hpc.bw: An Evaluation of Short-Term Performance Engineering Projects

Johann Antonio Duffek,
Imane Bechalaoui,
Willi Leinen,
Hauke Preuß,
Simon Schlumbohn,
Yannis Schumann
High Performance Computing
Helmut Schmidt University/
University of the Federal Armed Forces
Hamburg, Germany
duffek@hsu-hh.de

Fabian Dethof,
Sylvia Keßler
Engineering Materials and
Building Preservation
Helmut Schmidt University/
University of the Federal Armed Forces
Hamburg, Germany
fabian.dethof@hsu-hh.de

Marie Rathmann,
Jessica Kleinschmidt,
Alexander Kolling,
Sabine Schmidt-Lauff
Continuing Education and
Lifelong Learning
Helmut Schmidt University/
University of the Federal Armed Forces
Hamburg, Germany
marie.rathmann@hsu-hh.de

Andreas Fink
Business Administration, in
particular Business Informatics
Helmut Schmidt University/
University of the Federal Armed Forces
Hamburg, Germany

Marcus Stiemer
Theorectical Electrical Engineering and
Numerical Simulations
Helmut Schmidt University/
University of the Federal Armed Forces
Hamburg, Germany

Matthias Mayr
Institute for Mathematics and
Computer-Based Simulation
University of the Bundeswehr Munich
Munich, Germany

Philipp Neumann
DESY, IT-Department and University Hamburg
High Performance Computing & Data Science
Hamburg, Germany

***Abstract* — Increasing amounts of data and simulations in scientific areas enforce the need of improved software performance. The maintaining scientific staff is often not primarily trained for this purpose or lacks personnel and time to address software performance issues.**

A particular aim of the dtec.bw-funded project hpc.bw is to tackle some of these shortcomings. A pillar of the hpc.bw agenda is the offer of a low-threshold consultancy and development support focused on performance engineering. This paper provides an insight on our related activities.

We illustrate the structure of our annual calls for short-term performance engineering projects, we outline our results at the example of the performance engineering project 'benEFIT-Numerical simulation of non-destructive testing in concrete', and we draw a first conclusion on the current procedure.

***Index Terms* — hpc.bw, HPC, performance engineering, parallel I/O, Fortran**

I. HPC.BW - Motivation and Introduction

The core objective in high performance computing (HPC) is the development of efficient software, that is often executed on supercomputers, in order to achieve computations within reasonable time frames and/or within resource bounds. Some relevant examples are optimisation problems in logistics and production, numerical simulations in engineering, or image recognition/analysis via machine learning with real-time demands for medical diagnostics.

However, software is often not utilising the full potential of compute resources. Reasons for this are, e.g. limited time for software optimisations or missing expertise in HPC. This results in a reduced productivity in the use of digital research techniques.

Our project hpc.bw aims to strengthen HPC-related research at the two universities of the Federal Armed Forces and to enforce the transfer of HPC knowledge across the various fields. Based on these project activities, hpc.bw strives

1) to strengthen the research and development in respective research fields,
2) to promote the interdisciplinary exchange between HPC-related problems,
3) to derive and answer new HPC-related research questions from the different field specific problems, and
4) to establish a common HPC competence platform.

A key to strengthen research and development lies in performance engineering (PE), i.e. investigating and optimising programs for (parallel) performance.

In the following, after introducing the key activities of the project in 2023/2024 in Sec. II, the concept of short-term

performance engineering projects, that have been established in hpc.bw, is explained in Sec. III. A particular realisation of this concept is laid out in Sec. IV. We close with a short summary and outlook to future project activities in Sec. V.

II. HPC.BW: Project Activities in 2023/2024

In the first two years of the hpc.bw project a container-based HPC centre was installed, including supercomputer HSUper, communication channels and community-building measures (such as a multiplicator programme, newsletters, seminar series "Computation & Data") were established and training concepts (such as workshops and performance engineering projects, cf. Sec. III for the latter) were created. The focus in the last one and a half years (2023-mid of 2024) was put on consolidation of these and running research activities (such as research on HPC for optimisation problems in logistics).

