **most effective execution units** on the chip: Modern chips offer multiple instructions for the same operation. To choose the most effective one is an important optimization on all modern architectures. The most prominent example is the utilization of data-parallel SIMD units, other examples are using single precision floating point instead of double precision, employing non-temporal store instructions for streaming data access patterns, or using lower accuracy alternatives for divide operations.

A.2 Detailed threshold analysis

A. MPI behavior

The following metrics and thesholds were developed as part of the *Performance Optimisation and Productivity* (POP) EU Center of Excellence[1]. To calculate detailed metrics at the MPI level, a trace analysis is performed using tools such as Scalasca/Score-P/Cube/Vampir or Extrae/Paraver/Dimemas.

To get a complete overview of application performance and a better understanding of the problems of a program at the multi-node level, all of the following detailed metrics must be measured. All metrics are numbers that are normalized to the range from 0 and 1 and can be represented as percentages. The inefficiency of the MPI program is broken down into three main factors: Load balancing, serialization/dependencies, and data transfer. The detailed indicators are as follows:

- **Load Balance Efficiency (LB)** reflects how well the work is distributed among the processes in the application. One distinguishes between load balance in time and in the number of statements executed. If processes invest different amounts of time in the computation, one should look at how well executed statements are distributed among the processes. Load Balance Efficiency is the ratio between the average time/instructions a process spends/executes for the computation and the maximum time/instructions a process spends/executes for the computation. The threshold is 85%. If the Load Balance Efficiency is less than 85%, check the problem with the Load Imbalance pattern.

  $\text{LB(time)} = \frac{\text{avg(tcomp)}}{\text{max(tcomp)}}$, $\text{LB(ins)} = \frac{\text{avg(ins)}}{\text{max(ins)}}$

- **Serialization Efficiency (SE)** describes efficiency losses due to dependencies between processes. Dependencies can be observed as wait times in MPI calls where no data is transferred because at least one process involved has not yet arrived at the communication call. In an ideal network with immediate data transfer, these inefficiencies are still present because no real data transfer takes place. Serialization efficiency is calculated as the ratio between the maximum computing time of a process and the total runtime in the ideal network (also known as the critical path). The threshold value is 90%. If the serialization level is less than 90%, check the issue pattern Serialization.

  $\text{SE} = \frac{\text{max(tcomp)}}{\text{total runtime on ideal network}}$

[1] `https://pop-coe.eu/`

- **Transfer Efficiency (TE)** can be calculated as the ratio between the total runtime in an ideal network (critical path) and the total measured runtime. The threshold value is 90%. If the transfer efficiency is less than 90%, the application is transfer-bound.

  $$\text{TE} = \frac{\text{total runtime on ideal network}}{\text{total runtime}}$$

- The Serialization and Transfer Efficiencies can be combined in the **Communication Efficiency (CommE)**, which reflects the loss of efficiency through communication. If it is not possible to create the trace analysis of an application or if the ideal network cannot be simulated, this metric can be used to show how efficient communication is in an application. If the communication efficiency is less than 80%, the application is communication-bound.

  $$\text{CommE} = \text{SE} * \text{TE} = \frac{\max(\text{tcomp})}{\text{total runtime}}$$

- **Computation Efficiency (CompE)** describes how well the compute load of an application scales with the number of processes. Computation efficiency is calculated by comparing the total time spent on multiple program runs with different numbers of processes. In a linearly scaling application, the total time spent on the computations is constant, and thus the computing efficiency is one. The Computation Efficiency is the ratio between the accumulated computation time with a smaller number of processes and the accumulated computation time with a larger number of processes. The Computation Efficiency depends on the ratio of the process numbers of two program executions. The threshold value is 80% by four times more processes. If the Computation Efficiency is low, check the issue pattern Bad Computational Scaling.

  $$\text{CompE} = \frac{\text{tcomp}_1}{\text{tcomp}_2}$$

- **Instruction Scaling (InsScal)** is a metric that can explain why Computation Efficiency is low. Typically, as the number of processes increases, more instructions need to be executed; for example, an additional computation is required for domain decomposition, and these computations are redundantly executed by all processes. Instruction scaling compares the total number of instructions executed for a different number of processes. This is the ratio between the number of instructions executed by processes in the computations with a smaller number of processes and the number of instructions executed with a larger number of processes. The threshold is 85% by four times more processes.

  $$\text{InsScal} = \frac{\text{ins}_1}{\text{ins}_2}$$

- The second possible reason for low Computation Efficiency is poor **IPC Scaling (IPCscal)**. In this case, the same number of instructions are computed, but the computation takes longer. This can happen, for example, by sharing resources such as memory channels. IPC Scaling compares how many instructions are executed per cycle for a different number of processes. This is the ratio between the number of instructions executed per cycle when calculating with a larger number of processes and the number of instructions executed per cycle with a smaller number of processes. The threshold is 85% by four times more processes.

  $$\text{IPCscal} = \frac{\text{ipc}_2}{\text{ipc}_1}$$

B. OpenMP behavior

The following metrics and thesholds were developed as part of the *Performance Optimisation and Productivity* (POP) EU Center of Excellence[2]. For a deeper analysis on the OpenMP level the tools Intel VTune or LIKWID [21] can be used. The tools Score-P and Extrae also partly support the analysis of OpenMP applications.

The detailed indicators for the OpenMP aspect are:

- **Load Balance (LB)** shows how well work is distributed among application threads. It is the ratio between the average computation time and the maximum computation time of all threads. The computation time can be identified by analysis tools as the time spent in the user code outside of synchronizations such as

[2] `https://pop-coe.eu/`

implicit or explicit barriers. The threshold is 85%.

$\text{LB(time)} = \frac{\text{avg(tcomp)}}{\text{max(tcomp)}}$, $\text{LB(ins)} = \frac{\text{avg(ins)}}{\text{max(ins)}}$

- **Serialization Efficiency (SE)** describes the loss of efficiency due to dependencies between threads and time spent on serial execution. The threshold is 80%.

  $\text{SE} = \frac{\text{max(tcomp)}}{\text{total runtime}}$

- Alternatively to Serialization Efficiency, the **Effective Time Rate (ETR)** can be calculated when analyzed with Intel VTune. It describes the synchronization overhead in an application due to threads that have a long idle time. This is the ratio between the effective CPU time, the accumulated time of threads outside OpenMP, measured by Intel VTune, and the total CPU time. The Effective Time Rate is the percentage of the total CPU time outside of OpenMP. The threshold is 80%.

  $\text{ETR} = \frac{\text{effective CPU time}}{\text{total CPU time}}$, effective CPU time is the accumulated time of threads outside OpenMP (vtune metric).

- **Computation Efficiency (CompE)** reflects the loss of efficiency by increasing the number of cores. To calculate the Computation Efficiency, two application runs are compared with a different number of threads. The aim is to determine whether the application performance deteriorates with a larger number of threads. The threshold is 80% by four times more processes. Similar to **CompE** for the MPI aspect **InsScal** and **IPCscal** can be used to explain **CompE**.

  $\text{CompE} = \frac{\text{tcomp}_1}{\text{tcomp}_2}$ $\text{InsScal} = \frac{\text{ins}_1}{\text{ins}_2}$ $\text{IPCscal} = \frac{\text{ipc}_2}{\text{ipc}_1}$

C. Node-level performance

A detailed analysis of the performance at node level only makes sense on a per kernel basis. This is especially true if the application consists of several kernels with very different behavior. In this context, kernel means a function or a loop nest that appears in a runtime profile. The metrics can be captured with any hardware counter profiling tool. Since the performance groups are preconfigured in likwid-perfctr, it is recommended to start with this tool. For many metrics also input from microbenchmark experiments are needed. Some metrics can only be captured with a thread or MPI parallel code, but solve performance problems related to single node performance.

The detailed indicators for the Node-level aspect are:

- **Memory bandwidth** is the most important bottleneck for shared node-level resources. To determine whether an application is memory-bandwidth bound, the measured memory bandwidth is compared with the result of a microbenchmark. This metric is only useful if all cores within a memory domain are utilized, as a subset of cores may not be able to saturate the memory bandwidth. This metric is measured for a single memory domain. If the condition applies, proceed with metrics under Case 1. If the condition does not apply, proceed with the metrics under Case 2. Threshold: $> 80\% =>$ memory bound.

  $\text{MEM}_{BW} = \frac{\text{memory bandwidth (measured)}}{\text{memory bandwidth max(load/copy/triad)}}$

- Case 1: **Use of parallel memory interfaces** characterizes if all parallel memory interfaces are utilized. Threshold: $> 80\%$.

  $\text{MEM}_{NUMA} = \frac{\text{memory bandwidth (measured)}}{\text{memory bandwidth one memory domain} * \text{number of used memory domains}}$

- Case 2: **Floating point operation rate**. Floating point operations are a direct representation of algorithmic work in many scientific codes. A high floating point operation rate is therefore a high level indicator for the overall performance of the code. Threshold: $> 70\%$.

  $\text{FLOPS}_{RATE} = \frac{\text{floating point rate (measured)}}{\text{floating point rate (triad running in L1 cache)}}$

- **SIMD usage.** SIMD is a central technology at ISA/hardware level to generate performance. Since it is an explicit feature, it depends on the algorithm whether and how efficiently it can be used. The metric characterizes the portion of the arithmetic instructions using the SIMD feature. Note: SIMD also applies to loads and stores. However, the width of loads and stores cannot be measured with HPM profiling. This

metric therefore only records part of the SIMD usage ratio. Threshold value: $> 70\%$.

$$\text{SIMD}_{RATIO} = \frac{\text{SIMD arithmetic instruction count}}{\text{scalar arithmetic instruction count}}$$

- **Instruction overhead.** This metric characterizes the ratio of instructions not related to the useful work of the algorithm. This metric only makes sense when arithmetic operations are related to this useful work of the algorithm. Overhead instructions can be added by the compiler (triggered by the implementation of programming language functionality or by transformations to SIMD vectorization) or by a runtime (e.g. spin wait loops). Threshold value: $> 40\%$.

  $$\text{INST}_{RATIO} = \frac{\text{total arithmetic instruction count}}{\text{total instruction count}}$$

- **Execution Efficiency.** Performance is defined by how many instructions I need to implement an algorithm and how efficiently these instructions are executed by a processor. This metric quantifies the efficient use of the parallelism of the processor at the instruction level as pipelining and superscalar execution. Threshold: CPI $> 60\%$.

  $$\text{CPI}_{RATIO} = \frac{\text{CPI (measured)}}{\text{optimal CPI}}$$

D. Input/Output performance

For a deeper analysis of I/O behavior you should look at the trace time line of the application, for example with the Vampir tool, to understand how Input/Output impacts the waiting time of processes and threads. Additionally there is the Darshan tool for a deeper analysis of HPC I/O characterization in an application and Intel Storage Performance Snapshot for a quick analysis of how efficiently a workload uses the available storage, CPU, memory, and network.

Detailed metrics:

- **I/O Bandwidth** is calculated as the percentage of the measured I/O bandwidth of the application to the maximum possible I/O bandwidth on the system. The threshold is 80%.

  $$\text{IO}_{BW} = \frac{\text{IO bandwidth (measured)}}{\text{IO bandwidth max}}$$

Appendix B

Distributed Support Infrastructure

B.1 Expertise on Participating Sites

RRZE RRZE's HPC expertise covers the following focal points, which have been defined for this computing center and thus represent the HPC expertise in Erlangen.

For node-level performance engineering, these are:

- Systematic performance engineering process
- Formulation of performance patterns
- Development of high performance prototype codes and libraries
- Performance analysis of codes and hardware platforms

In performance modelling, the focus is on:

- Analytic and diagnostic performance modelling
- Execution-Cache-Memory (ECM) model—a refinement of the well-known Roofline model

Concerning too development, the focus lies on:

- Likwid Performance Tool Suite
- Nuclear Loop Kernel Analysis and Performance Modelling Toolkit
- Open Source Architecture Code Analyzer (OSACA)

IT Center For the IT Center of RWTH Aachen University, the following defined areas of competences arise, which constitute expertise at the Aachen location within the project. RWTH Aachen University is also involved in Material Sciences as well as Engineering to a large extent. However, the focus of the IT Center is on the analysis and parallelization of large-scale simulation codes for different system architectures. These also include heterogeneous and multi-core architectures, e.g., NVIDIA's GPUs.