Various groups have adopted to using the HPC system HSUper, an initial TOP500 supercomputer (i.e. ranked 339 among the 500 fastest supercomputers in the world at the time of its inauguration). More than 180 users currently have access to the machine, more than 100 users rely on HSUper currently in their research and actively submit compute jobs. This also includes researchers funded directly from hpc.bw, conducting research on efficiently solving optimisation problems in logistics. This resulted in a total use of 73% of the available compute resources, that was measured over the past months (February–June 2024), demonstrating a currently well-defined size of the compute system and a functioning software stack for a wide range of applications (such as from fluid dynamics, materials science, algorithms research and numerical mathematics, or optimisation in logistics). To enable researchers to use HSUper, three workshops were conducted: two full-day workshops addressing beginners in HPC (held on 28 Sep 2023 and on 18 Apr 2024 at HSU) as well as one full-day workshop for more advanced users of HPC resources (held on 19 Apr 2024 at HSU).

To provide HPC resources also for researchers with less affinity to supercomputing and computer science in general, the *Interactive Scientific Computing Cloud (ISCC)* has been established in summer 2023. It consists of ten dual-socket Intel(R) Xeon (R) Platinum 8360Y nodes (which is the same architecture as used in HSUper) with 256 GB RAM each, as well as of two more nodes of the same type, but with 1TB RAM, additional 2TB local scratch storage and 8 NVIDIA A30 GPUs for more compute-heavy machine learning tasks. In contrast to HSUper, which runs a current Linux distribution and relies on job scheduling via SLURM, the ISCC uses a virtualisation environment built on top of VMware's vSphere. This allows users to launch a virtual machine with Windows or Linux and run it including the entire desktop and GUI environments on (single) high performance compute nodes. In the mean time, more than 60 virtual machines have already been created and are in use.

Moreover, the HPC portal[1] has been launched to provide a virtual competence platform for the identified user groups at the universities at the Federal Armed Forces and beyond. The platform will be extended in the future with e-learning, more videos on selected HPC topics and continuous documentation of the compute platforms (such as HSUper and ISCC).

Moreover, hpc.bw staff actively reached out further to HPC and education communities: various renowned speakers were invited to the hpc.bw-empowered seminar series on "Computation & Data", the project was represented, amongst others, at the NHR conference 2023, the Section Conference for Adult Education 2023 as well as at the ISC HPC congress 2024. From 24-26 June 2024, hpc.bw organised the European Trilinos User Group Meeting (EuroTUG) at HSU, with ca. 20 participants from more than ten European research institutions.

III. Short-Term Performance Engineering Projects

The project hpc.bw organises short-term performance engineering projects in the form of easily requestable consultancy on HPC-related questions. Their aim is to provide help and support to users confronted with software performance issues. Figure 1 provides a schematic overview to the following project characterisation.

A. Application Process

Eligible groups for the short-term PE projects are all departments at the Universities of the Federal Armed Forces. Annually, a call for PE projects is launched that groups can apply for. The application scheme is straightforward, with application forms not exceeding 2-3 pages. The application form asks to characterise

1) the applicant's research project and its relation to HPC,
2) the existing software to be optimised,
3) the problems faced with the current software,
4) the anticipated gains in performance improvement for the software, required to answer related research questions, and
5) the necessary support needed as well as the estimated project duration.

Overall support must not exceed the equivalent of six person months of an HPC-affine scientific co-worker from the hpc.bw project. The applicants are additionally asked to categorise their HPC-related knowledge into 'no', 'basic', 'advanced', or 'expert'.

This deliberately simple application procedure lowers the barrier for candidates without prior knowledge in HPC, and it provides us with a rough overview of the underlying problem and expected workload.

The applications are internally reviewed by the hpc.bw multiplicator team, which represents all involved institutions/departments of the Universities of the Federal Armed Forces, and they are externally reviewed by one more person. After review, projects are selected, and, if necessary, adapted in their extent and duration. Table I provides an overview on the granted projects, their extent, and involved personnel.