RWTH's core area of work lies here in:

- Parallel Efficiency
- Immersive Visualization
- OpenMP 4.0/4.5
- Combination of results gathered from perfomance model with OpenMP and HPC performance analysis tools
- MUST (supports a set of checks for MPI, OpenMP and hybrid-parallel programs)

ZIH The following areas of competence have been defined for the ZIH computing center in Dresden. These are the hallmarks of the ZIH's expertise:

- Parallel programming and performance optimization
- Interactive performance analysis and visualization (with Score-P and Vampir)
- FPGA instrumentation for energy measurement with fine temporal (kHz range) and spatial (e.g. CPU socket) granularity on large HPC partition and per-job-monitoring of I/O activities
- I/O infrastructure (for exascale systems)
- Exploit dynamic behavior of applications to achieve improved energy efficiency and performance
- GPU

B.2 Formal Description of Core Process

The 1st-Level of the respective home site is the primary contact for customer inquiries. It is the responsibility of this organization to ensure that the center's support processes are maintained and implemented. According to this rule, the customer communication takes place, for example, in the case of queries, solution releases or the communication of maintenances and incidents. If a request cannot be processed in the 1st-level, it will be forwarded to the corresponding 2nd-level support for further processing. If the 2nd-level of the home site cannot find any solution, the corresponding 2nd-level supporter reaches out to the cooperating sites for further assistance. Supporters of the 2nd-level of the cooperation sites then enter the function as 3rd-level supporter.

a) Taking on a customer request: The customer request is received in the 1st-level of the home site computing center. The request is recorded in the ticket system used on-site and checked whether processing in the 1st-level is possible. If a solution can be offered through the 1st-level, this is communicated to the customer through the same level without concerning a specialist on the 2nd- or even 3rd-level.

b) Forwarding to the 2nd-level: The customer request is handled according to local support processes. If no solution can be determined by the 1st level, the request is forwarded to the local 2nd level support. Again, the site-specific processes are addressed in the context of local support processes. This may include the following issues:

- How does the information transfer to the 2nd level work?
- Is it necessary to obtain standard information in advance from the customer before forwarding?
- Does the customer have to be informed about intermediate steps?
- Which response times, solution times and escalation times need to be considered?

c) Forwarding to the 3rd-Level As soon as the customer request has reached the 2nd-level support of the home site, it will be processed according to the location-dependent processes. If a solution to the problem can be identified, it will be communicated to the customer according to the site-specific processes. If the 2nd-level of the home site cannot provide a solution to the problem, support from a remote competence center can be requested. The remote competence center, referred to as the remote site, will then serve as 3rd-level support. In this case, it is obligatory to obtain consent from the customer in advance whether it is in the customer's interest to forward the inquiry to another cooperating computing center and if the customer agrees to the disclosure of personal data. For this purpose, the user will be presented with a digital consent form usually by the 1st-level.

The inquiry is usually pushed back to the 1st-level in order to gather the data protection declaration of consent and conditions from the customer. Only if the customer agrees to the forwarding of personal data and provides relevant information, further processing to another site is possible. The 1st-level waits for the response and passes it then on to the 2nd-level. In the event that the customer does not agree to further involvement of another computing center, it is in the responsibility of the 2nd-level support to decide if he or she reaches out to the customer to find other solutions.

After receiving the consent form, the 2nd-level supporter opens a new ticket in the common ticketing tool. It contains the request for support of another specialist on another site. On the home site, the 2nd-level specialist

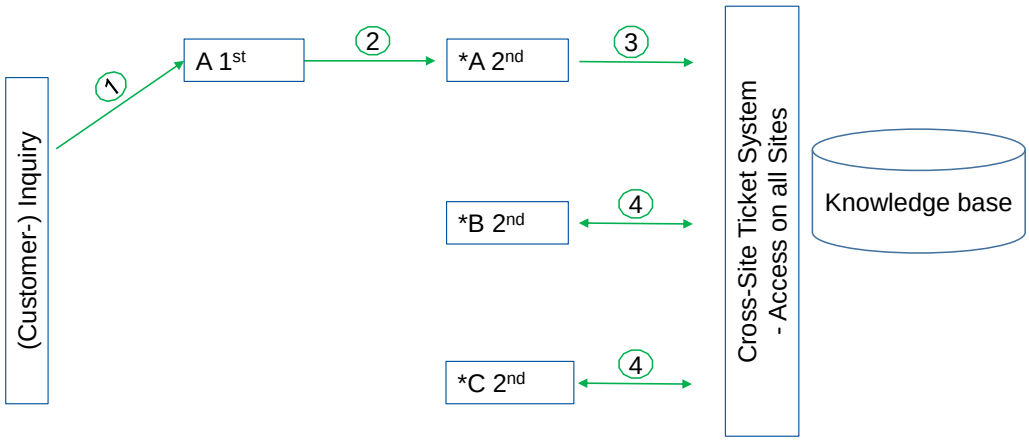

Figure B.1: Multi-Tier distributed support process with three support levels.

documents the forwarding in the home site's ticketing tool. The customer request is then forwarded to the 3rd-level of inter site.

The jointly editable ticket contains not only the actual request and/or question of the customer, but also all the standard information required for the support provided by the 3rd-level (e.g. e-mail address, information on the systems used, what the code looks like, what error message appears, where/on which system it is being calculated). Simultaneously, the necessary and relevant files (e.g. source codes or binaries) are provided by the 2nd-level supporter for further processing on the platform GigaMove. This platform provides all relevant data for all 3rd-level supporters in case huge files (such as code etc.) need to be exchanged or transferred. Every specialist receives access to the files in order to process the request.

As soon as the ticket enters the common ticketing tool, the concept envisages that the participants voluntarily reserve the ticket for processing according to their area of expertise. This is possible by reservation of the ticket within the first 72 h after the inquiry has reached the common ticketing tool. The inquiry is then ready to be processed in the Performance Engineering Queue in this case.

If no 3rd-level supporter reserves the ticket within the first 72 hours after release, the ticket escalates. In order to avoid escalation of inquiries, a concept of responsibility for incoming tickets has been designed. This concept provides that the responsibility for the ticket queue changes monthly. In concrete terms, this means that the participating centers alternately keep an eye on the performance engineering queue. A substitute is also considered in the event that the responsible site is unable to meet its responsibilities due to a lack of personnel resources. In this way, the risk of tickets escalating or being noticed too late in the additional ticket tool is counteracted. The responsible site then analyzes the inquiry and acts upon the request. Responsible agents then distribute or assign tickets based on defined competences and expertise on sites. The aim here is to provide quality-assured support, both competence oriented and organized. If questions arise during the process, it is possible that the supporting specialist contacts the customer in the course of process. However the communication takes place, it is crucial that it is documented in the respective working step in the ticket of the common ticketing tool. The responsible 3rd-level supporter also documents all activities (inquiries/ feedback to/ by the customer, milestones, solutions or forwarding) in the processed ticket.

Communication of Solutions As soon as the request has been processed by the 3rd-level, it returns to the home site. The ticket contains the documented solution and is sent to the home site within the common ticketing tool. The home site specialist extracts all information and documents it in the home site ticketing tool. The ticket that has been opened in the home computing center is then closed by the responsible home-site supporter in the 2nd-level according to the location-dependent processes.

The gained knowledge and solution from the 3rd-level support is accessible to the 2nd-level of the home site through the documentation in the common ticket tool. The individual processing steps have been recorded chronologically in the respective ticket and then subsumed with a summary of the solution process. This serves

the purpose of clarity and information transfer for all cooperating centers involved. The specialist in the 2nd-level documents the solution steps in the wiki, which is used jointly by the cooperation partners, in order to build up a common knowledge base that initially serves the supraregional customer support for the PE processes presented.

If applicable, detailed information is provided from the 3rd-level via the GigaMove platform and can be found in the corresponding file, which is available for download. The solution is communicated in the respective service unit in the shared ticket.

The following step is the communication to the customer either through the 2nd-level of the home site or the redirection to the 1st-level of the home site, which originally received the request. Depending on the location-specific processes, customer communication normally takes place through the 1st-level support. However, a customer notification through the 2nd- or 3rd-level is not excluded, considering that the corresponding notifier will also close the originating ticket in the home computing center.

d) Unresolved Request If a 2nd-level supporter of another site, functioning as a 3rd-level support, cannot aid to solve the customer request, it must be released again for processing in the common ticketing tool. In this case, the processes described above apply. The same applies to escalation scenarios and in the case of illness or non-existent/insufficient resources.

B.3 User Satisfaction Survey Template

Dear Users,

Welcome to our HPC Performance Engineering Support (PE) survey. As an HPC user, you are particularly important to us. The same applies to your opinion about our service! We therefore ask you to give us a few minutes of your time to answer the 10 short questions about our Performance Engineering Support. Please feel free to read through the questionnaire once. We are particularly successful in adapting our service to your wishes if we know what you want and need. Your data will be evaluated anonymously, unless you wish a personal contact.

Many thanks in advance!
Your Performance Engineering Support

How satisfied are you with our Performance Engineering support?

- very pleased
- rather satisfied
- reasonably satisfied
- not really satisfied
- rather satisfied

How likely is it that you will recommend our support to a colleague? Please rate on a scale from 1 to 10

How satisfied were you with the communication to our support (e.g.duration of response time, reliability, service orientation)?

- very pleased
- rather satisfied
- reasonably satisfied
- not really satisfied
- rather dissatisfied

Which of the following words would you choose to rate our Performance Engineering Support? You can also choose multiple times here.

- trustworthy
- highly qualitative
- beneficial
- efficient
- goal-oriented
- useful
- inefficient
- overvalued
- inferior
- unreliable

How well did our support meet your needs?

- extremely well
- very well
- roughly well
- rather not well
- not well at all

How well was your corresponding support contact reachable?

- very easy to reach
- easy to reach
- generally reachable
- difficult to reach
- not available

Have you already used the Performance Engineering Support before?

- Yes, I have already used the support before.
- No, this was the first time I ever used the support.

How likely is it that you would want to make use of the Performance Engineering Support again?

- extremely likely
- very likely
- indecisive
- rather unlikely
- very unlikely

Do you have further question, suggestions or would you like to leave us feedback? Feel free to use the comment section below: Feel free to comment here

Thank you for your time and participation in our survey. We will work on fulfilling your wishes.
Kind regards, Your Performance Engineering Support Team

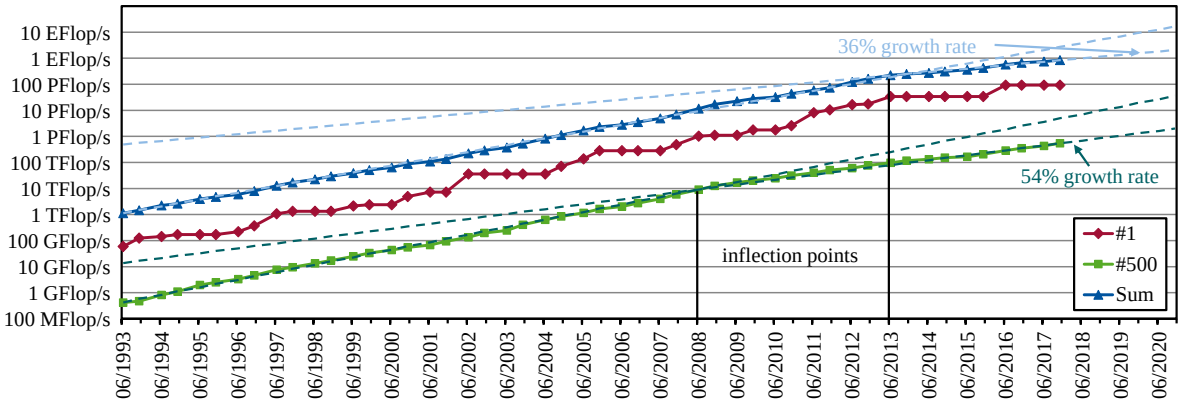

Figure B.2: Top500 performance development over time. Dashed lines approximate yearly growth rate similar to [19]. Sum systems represent 500× the average.

B.4 3-Element Cost Model

The 3-element (3E) cost model covers the three most relevant factors to investigate the impact of a PE process on TCO, i.e., costs for hardware, energy and brainware. The aim of this 3E cost model is to allow setup cost elements with reasonable effort, but avoiding a comparison of HPC centers instead of the effect of the PE process that was applied at a specific center.

To further foster the focus on the quantification of the effectiveness of the PE process (and not the HPC centers themselves), we collect cost values across the three contributing sites and only present average values. These values can also serve as basis to apply the 3E cost model to other local HPC sites if specific data, e.g., on hardware costs, is missing.

1. **Hardware Costs** C_{HW}: These costs describe the acquisition costs for compute nodes, network and, if applicable, storage. The costs can be derived from an HPC cluster procurement by breaking down overall costs in average per-node costs. Alternatively, they can be taken from a vendor's successive supply contract, if available, or otherwise from vendor's list prices.