[1] https://portal.hpc.hsu-hh.de/

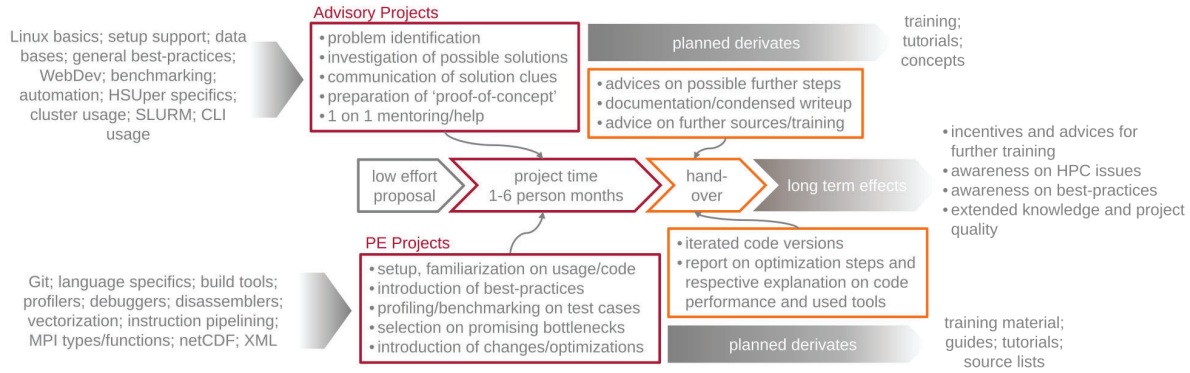


FIGURE 1. A SCHEMATIC OVERVIEW OF THE SHORT-TERM PERFORMANCE ENGINEERING PROJECTS OF 'HPC.BW'

TABLE I. Overview on the granted short-term PE projects. Project Count: total number of supported projects. Granted Person Months: total number of person months allocated in the respective call. Involved Staff: number of staff members from hpc.bw and the chair for HPC at HSU involved in the PE projects

Year	Project Count	Granted Person Months	Involved Staff
2022	7	12	4
2023	4	12	3
2024	6	10	6

B. Project Implementation

Kick-off meetings are scheduled with the applicants of the accepted projects within a month after the submission deadline. Their purpose is to

1) introduce the involved personnel to each other,
2) gain a better insight into the outlined problem,
3) understand expectations and fix the scope of the project work,
4) establish and ask for further required prerequisites (such as software stacks, source code and compile scripts, test cases to evaluate performance gains),
5) define the communication channels form of joint project work (e.g. defining meeting schedules or organising time line of the project), and to
6) reiterate the project conditions.

The project conditions entail the contributions of material and abstracts for reports and publications as well as the acknowledgement of significant contributions.

Our staff – scientific co-workers from the project hpc.bw and members of the chair for HPC – then proceed to familiarise themselves with the project and work on the defined problems. A synchronisation is conducted according to the project nature and according to the preference of the involved people.

Each project is usually concluded by a spin-off meeting, presentation, and/or report to transfer our acquired findings to the project partner. An additional one-page report with a brief description of the measures and their outcome is written up by the involved project members to document the results.

Our intention is furthermore to simplify the typically encountered problems and to track lessons-learned in order to reuse them in training materials afterwards.

C. Typical encountered Projects

The short-term PE projects can be usually classified into two kinds of projects.

a) The classical PE projects.

These are projects requiring the optimisation of some existing code. An example of such project and its outcome is given in Sec. IV. Their process is roughly as follows:

1) Request of source code access and local setup of the project.
2) Familiarisation with the source code and project.
3) Optional further/advanced preparation of the project.
4) Repeated iterations of profiling and adaptations of the program/source code/software environment.
5) Documentation of the changes and program hand-over.

The ideal outcome is a list of required (or already completed) program modifications and a measurable outcome/improvement. These improvements range from negligible changes (e.g. 5% runtime reduction) to runtime reductions of over 50%. The encountered problems range from a proper software setup, over the conscious use of libraries and system resources, to in-depth PE techniques applied to user-specific (or even third-party) code such as vectorisation and pipelining.

b) The advisory projects.

These projects have a wide variety of appearances and range from 'simple on-boarding help' to use an HPC cluster, over the setup and use of a particular compute- or data-intensive software, to the full build of a proof-of-concept. These are generally handed in by beginners.