    - **Wear and Tear**: To do a fair comparison across compute nodes that are in operation for different lengths of time, we include guidelines for wear and tear with respect to the acquisition costs. Here, the basic idea is to reduce the initial acquisition costs for each year of operation. To get a reasonable approach, we investigate German depreciation rules and extend these to HPC use cases: The German Federal Ministry of Finance assumes a linear depreciation along 7 years for big computers [4]. However, a linear depreciation might not correctly reflect the reality in the fast-developing HPC world. For example, the performance development given by the Top500 list shows an exponential average performance growth of 36 % per year (compare Fig. B.2). Assuming a direct correlation between performance and acquisition cost, an approach of declining-balance depreciation (*degressive Abschreibung*) is more appropriate where 36 % of performance growth corresponds to an annual depreciation of roughly 26 %. This value is in line with former declining-balance percentages given by the German Federal Ministry of Finance, lastly specified with a maximum of 25 %. Thus, as default, we reduce the initial acquisition costs by 26 % per year.
    - **Maintenance**: Hardware maintenance costs can often be defined as certain percentage of the initial hardware (net) acquisition costs. If these costs cannot be directly derived, we assume 10 % as default value. Note that maintenance contracts cover a limited time period only.

2. **Energy Costs** C_{EG}: The energy costs reflect the expenses that are related to the power consumption of user jobs over their runtime. If power meters are not available on a fine-granular basis at the HPC center, the measurement results of power hardware counters can be retrieved instead. For example, Intel processors provide the RAPL (Running Average Power Limit) interface for this purpose where values for CPU and DRAM power consumption must be aggregated.

    - **Power Consumption & Energy**: Since energy is the product of the job's power consumption and its runtime, we need to collect both values and integrate over the time component. Alternatively, the

(estimated) average power consumption of the job can be a substitution for detailed measurements. The easiest way is the usage of tools such as LIKWID that directly provide the job's total energy consumption in joule. If the collection of energy data per job is not feasible or recommended, e.g., because of job-shared compute nodes, we use as estimation the corresponding power shares. They can be based on the number of cores used to overall power consumption on the node.

- **Electricity**: Although HPC power loads can fluctuate by a few megawatts, electricity costs are usually not managed by dynamic pricing programs in Europe [16]. Thus, we assume a fixed cost for electricity in terms of €/kWh. As default value, we take 0.15 €/kWh as stated in [23, 24].
- **Power Usage Effectiveness (PUE)**: PUE is a simplified key performance indicator to determine the energy efficiency of an HPC center. It describes the ratio of overall power consumed by the HPC center and the power needed to run the computer infrastructure. Thus, it captures cooling needs, power losses and inefficiencies of electrical components, and in turn reflects the occurrence of increased energy costs. The target is to get a PUE close to 1. Although discussion exists on its meaningfulness and comparability, it is a commonly-used metric that we will also use.

3. **Brainware Costs** C_{BW}: Brainware costs include the effort needed for HPC-related activities such as supporting developers in HPC tuning processes, as well as performance engineering and optimization of HPC applications (sample list of these activities can be found in [23, p. 94]). These costs depend on the HPC application under investigation and the skills of the performance engineer and tuner. Applying a PE process and optimization to an HPC application means additional effort that usually pays off in the long term by improved resource efficiency [3], [25], [24].

    - **Human Effort in Person-Hours**: While numerous software metrics related to effort are known from the software engineering domain, most of them do not directly fit to HPC-related activities [26]. Therefore, we focus on the basic development time metric that represents human effort in person-hours [26], [23, pp. 94]. It can be tracked manually (e.g., developer diaries also in the form of our ticket tool or PE logbook), automatically (e.g., HackyStat) or semi-automatically (e.g., EffortLog [14]). Wherever possible, we rely on the semi-automatic approach using EffortLog since it combines many advantages [23, pp. 214] and showed reasonable results in previous effort studies [26], [15]. Nevertheless, it must be noted that collected HPC development time data is highly dependent on different impact factors such as, e.g., the application's algorithm, the hardware and software environment, the skills and pre-knowledge of the developer [26]. While research approaches started to quantify these impact factors in HPC [26], [23], for now, we lump the effort data all together.
    - **Salary**: Given the effort on conducting the PE process in person-hours, brainware costs are computed as the product of these person-hours and a person's salary in €/h. Since salaries for scientists differ across German federal states (or countries), we use the DFG personnel rates [6] as reference value. The DFG approves funding of 64,500 € per year for doctoral researchers (or comparable) in 2018. Secondly, we assume 210 working days per year according to the European Commission's CORDIS [9] when excluding vacations and sick days. We further assume 8 working hours per day. Hence, we take as default value 38.39 €/h as salary for HPC-related activites.

Time Components All cost factors mentioned above are either dependent on a certain time span or occur in an one-time effort. We distinguish between runtime, project time and the system's lifetime and provide guidelines on valuing one-time costs over specific periods of time to enable a reasonable cost-benefit analysis.

- **Core Hours**: One core hour (core-h) corresponds to the capacity consumption of a job running for one hour on one core of an arbitrary compute node. Core-h are not directly comparable across computer architectures due to differing hardware features such as clock frequency. Thus, they need some kind of reference hardware information to be meaningful. Furthermore, accelerator-based compute nodes often only count core-h of the respective host processor, hence, not taking core-h of e.g. GPUs into account. Nevertheless, core-h is a very common metric that we will also use to accommodate costs on a fine-grained basis.
- **Runtime**: The runtime of an application is the most important component for evaluating the resource efficiency achieved by applying the PE process to this job. Furthermore, it directly affects the job-specific energy costs.

- **Project Time**: We assume that cluster resources are managed based on project applications [13, 17, 30]. There, researchers have to justify the required cluster resources by their scientific work. The corresponding project proposals are scientifically reviewed and assessed before acceptance [28, p. 21]. For simplicity, we assume that a project refers to a single simulation application that runs in a 'typical' configuration (although we actually have more variety within a project). The project time refers to the period of time that a project is assigned to run on the cluster. For now, we assume one year as default value. Nevertheless, we typically see follow-up compute applications of projects so that projects benefit longer than one year from runtime improvements. We will cover this in further refinements of our cost model. Project applications usually request a certain number of core hours that must be explained by the amount of scientific work to accomplish. Tier-2 centers track the number of used core hours within the project, as well as the remaining core hours. The latter serve as basis to amortize brainware costs across a longer period of time since brainware costs would not pay off if we fully account it to a single program run after the PE process application. Tier-3 centers might not ask for and record project core hours. In this case, we amortize brainware costs across 10,000,000 core hours.
- **System's Lifetime**: The system's lifetime refers to the time span that a cluster is up and running. The time validity of hardware maintenance contract, e.g., 5 years, usually determines the end of system lifetime. Nevertheless, the actual lifetime may also exceed this time span. Then, broken hardware components may not be replaced by new ones, but reduce the availability of hardware capacity.
- **Year of Acquisition**: For fair comparisons across compute nodes of different age, we apply a declining-balance depreciation. For this, it is essential to know the acquisition year of the hardware component. In the actual year of acquisition, we account for the full purchase costs of the hardware.

Computation of Costs With the given cost and time components, we compute the *costs per application run* based on the project time. This metric focuses on the scientific work and output generated by a certain application and, hence, is in line with common rules on how to request compute time in project proposals. Furthermore, this metric enables the comparison between the price of a job run before and after the application of a structured PE process that improved the job's performance. The basis of our cost computations are performance snapshots of a specific program before and after the application of a PE process.

A project starts at time pt_0 with a certain amount of overall assigned core hours $core\text{-}h_0$. We assume the project describes application runs in 'typical' configuration defined as app_0. The project ends at time pt_e with assumable no core hours left, i.e., $core\text{-}h_e = 0$. The point in project time where a structured PE process will be readily applied is referred to as pt_{PE} and results in the program configuration app_{PE}. For simplicity, we assume that the application of a PE process will occur transparently with respect to the project time. That means that we assume that the program will be continuously executed up to pt_{PE} in the 'original configuration' app_0 and does not pause or use interim solutions during the tuning cycle of the PE process. At point pt_{PE}, $core\text{-}h_{PE}$ core hours will be used up by the project's jobs while $core\text{-}h_{rem} = core\text{-}h_0 - core\text{-}h_{PE}$ core hours are remaining in the project.

The (performance) snapshots before and after the application of a PE process reveals the job's configurations app_0 and app_{PE}. The job configuration contains the runtime t (wall clock time), the number of cores used p and the amount of energy needed to run the job. Here, because of accounting reasons, the *number of cores used* given by p always refers to the number of cores reserved by the batch scheduler for the job. This is mainly due to node-exclusive jobs where the program might not use all available cores, however, the compute node must still be blocked for other users. Furthermore, the performance snapshots are dependent on a specific hardware configuration that includes, e.g., processor type and core count per node. Since the application of a PE process might reveal a performance improvement when running on another hardware (architecture), the hardware configuration may differ for the original setup—given as hw_0—and the configuration hw_{PE} used after performance engineering. Nevertheless, we take the most likely setup with $hw_0 = hw_{PE}$ as default.

Knowing the recorded core hours $core\text{-}h_{PE}$ and configuration app_0, we can compute the number of applications runs r_{PE} at time pt_{PE} by

$$r_{PE} = \left\lfloor \frac{core\text{-}h_{PE}}{t(app_0) \cdot p(app_0)} \right\rfloor . \tag{B.1}$$

Taking the original application configuration app_0 as basis, we estimate the remaining number of application runs r_{rem} that can be executed from project time pt_{PE} to pt_e with $core\text{-}h_{rem}$ core hours left by

$$r_{rem} = \left\lfloor \frac{core\text{-}h_{rem}}{t(app_0) \cdot p(app_0)} \right\rfloor = \left\lfloor \frac{core\text{-}h_0 - core\text{-}h_{PE}}{t(app_0) \cdot p(app_0)} \right\rfloor . \tag{B.2}$$

We further assume that a tuned program, i.e., it is in configuration app_{PE}, will need the same amount of application runs r_{rem} as the original configuration. The benefit lies in a better resource efficiency as targeted by our ProPE process.

Now, to compute the actual costs, we first investigate the three elements of our cost model (hardware costs C_{HW}, energy costs C_{EG} and brainware costs C_{BW}) while abstracting them to the unit of €/core-h. From that, we derive the costs per application run (1) for a single execution (as reference value) and (2) for all remaining program executions r_{rem}. For the hardware costs C_{HW} per compute node, we take the price of a compute node in the year of accounting by including the annual tear and wear of 26 % and adding the annual hardware maintenance costs. Giving an annual per-core capacity of 365 days × 24 hours, a system availability α (default $\alpha = 80\,\%$) and the number of cores available per node $p(hw_0)$ or $p(hw_{PE})$ in the respective hardware configuration, the compute node costs are presented as

$$c_{HW}(hw_0) = \frac{C_{HW}(hw_0)}{\alpha \cdot 365 \cdot 24 \cdot p(hw_0)} \left[\frac{€}{\text{core-h}}\right] \quad \text{or} \quad c_{HW}(hw_{PE}) = \frac{C_{HW}(hw_{PE})}{\alpha \cdot 365 \cdot 24 \cdot p(hw_{PE})} \left[\frac{€}{\text{core-h}}\right]. \tag{B.3}$$

For the energy costs C_{EG} per application run, we collect information on the required energy consumption in kW h and multiply it with the electricity price and PUE of the HPC center. For consistency, we express these costs as costs per core-h by incorporating the required core hours for a single application run. This results in

$$c_{EG}(app_0) = \frac{C_{EG}(app_0)}{t(app_0) \cdot p(app_0)} \left[\frac{€}{\text{core-h}}\right] \quad \text{or} \quad c_{EG}(app_{PE}) = \frac{C_{EG}(app_{PE})}{t(app_{PE}) \cdot p(app_{PE})} \left[\frac{€}{\text{core-h}}\right], \tag{B.4}$$

for the original configuration app_0 or the tuned configuration app_{PE}. Third, the brainware costs C_{BW} are derived by the product of person-hours spent for applying the PE process and the corresponding hourly salary, e.g., as given by the DFG funding guidelines. They are amortized over the remaining core hours needed to accomplish the scientific work justified in the project proposal. These are given by the remaining application runs r_{rem} and the core hours required per run in configuration app_{PE}. Expressing this in monetary value per core hour, we get

$$c_{BW}(app_{PE}) = \frac{C_{BW}(app_{PE})}{r_{rem} \cdot t(app_{PE}) \cdot p(app_{PE})} \left[\frac{€}{\text{core-h}}\right]. \tag{B.5}$$

Overall, the computation of the cost per single application run C_s before and after the application of a PE process resolves to

$$C_s(app_0) = \quad t(app_0) \cdot p(app_0) \cdot \quad (c_{HW}(hw_0) + c_{EG}(app_0)) \tag{B.6}$$

$$C_s(app_{PE}) = \quad t(app_{PE}) \cdot p(app_{PE}) \cdot \quad (c_{HW}(hw_{PE}) + c_{EG}(app_{PE}) + c_{BW}(app_{PE})). \tag{B.7}$$