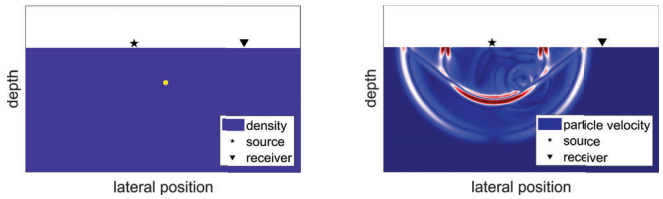
The programming languages covered in our projects so far include, amongst others, Python, Matlab, Fortran, and C++ with project sizes from 400 up to 75,000 lines of code.

IV. BENEFIT

The project 'benEFIT - Numerical simulation of non-destructive testing of concrete' is one of the classical PE

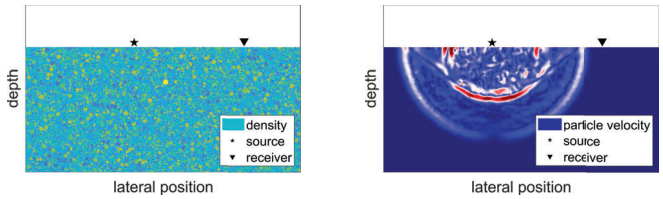
projects (see Sec. III-C) and was granted three person months of attention on our first short-term PE project call. Its subject was a self-written Fortran code of approximately 5,000 lines of code.

Its purpose is to simulate the elastic wave propagation during two different non-destructive testing methods (ultrasonic testing and impact echo testing). These methods are used to assess the condition of concrete structures, such as buildings or bridges, and detect possible defects inside the inspected concrete structures. The respective software elastodynamic finite integration technique (EFIT) utilises a finite difference staggered-grid method which is able to handle strongly heterogeneous domains more robustly.



(a) homogeneous setup

(b) simulation step of homog. setup



(c) heterogeneous setup

(d) simulation step of heter. setup

FIGURE 2. EXAMPLE SIMULATIONS CONDUCTED WITH BENEFIT

Figure 2 depicts two example calculations conducted with EFIT. Figure 2(a)-(b) depict a slice of a homogeneous simulation with embedded steel reinforcement (yellow point) and easily recognisable wave reflections. Figure 2(c)-(d) depict a heterogeneous - more realistic - scenario with a - therefore - very noisy response. EFIT is used to simulate such ensembles to examine the detectability of defects or other features in concrete using the two aforementioned non-destructive testing methods [1].

As the introduction of absorbing boundaries, here implemented with perfectly matched layers (PMLs), requires the use of many additional variables, two basic versions of the EFIT code exist (with and without PML).

A. Initial State and Project Aim

Such simulations require fine domain resolutions due to the heterogeneous material and many time steps needed for the comparatively long simulated time interval, especially for impact echo testing, which results in high memory and runtime demands. Those had been already addressed by a distributed-memory parallelisation with MPI (Message Passing Interface). The declared project aim was the further reduction of runtime and memory demands of EFIT, as well as the parallelisation of a separated accompanying domain creation tool.

B. Initial Steps and Preparation

Initial steps involved the collapsing of several hard-coded program versions into one source code version, modifiable via configuration parameters. This reduced the lines of code considerably (see Fig. 3 `baseline`). A build setup via GNU Make was introduced to facilitate the build of the different program flavours and to incorporate instrumentation with profiling tools.

Those and all further changes were tracked and documented with the versioning system Git. This had the aim to acquaint our project partner with such a system and to separate the steps into comprehensible changes and document-able iterations (without the need of explicit version copies).

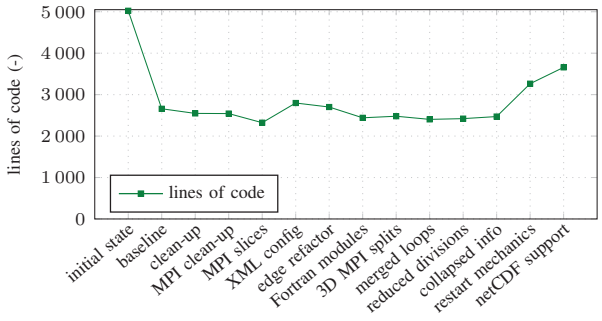
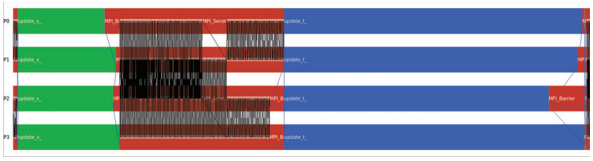