Please remind that C_s is just a reference value that does not cover the full amortization of the additional brainware costs. Instead, we evaluate the *remaining project costs* C_{rem} based on the number of remaining application runs r_{rem} for an integral comparison of the value of an application of the PE process. For that, we put the difference $\Delta C_{rem} = C_{rem}(app_0) - C_{rem}(app_{PE})$ in perspective computed by using (B.2) and

$$C_{rem}(app_0) = \quad r_{rem} \cdot \quad C_s(app_0) \tag{B.8}$$

$$C_{rem}(app_{PE}) = \quad r_{rem} \cdot \quad C_s(app_{PE}). \tag{B.9}$$

If it yields that $\Delta C_{rem} > 0$, the application of a PE process paid off in terms of improved resource efficiency and cost effectiveness. It is noteworthy that an amortization over the remaining core hours within the *same* project is very challenging to achieve due to the short return cycles. However, as mentioned earlier, numerous researchers file follow-up projects so that the return of investment usually extends to these follow-up applications as well. For example, compute projects in the JARA-HPC and GCS context (running at the Forschungszentrum Jülich (Tier-1) or at the RWTH Aachen University (Tier-2)) have an average project time [1], i.e., including follow-up projects, between roughly 20 months to 30 months. From these numbers, we suggest to extend the amortization time to 24 months (instead of currently 12 months per default). This time frame is in-line with previous investigations where we assumed that applications benefit from tuning activities for two years [3]. We will incorporate these guidelines in future refinements of our cost models.

[1] Averages cover the time period May 2012 to April 2018, or July 2009 to April 2018, respectively.

Hardware Costs				Energy Costs		Brainware Costs
Acqu. [€]	Acqu./core-h $\left[\frac{€}{\text{core-h}}\right]$	Wear & Tear [%]	#cores	Elec. $\left[\frac{€}{\text{kWh}}\right]$	PUE	Salary $\left[\frac{€}{\text{person-h}}\right]$
7,200	0.05	26	20	0.16	1.4	38.39

Table B.1: Averaged or default cost components. Averages are computed from real data provided by the participating partners across numerous (non-accelerated) HPC systems. Acquisition costs are brutto values including taxes and maintenance. They represent costs per compute node for the (initial) year of acquisition.

	app_0	app_{PE}
runtime t [h]	3	2
energy [kWh]	3.6	2.4
cores used p	80	80
core-h per run	240	160
years since acqu.	1	1
brainware C_{BW} [€]	-	1,500
hardware costs c_{HW} $\left[\frac{€}{\text{core-h}}\right]$	0.03801	0.03801
energy costs c_{EG} $\left[\frac{€}{\text{core-h}}\right]$	0.00336	0.00336
brainware costs c_{BW} $\left[\frac{€}{\text{core-h}}\right]$	-	0.00938
cost per single run C_s [€]	9.9288	8.1200
remaining project costs C_{rem} [€]	9928.8	8120.0

Table B.2: Assumptions for application-based cost-relevant values (with $hw_0 = hw_{PE}$).

Case Study To illustrate the application of our 3E cost model, we provide a theoretical case study with sample cost computations. This case study is based on averaged real cost numbers across the three participating partner centers as stated in Table B.1. Furthermore, we assume that a simulation application in configuration app_0 has been optimized as part of a PE process and, hence, transferred into configuration app_{PE}. An overview of our assumed and computed values can be found in Table B.2. By applying a PE process, a runtime reduction from 3 hours to 2 hours of wall clock time is assumed where we use the same hardware before and after the PE process, i.e., $hw_0 = hw_{PE}$. The application uses (either way) 4 compute nodes with each 20 cores per node resulting in $p = 80$. When taking 240,000 core hours as remaining project capacity, this results in the number of remaining runs $r_{rem} = 1{,}000$. We further assume that the hardware was acquired one year ago so that we account for wear and tear of 26 % giving the absolute hardware costs of 5,328 € per compute node (including taxes and maintenance). The hardware is operated with a system availability of $\alpha = 80\,\%$ and consumes 300 W per node and over the duration of the application runs. Finally, we arbitrarily estimate the effort needed for the application of the PE process as a 40-hour week of a full time employee which gives us roughly 1,500 €. Doing the math, we can compute hardware, energy and brainware costs per core hour. From that, we derive the costs per single run and the remaining project costs in comparison for app_0 and app_{PE} (see Table B.2). In this example, we see that the application of a structured PE process would save 1.808,8 € per single run and 1,808.8 € for the remaining project time. Thus, putting brainware into performance optimization can deliver a return on investment and improves resource efficiency.

Appendix C

Train the Trainer Course Feedback

C.1 Anonymous Questionnaire

This is intended for the Train-the-Trainer (TtT) course participants who already completed the course. The aim of this anonymous questionnaire is to collect feedback from the past TtT course participants and improve the TtT program based on their recommendations. The survey, as any other anonymous surveys, has to carry a legal note which needs to be made clear to the interviewees before the interview or in the beginning of the questionnaire. The legal note has to explain how collected data will be processed and stored and has to be in accordance with the privacy policy of the institution conducting the survey. The legal note can be as following: ”By completing this questionnaire (or completing this survey) you agree to participate in this survey. Your answers will be stored separately from your contact details, will remain anonymous, and processed in accordance with the privacy policy of the “‘xyz institution”’.

Qualitative survey:

1. In what way the TtT course helped you to prepare to teach an HPC course?
2. What skills/subjects you sought to learn at the TtT course?
3. How would you develop or improve the pedagogic material?
4. What are the main skills a student needs to be taught/to possess in order to be able to teach HPC courses?
5. Do you think you are sufficiently prepared for teaching after participating in the TtT course?
6. Did you learn/had to deal with how to handle difficult situations in a teaching environment? If yes, please specify:
7. What would you like to learn/teach about handling such a situation?
8. What information was most and least useful for you in the TtT course?
9. How would you develop or improve the teaching process for the TtT course?
10. Do you have any recommendations for instructors of the Train-the-Trainer course? If yes, please specify:
11. Have you taught an HPC course before participating in the TtT course? If yes, how many hours (approx.)?
12. Have you been teaching after attending the TtT course? If yes, how many hours and students (approx.)?
13. Would you develop or improve the TtT practical sessions setup to improve it? If yes, how?
14. How did you hear about this course?
15. Would you recommend this course to others?

Bibliography

[1] PeCoH - Performance Conscious HPC. https://wr.informatik.uni-hamburg.de/research/projects/pecoh/, Accessed September 2018.

[2] The HPC Certification Forum. https://www.hpc-certification.org/, Accessed September 2018, 2018.

[3] Christian Bischof, Dieter an Mey, and Christian Iwainsky. Brainware for green HPC. *Computer Science - Research and Development*, 27(4):227–233, 2012.