FIGURE 3. THE LINES OF CODE AT MAJOR PROJECT MILESTONES

C. Initial Performance Engineering Process



(a) trace excerpt

Name	TSelf	TSelf	TTotal	TTotal	#Calls	#Calls	TSelf /Call
All Processes							
MPI_Sendrecv	30.359e-3 s		30.359e-3 s		7957		3.81538e-6 s
exchng_t_	117e-6 s		841.988e-6 s		0		n.a.
update_v_	19.873e-3 s		19.873e-3 s		4		4.96825e-3 s
exchng_v_	3.115e-3 s		32.749e-3 s		4		778.75e-6 s
update_t_	57.9932e-3 s		57.9932e-3 s		4		14.4983e-3 s
User_Code	0 s		115.523e-3 s		0		n.a.
MAIN__	94e-6 s		115.523e-3 s		0		n.a.
MPI_Barrier	3.9714e-3 s		3.9714e-3 s		13		305.492e-6 s

(b) accumulated time values of the trace excerpt

FIGURE 4. TRACING RESULTS OF ONE PROGRAM CYCLE

The previously introduced instrumentation was then utilised to acquire a deeper insight of the time-consuming program routines. Figure 4 displays the excerpt of one sampled program cycle (4 MPI processes) acquired via Intel® Trace Analyzer and Collector [2]. The two visible sub-compute steps (green and blue) are encased and interrupted by MPI information interchange steps (red). The table lists the accumulated time of the shown routines (`TSelf`).

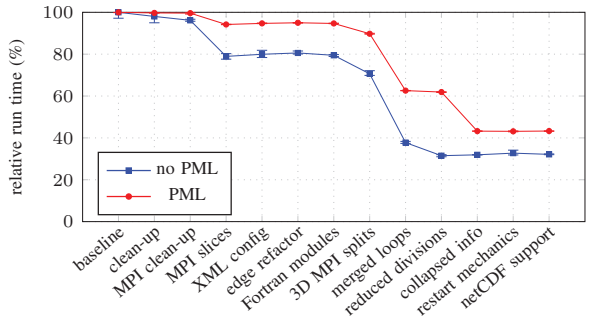


FIGURE 5. PROGRAM RUNTIME AT THE MAJOR PROJECT ITERATIONS AVERAGED OVER THREE RUNS (WHISKERS DENOTE MIN/MAX VALUES)

(a) trace excerpt before the change of MPI synchronisation

(b) trace excerpt after change the of MPI synchronisation

FIGURE 6. TRACE EXCERPT FOR THE `NO PML` PROGRAM VERSION, BEFORE AND AFTER THE STEP `MPI SLICES`

The relatively big amount of time spent in the update routines led to the decision to optimise these routines. Several attempts to apply loop fusion (the merging of different loops) and loop nesting optimisation (the blocking of n-dimensional loops to optimise memory access) led to no immediate significant improvement.

A second approach initiated a structural clean-up of the source code with a simplification of the MPI information exchange. These preceding steps enabled us to easily introduce a proper MPI information exchange for n-dimensional arrays leading to a noticeable runtime reduction (see Fig. 5 up to step `MPI slices`). This reduced runtime can be attributed to the reduction of MPI exchange operations and had a different impact on our two test cases (with/without PML). This is due to an increased share of computation to information exchange for the EFIT feature PML. The reduction and effect of this change is visualised in Fig. 6 (the black lines between the bars represent the exchange operations).

D. Interlude

We recommended and implemented an XML-based simulation configuration in order to avoid hard-coded program variants and program recompilation. This step has been undertaken to facilitate the use of EFIT within an extended/automated context, with the benefit of reproducible runs due to the separation of a static/fixed program from input and configuration files (simulation reproduction possible via configuration and input file). The mature Fortran XML-library TIXI [3] has been chosen as backbone for this implementation.

The changes encompassed refactoring (restructuring) steps of the source code and the transitioning to a more modern Fortran syntax. These changes were undertaken for the sake of a better clarity of the code base and led therefore to no significant performance changes.

The previous steps facilitated the introduction of a generalised domain splitting, which enabled us to choose a more cache-friendly domain splitting scheme (splitting along the dimension of the largest array stride). This led to an additional performance improvement (step `3D MPI splits`).

E. Performance Engineering Process

A second attempt to merge simulation loops resulted in a very noticeable performance gain (step `merged loops`). This is due to a more informed approach about additional circumvention of intermediate variables. The results are a lower memory footprint, better cache utilisation (memory utilisation), and therefore an improved runtime.

Further optimisations included the avoidance of division operations by re-using results from prior calculations, and the reduction of artificially inflated information. The later change resulted in further significant runtime improvements for the `PML` case. This case requires more cache accesses and profits therefore from a reduced data volume. The division-reduction led to no noticeable improvement for the `PML` case as due to more prevalent memory instructions which masked this improvement. The less memory hungry case `no PML` in turn profits slightly.

F. Finishing Touches

A remark from our project partner regarding very long runtimes led to the implementation of a check-pointing mechanism with support of netCDF files [4] to EFIT.

This check-pointing enables the full dump of the state in a time step in the case of an externally triggered signal. This is very useful on compute job scheduled (i.e. Slurm managed) machines. It avoids the necessity to write frequent simulation dumps to the local storage, effectively reducing the potential runtime for cases which require a sparse recording of domain states.

NetCDF reduces the data written from and to disk even further. The ASCII based data-files formerly used by EFIT inflated the required data artificially (besides enforcing a text to data conversion during a read).

The impact of those changes had not been evaluated with the provided test cases. But measurements on in-production problems provided significant runtime improvements from e.g. 72h down to 15h. These effects originated primarily in the avoided heavy use of hard drive reads and writes (the file sizes were reduced by approximately 70%).

The project itself was concluded with a report providing an overview on the changes and their motivation. Additional hints on further steps such as MPI communication-hiding were provided as well.

Work on the domain generation program was skipped due to time constraints.

V. Conclusions and Outlook

In this article, we provided an update on the project status of hpc.bw and detailed the short-term performance engineering projects, that are being carried out within the scope of our project. In particular, we demonstrated how hpc.bw can enable more efficient research at the example of the PE project benEFIT.

While benEFIT can be considered a success story, the overall chances to improve code performance depends on various aspects, ranging from code base and quality over program complexity and problem setting to be solved to actual skills available in the team. While the latter can, of course, built up over time, the structure of performance engineering project calls does not put any constraints on software (open-source, closed-source, quality of documentation, etc.), programming language, and so forth. While this allows us to support an as heterogeneous HPC user group as possible, it increases the need for a wide range of expertise in the PE project team. Therefore, other project activities on building up an actual competence platform with online portal, documentation, and learning material is essential to help users directly and leverage PE projects for individual needs.

Another aspect to critically keep track of in the future is the overall time frame dedicated to the projects. Designed as *short-term* projects, first statistics have shown that the entire work time from hpc.bw PE project staff is distributed to ca. 70% on familiarisation with the considered software/ problem and its preparation for the actual performance engineering, to 10% on project result reporting and only to ca. 20% on actual performance optimisation tasks. The longer a project, the better the ratio of familiarisation-to-optimisation time becomes and the smaller the number of potentially supportable projects becomes (due to limits in the human resources). And the more researchers are already well-educated on programming (e.g. writing modular, well-understandable code with extensive documentation or using code versioning systems such as Git), the smaller the familiarisation phase typically becomes. Careful consideration of these aspects will be required to be monitored in the future.

Acknowledgment

hpc.bw is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr. dtec.bw is funded by the European Union – NextGenerationEU.

References

[1] F. Dethof & S. Keßler, “Design of Concrete Mock-Ups with Complex Defect Scenarios Using Numerical Simulations”, *Journal of Nondestructive Evaluation*, vol. 43(59), 2024, doi: 10.1007/s10921-024-01074-9.

[2] “Intel® Trace Analyzer and Collector 2022.1.0”, Intel®, 2022.

[3] M. Siggel et al., “TIXI 3.3.0”, DLR, 2022, available: github.com/DLR-SC/tixi.

[4] Unidata, “NetCDF-Fortran 4.6.1”, UCAR/Unidata Program Center, 2023, doi: 10.5065/D6H70CW6.