[4] Bundesministerium der Finanzen. AfA-Tabelle für die allgemein verwendbaren Anlagegüter (AfA-Tabelle ÄV). https://www.bundesfinanzministerium.de/Content/DE/Standardartikel/Themen/Steuern/Weitere_Steuerthemen/Betriebspruefung/AfA-Tabellen/2000-12-15-afa-103.pdf, Accessed May 2018, 2000.

[5] Chris Churilo. InfluxDB vs. Elasticsearch for Time Series Data & Metrics Benchmark. https://www.influxdata.com/blog/influxdb-markedly-elasticsearch-in-time-series-data-metrics-benchmark/, Accessed November 2019, 2018.

[6] Deutsche Forschungsgemeinschaft. DFG Personnel Rates for 2018. http://www.dfg.de/formulare/60_12/60_12_en.pdf, Accessed April 2018, 2018.

[7] Jan Eitzinger. GitHub RRZE-HPC: HPC Job Database. https://github.com/RRZE-HPC/HPCJobDatabase, Accessed December 2019, 2019.

[8] Jan Eitzinger. GitHub RRZE-HPC: The Performance Logbook. https://github.com/RRZE-HPC/ThePerformanceLogbook, Accessed December 2019, 2019.

[9] European Commission - Community Research and Development Information Service (CORDIS). Guide to Financial Issues relating to FP7 Indirect Actions. http://ec.europa.eu/research/participants/data/ref/fp7/89556/financial_guidelines_en.pdf, Accessed April 2018, 2014.

[10] High Performance Computing Center Stuttgart (HLRS). Parallel Programming Workshop (Train the Trainer). http://www.hlrs.de/events/detail-view/2018-10-15-ttt/, Accessed September 2018, 2018.

[11] Kai Himstedt, Nathanael Hübbe, Julian Kunkel, and Hinnerk Stüben. An hpc certification program proposal meeting hpc users' varied backgrounds. https://www.hhcc.uni-hamburg.de/en/files/hpccp-concept-paper-180201.pdf, 2018.

[12] HPC Carpentry. Teaching basic lab skills for high-performance computing. https://hpc-carpentry.github.io/, Accessed September 2018.

[13] IT Center of RWTH Aachen University. Project-based Management of Resources of the RWTH Compute Cluster. https://doc.itc.rwth-aachen.de/display/CC/Project-based+Management+of+Resources+of+the+RWTH+Compute+Cluster, Accessed June 2018, 2018.

[14] Julian Miller and Sandra Wienke. EffortLog. http://www.hpc.rwth-aachen.de/research/tco, Accessed April 2018, 2016.

[15] Julian Miller, Sandra Wienke, Michael Schlottke-Lakemper, Matthias Meinke, and Matthias S. Müller. Applicability of the Software Cost Model COCOMO II to HPC Projects. *International Journal of Computational Science and Engineering*, 2017.

[16] Tapasya Patki, Natalie Bates, Girish Ghatikar, Anders Clausen, Sonja Klingert, Ghaleb Abdulla, and Mehdi Sheikhalishahi. *Supercomputing Centers and Electricity Service Providers: A Geographically Distributed Perspective on Demand Management in Europe and the United States*, pages 243–260. Springer International Publishing, Cham, 2016.

[17] RRZE of University Erlangen. HPC-Rechenleistung. https://www.dlp.rrze.uni-erlangen.de/

hpc-rechenleistung/, Accessed June 2018, 2015.
[18] Software Carpentry. Teaching basic lab skills for research computing. https://software-carpentry.org/, Accessed September 2018.
[19] Erich Strohmaier, Hans W. Meuer, Jack Dongarra, and Horst D. Simon. The TOP500 List and Progress in High-Performance Computing. *Computer*, 48(11):42–49, 2015.
[20] The Carpentries. We teach foundational coding and data science skills to researchers worldwide. https://carpentries.org/, Accessed September 2018.
[21] J. Treibig, G. Hager, and G. Wellein. Likwid: A lightweight performance-oriented tool suite for x86 multicore environments. In *Proceedings of PSTI2010, the First International Workshop on Parallel Software Tools and Tool Infrastructures*, San Diego CA, 2010.
[22] Jan Treibig, Georg Hager, and Gerhard Wellein. Performance Patterns and Hardware Metrics on Modern Multicore Processors: Best Practices for Performance Engineering. In *Euro-Par 2012: Parallel Processing Workshops*, volume 7640 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2013.
[23] Sandra Wienke. *Productivity and Software Development Effort Estimation in High-Performance Computing; 1. Auflage*. Dissertation, RWTH Aachen University, Aachen, 2017. Veröffentlicht auf dem Publikationsserver der RWTH Aachen University 2018; Dissertation, RWTH Aachen University, 2017.
[24] Sandra Wienke, Dieter an Mey, and Matthias S. Müller. Accelerators for technical computing: Is it worth the pain? a tco perspective. In Julian Martin Kunkel, Thomas Ludwig, and Hans Werner Meuer, editors, *Supercomputing*, pages 330–342, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
[25] Sandra Wienke, Hristo Iliev, Dieter an Mey, and Matthias S. Müller. Modeling the Productivity of HPC Systems on a Computing Center Scale. In Julian M. Kunkel and Thomas Ludwig, editors, *High Performance Computing*, volume 9137 of *Lecture Notes in Computer Science*, pages 358–375. Springer International Publishing, 2015.
[26] Sandra Wienke, Julian Miller, Martin Schulz, and Matthias S. Müller. Development effort estimation in hpc. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '16, pages 10:1–10:12, Piscataway, NJ, USA, 2016. IEEE Press.
[27] Wissenschaftsrat. Strategische Weiterentwicklung des Hoch-und-Höchstleistungsrechnens in Deutschland. https://www.wissenschaftsrat.de/download/archiv/1838-12.html, Accesse d July 2020, 2012.
[28] Wissenschaftsrat. Empfehlungen zur Finanzierung des Nationalen Hoch- und Höchstleistungsrechnens in Deutschland. Technical Report Drs. 4488-15, 2015.
[29] Wissenschaftsrat. Empfehlungen zur Finanzierung des Nationalen Hoch-und-Höchstleistungsrechnens in Deutschland. https://www.wissenschaftsrat.de/download/archiv/4488-15.html, Accesse d July 2020, 2015.
[30] ZIH of TU Dresden. Project Application for using the High Performance Computers. https://tu-dresden.de/zih/hochleistungsrechnen/zugang/projektantrag?set_language=en, Accessed June 2018, 2018.